

Bucknell University Bucknell Digital Commons

Master's Theses

Student Theses

2010

The Resonate-and-fire Neuron: Time Dependent and Frequency Selective Neurons in Neural Networks

Himadri Mukhopadhyay
Bucknell University

Follow this and additional works at: https://digitalcommons.bucknell.edu/masters_theses

Recommended Citation

Mukhopadhyay, Himadri, "The Resonate-and-fire Neuron: Time Dependent and Frequency Selective Neurons in Neural Networks" (2010). *Master's Theses*. 23.
https://digitalcommons.bucknell.edu/masters_theses/23

This Masters Thesis is brought to you for free and open access by the Student Theses at Bucknell Digital Commons. It has been accepted for inclusion in Master's Theses by an authorized administrator of Bucknell Digital Commons. For more information, please contact dcadmin@bucknell.edu.

**THE RESONATE-AND-FIRE NEURON: TIME
DEPENDENT AND FREQUENCY SELECTIVE
NEURONS IN NEURAL NETWORKS**

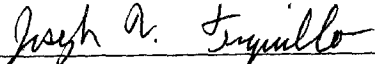
by

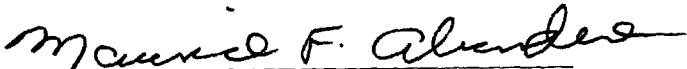
Himadri Mukhopadhyay

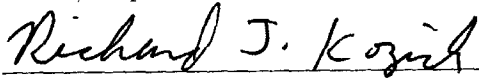
A Thesis


Presented to the Faculty of
Bucknell University

In Partial Fulfillment of the Requirements for the Degree of
Master of Science in the Department of Electrical Engineering

Approved: 
Joseph V. Tranquillo
Thesis Advisor


Maurice F. Aburdene
Chair, Department of Electrical Engineering


Richard J. Kozick
Engineering Thesis Committee


Robert M. Nickel
Engineering Thesis Committee

May 14, 2010

Acknowledgments

This thesis is an exploration of how the insights gained from neuronal modeling may be implemented within neural networks, and how such networks could be trained to learn to perform certain desired tasks. The thesis represents a culmination of our endeavors over the past two years to understand nervous system function, and to reflect on the principles that govern engineered as well as living systems. I am grateful to Bucknell University and the Department of Electrical Engineering for giving me the opportunity to be involved in research, and to undertake this project. I express my heartfelt gratitude to the department faculty for teaching us the principles of engineering and the methods of engineering analysis. Also, a special thank you to Dr. Richard Kozick and Dr. Robert Nickel, for being our professors in *Signal Processing* and *Control Theory*, as well as for serving on my thesis committee.

Through my graduate studies, I have strived to understand the interplay between theoretical neural modeling and the biological principles that govern neural science. Thus I wholeheartedly thank Dr. Elizabeth Marin, our professor in *Molecular Biology*, and Dr. Owen Floody, who taught us *Neural Plasticity*. I am truly grateful that they gave me the opportunity to study graduate level biology in their classes, and I sincerely thank them for the knowledge that they imparted me with.

I first encountered neuronal signaling as a senior year undergraduate student of electrical engineering, in Dr. Joseph Tranquillo's class on *Neural Signals and Systems*. Since then I have had the immense privilege of studying neural signaling and neural networks under Dr. Tranquillo's supervision. The range of questions and ideas that Dr. Tranquillo inspired me to ponder have provided the motivation for this thesis, and for my graduate studies at Bucknell in general. Thus my heartfelt gratitude to Dr. Tranquillo for introducing me to neuronal signaling and neural networks, for guiding me in my endeavors to understand nonlinear dynamics, for initiating my curiosity

regarding the relationship between structure and function in biology, and of course, for being my thesis advisor.

Contents

Abstract	x
1 Introduction	1
1.1 Neurophysiological Considerations	2
1.2 Neural Networks - An Overview	5
1.2.1 Neuron models with static activation functions	6
1.2.2 The Integrate-and-Fire Neuron	8
1.2.3 The Learning Procedure	14
1.3 Biophysical Neural Modeling	15
1.4 Neuronal selectivity and the derivation of tuning curves	19
1.5 Frequency and Time Dependence in Neural Networks	22
1.6 Project Motivation	23
2 The Resonate-and-Fire Neuron	24
2.1 System definition of the resonate-and-fire neuron	25

2.2	Dynamics of the Resonate-and-Fire Model	28
2.3	Application of a single stimulus to resonate-and-fire neuron	29
2.4	Application of a pair of input stimuli to the Resonate-and-Fire Neuron	33
2.5	Parameters and Possible Configurations of Resonate-and-Fire Neuron Model	35
3	The Classification of Logic Gates and The XOR Problem	41
3.1	The "OR" and "AND" problems	42
3.1.1	Single Stimulus Solution to the "OR" and "AND" problems with the Resonate-and-Fire Neuron	42
3.2	The XOR Problem	48
3.3	Application of a train of stimuli to the Resonate-and-Fire neuron . .	51
4	The Temporal Backpropagation Algorithm	53
4.1	Segmenting the training procedure in resonator neural networks . . .	56
4.1.1	Adapting RAF neural networks	57
4.2	Discrete Time Implementation	58
4.3	Computation of the local gradient	60
4.3.1	Adapting the synaptic weights	64
4.3.2	Adapting the synaptic delays	65
4.3.3	Adapting the Damping and Natural Frequency of each neuron	65
4.3.4	Adapting the slope of the sigmoidal nonlinearity	66

<i>CONTENTS</i>	vi
4.4 Practical Considerations and Implementation in MATLAB	68
5 Conclusions and Future Directions	70
5.1 Resonator Neural Networks and Engineering Design	71
5.2 The RAF Neuron and Neurobiology	72
A Derivation of the Resonate-and-Fire Neuron	75
A.1 The Hodgkin-Huxley model	76
A.2 The FitzHugh-Nagumo Model	78
B Derivation of The Static Backpropagation Algorithm	82
B.1 The computation of the local gradient for hidden neurons	85
B.2 Summary of the static backpropagation algorithm	87
B.3 Example: A neural network implementation of a square-root finding algorithm	87
C Introduction to Phase Space Analysis	89
D Computer Simulations Program Code	95

List of Figures

1.1	Neuron, corresponding model, and neural networks	3
1.2	The realm of neuronal modeling	4
1.3	Three generations of neural networks	7
1.4	The integrate-and-fire neuron	10
1.5	Phase space representation of the Integrate-and-Fire Neuron	12
1.6	The dynamics of the Integrate-and-Fire Neuron in response to depolarizing stimuli	13
1.7	The Hodgkin-Huxley model	16
1.8	The derivation of tuning curves	21
2.1	The resonate-and-fire neuron	25
2.2	The resonate-and-fire phase portrait and nullclines	29
2.3	Time and phase space representation of resonate-and-fire dynamics	30
2.4	Response of the resonate-and-fire neuron to a pair of input stimuli	32
2.5	Postinhibitory facilitation in the resonate-and-fire model	34

2.6	The integrate-and-fire neuron in response to an inhibitory stimulus	36
2.7	Excitatory input train to resonate-and-fire neuron	37
2.8	Excitatory/Inhibitory inputs to resonate-and-fire neuron	39
2.9	Input configurations and parameter choices in the resonate-and-fire model	40
3.1	The "OR" and "AND" Problems	43
3.2	The resonate-and-fire model for the solution of the "OR" and "AND" problems	45
3.3	The "OR" Problem solution with the resonate-and-fire neuron	46
3.4	Solution to the "AND" Problem with the resonate-and-fire neuron	47
3.5	The XOR Problem classification	49
3.6	Solution to The XOR Problem with the Resonate-and-Fire Neuron - Single Stimulus	50
3.7	The Resonate-and-Fire Neuron's dependence on the timing of input stimuli	52
4.1	Neuron model for temporal backpropagation algorithm	55
5.1	Neuronal Pathways, Network Motifs, and Information Representation	74
A.1	The FitzHugh-Nagumo model: Nullclines and Phase Portrait	79
A.2	Integrators and Resonators in the FitzHugh-Nagumo model	81
C.1	Phase line for one-dimensional system	90

LIST OF FIGURES

C.2 Time responses of one dimensional system 91

C.3 Phase portrait and state trajectory for two dimensional system 92

Abstract

The means through which the nervous system perceives its environment is one of the most fascinating questions in contemporary science. Our endeavors to comprehend the principles of neural science provide an instance of how biological processes may inspire novel methods in mathematical modeling and engineering. The application of mathematical models towards understanding neural signals and systems represents a vibrant field of research that has spanned over half a century. During this period, multiple approaches to neuronal modeling have been adopted, and each approach is adept at elucidating a specific aspect of nervous system function. Thus while biophysical models have strived to comprehend the dynamics of actual physical processes occurring within a nerve cell, the phenomenological approach has conceived models that relate the ionic properties of nerve cells to transitions in neural activity. Furthermore, the field of neural networks has endeavored to explore how distributed parallel processing systems may become capable of storing memory.

Through this project, we strive to explore how some of the insights gained from biophysical neuronal modeling may be incorporated within the field of neural networks. We specifically study the capabilities of a simple neural model, the Resonate-and-Fire (RAF) neuron, whose derivation is inspired by biophysical neural modeling. While reflecting further biological plausibility, the RAF neuron is also analytically tractable, and thus may be implemented within neural networks. In the following thesis, we provide a brief overview of the different approaches that have been adopted towards comprehending the properties of nerve cells, along with the framework under which our specific neuron model relates to the field of neuronal modeling. Subsequently, we explore some of the time-dependent neurocomputational capabilities of the RAF neuron, and we utilize the model to classify logic gates, and solve the classic XOR problem. Finally we explore how the resonate-and-fire neuron may be implemented within neural networks, and how such a network could be adapted through the temporal backpropagation algorithm.

Chapter 1

Introduction

The inherent challenge of all mathematical modeling is to create an optimal balance between realism, complexity, and the level of insight that the model provides. The objective in conceiving theoretical biological models is to capture the essential aspects of a living system, and thus to enhance our understanding of the mechanisms through which the natural system functions. However, the inherent complexity of natural phenomena entails that the state of a system at any given time is influenced by multiple variables. Thus an extremely realistic model may also be tremendously complex and intractable, and it is often difficult to comprehend the fundamental properties of a system through an overly complicated model.

Through this chapter we explore the challenges that arise in conceiving neuronal models, and the vastly different approaches to neural modeling adopted by neural networks and biophysical neural models. The intersection of biophysical neural modeling with the field of neural networks provides the foundation for this thesis, and thus through this chapter we review each approach to neural modeling in turn. Our objective in this chapter is to provide the historical framework of neuronal modeling that has motivated this project, and to locate our particular neuron model within the vast spectrum of neural modeling. We conclude this chapter with a discussion of the phenomena that are currently lacking with regards to neural networks, and thus form the motivation for this thesis.

1.1 Neurophysiological Considerations

Let us begin by considering the morphology of a typical nerve cell, along with the general approach utilized to model the various electrophysiological aspects of neurons. Panel (A) in figure 1.1 depicts the various specialized cellular processes that characterizes nerve cells, and panel (B) represents a general model neuron. The cell body corresponds to the metabolic center of a neuron, and it gives rise to two types of processes - the dendrites and the axon. The dendrites, commonly referred to as the dendritic tree, serve as the main apparatus through which a neuron receives input signals arising from other nerve cells, or pre-synaptic neurons. The signals that arrive at the dendritic tree usually constitute chemical neurotransmitters. At the input terminals of the dendritic tree are individual synapses, which contain various receptors that selectively bind a specific neurotransmitter secreted by the presynaptic neuron. Upon binding neurotransmitters, the receptor protein initiates a unidirectional current flow in the postsynaptic neuron [1]. Depicted in panel (B) of figure 1.1 is a neuron model, where the input signals are represented by an input vector, \vec{x} . The signal received on each input is subsequently multiplied by synaptic weights, that correspond to the strength of a specific synapse. The synaptic weights in the model neuron conceptually model two different aspects of neuronal signaling, namely, the efficacy of synaptic transmission, along with the amount of attenuation that an input signal undergoes as it reaches the cell body.

Electrical signaling in nerve cells is mediated through two generic mechanisms: i) the passive spread of attenuated signals through the dendritic tree, and ii) the propagation of all-or-none *action potentials*, or nerve impulses, along the axon of the neuron [2]. While the dendritic tree receives input signals from other neurons, the axon represents the neuronal process that transmits output signals. The axon is the main conducting unit of a neuron and it conveys information to other neurons (post-synaptic neurons) by propagating action potentials. Unlike the attenuated signals that traverse the dendritic tree, the action potential is not attenuated as it traverses the axon.

Underlying the mechanism of action potential generation is the potential difference across the nerve cell membrane, and the various voltage-sensitive ion channels that traverse the neuronal membrane. A very basic explanation for the phenomenon of action potential generation is that any time the potential difference across the neural membrane exceeds some threshold value, thousands of sodium channels are activated, which results in the rapid influx of sodium ions into the nerve cell. This rapid influx of positive charge creates a large increase in the membrane potential (depolarization),

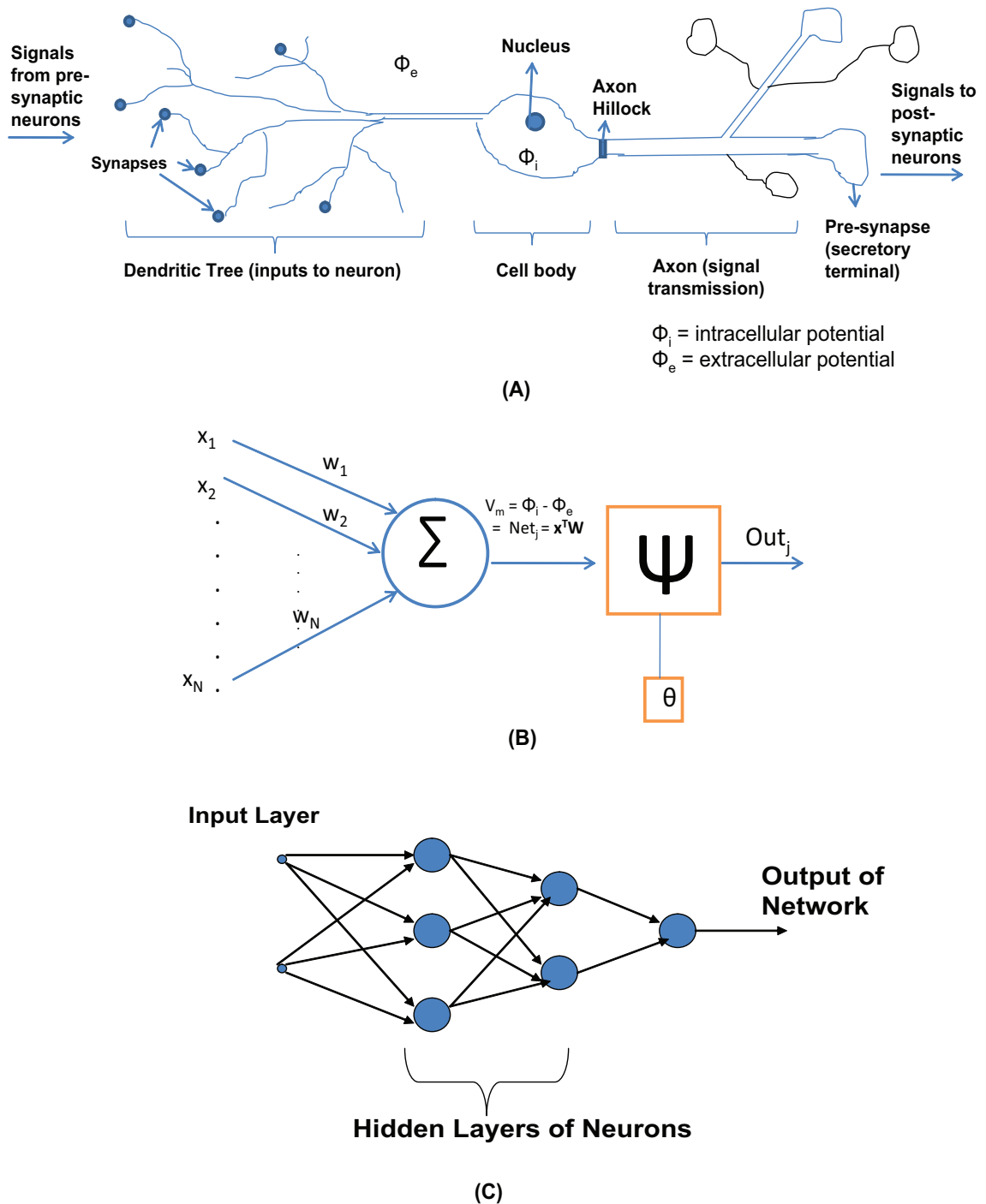


Figure 1.1: The neural morphology and corresponding neuron model in neural networks. (A) A neuron typically has four major processes: i) a dendritic tree that receives input signals from pre-synaptic neurons, ii) the cell body, iii) the axon, which transmits signals to post-synaptic neurons; the axon hillock is the region where action potentials are usually generated, and iv) the pre-synaptic junction which secretes chemical transmitters, and thus enables the neuron to communicate with post-synaptic neurons. (B) A model neuron, where weighted input signals are summed, and the result is passed onto an activation function, Ψ . The activation function mimics the decision to generate an action potential, which is usually made at the axon hillock in (A). If the output of the activation function is greater than the threshold, θ , the output is transmitted to all post-synaptic neurons. (C) A multilayer perceptron neural network where each of the neurons depicted in the hidden layer represent a neuron model as in panel (B). The connections depicted between each neuron represents the synaptic weights. See the text for further explanation.

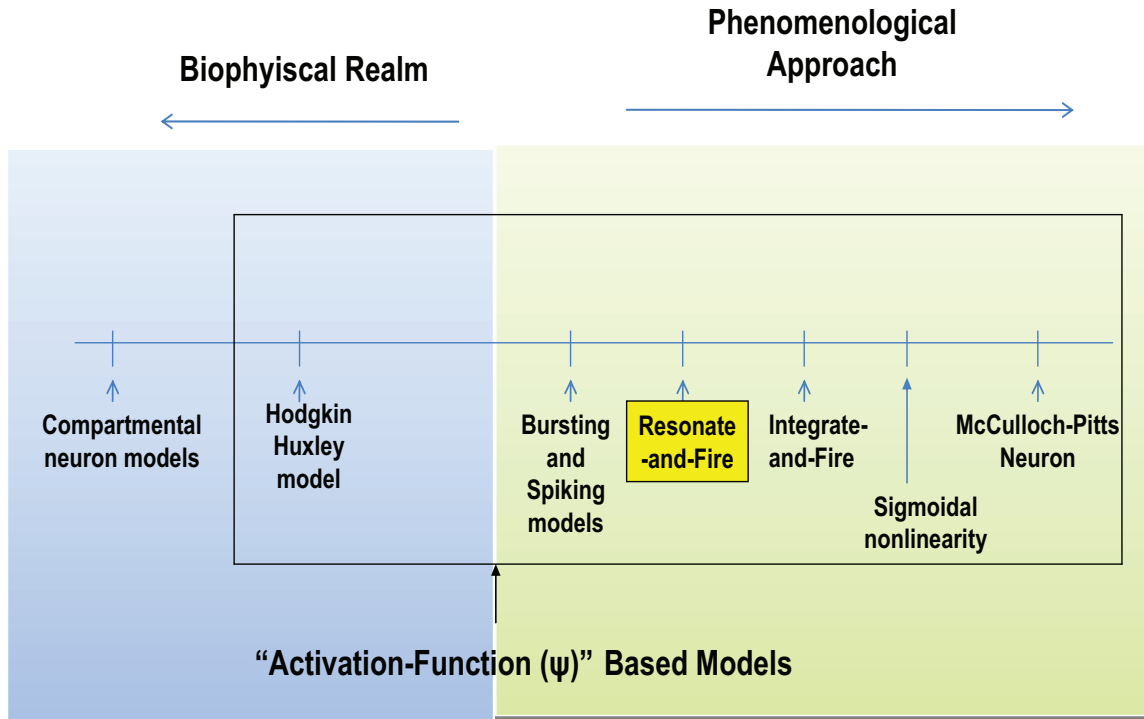


Figure 1.2: The vast spectrum of approaches towards neuronal modeling. Neural models range from the elaborate compartmental models to the simple perceptron-based modeling of neural networks. A significant idea in neural networks regards the form of the activation function, and thus depicted in the figure are models that utilize some form of an activation function (enclosed in the rectangle).

and this depolarization in turn triggers the generation of the action potential at the axon hillock. The depolarization also activates potassium channels, and this leads to potassium ions flowing out of the cell. Since in this instance, positive charge leaves the nerve cell, the membrane potential decreases (repolarization), and thus the membrane potential returns to its resting state.

There exist various approaches to neuronal modeling, and the various details of neurophysiology that a model incorporates depend on the types of insights that we wish to gain through the model. Depicted in figure 1.2 is a brief summary of the multiple levels of neuronal modeling that have been pursued over the last few decades - from the elaborate contemporary compartmental biophysical models, to the simple threshold gates that characterized the early history of neural networks. We refer the

reader to [3] for an overview of theoretical neuroscience, and to [4] for an introduction to quantitative neurophysiology. Furthermore, a review of common models in neurobiology may be accessed in [5], while an introduction to the phase space approach to neuronal modeling and analysis is found in [6]. Moreover, reference texts on the theory of neural networks may be accessed in [7] and [8]. Finally, a collection of representative publications which exemplify the multiple approaches to neuronal modeling is found in [9]. In the remainder of this chapter we strive to explore where our specific neuron model - the Resonate-and-Fire (RAF) Neuron, is situated within the greater context of neuronal modeling.

1.2 Neural Networks - An Overview

The study of neural networks was originally inspired by the structure and function of biological nervous systems. A biological neuron receives thousands of signals from pre-synaptic neurons, and in turn transmits signals to thousands of other neurons. The sheer complexity underlying higher level phenomena in nervous systems, such as learning, necessitates simpler models of neurons, and one such generic model is depicted in panel (B) of figure 1.1.

In panel (B) of figure 1.1, a signal vector x_i at the input of a synapse is connected to neuron j through a synaptic weight w_{ij} . The signal vector may constitute input data in the form of external stimuli applied to the network, or it may reflect the signals that neuron j receives from other pre-synaptic neurons. The products of the weight and signal are passed through a summation, i.e. a *spatial integrator*, which models the cell body of the neuron. The resulting sum constitutes the net input to neuron j , labelled as Net_j in panel (B) of figure 1.1. The neuron model depicted in figure 1.1 is an extremely simple representation of the myriad factors associated with a single neuron. For instance, the pre-synaptic signal received by neuron j (i.e. the output of the spatial integrator) is in reality a current, which mimics the flow of neurotransmitters across a biological synaptic junction. The synaptic weights depicted in figure 1.1 represent several physiological characteristics of neurons. The first is that within nervous systems, neurons are coupled together through synaptic junctions. Thus the significance of the synaptic weights is to approximate the coupling strength between neurons, and thus reflect the strength of the connection between neurons. Furthermore, a pre-synaptic signal is usually attenuated as it traverses a neuron's dendritic tree before it reaches the soma of a neuron. Therefore the synaptic weights also reflect the attenuated effect that pre-synaptic signals have on

the membrane voltage of the post-synaptic neuron.

As shown in the neuron model of figure 1.1, the net input to neuron j is subsequently passed through an activation function, Ψ , that maps the net input to a corresponding activation pattern. The threshold, θ , has the effect of altering the net input of the activation function. The model depicted in figure 1.1 can be summarized by the following two equations:

$$Net_j = \sum_{i=1}^N w_{ij} * x_i \quad (1.1)$$

$$Out_j = \Psi(Net_j) - \theta \quad (1.2)$$

where Net_j refers to the sum of weighted inputs that are presented to perceptron j , and it corresponds to the net internal activity level of the neuron. The output of the neuron, Out_j , is obtained by mapping the internal activity level through an activation function, Ψ . The specific form of the activation function utilized, is where the various approaches to neural modeling differ from each other. As is discussed in the subsequent section, the biophysical approach models the activation function through nonlinear dynamic differential equations. On the other hand, neural networks have commonly attributed a static form to the activation function. A summary of the various neural models utilized in neural networks is presented in figure 1.3.

1.2.1 Neuron models with static activation functions

The field of neural networks has been largely inspired by how we may create a theoretical model of nervous system structure and function. Through more than half a century of research in connectionist neural networks, we may broadly categorize the activation functions that neural network models utilize as belonging to one of three general categories, namely, i) threshold gates, ii) sigmoidal nonlinearity, and iii) time-dependent models. These categories reflect generations and paradigms of neural network research, and thus in this section, we briefly review the forms of the activation functions that epitomize each of these generations.

The first formal models of a neuron utilized activation functions (Ψ in panel (B) of figure 1.1) that followed the generic idea underlying the McCulloch-Pitts model [10], which treats a neuron as a threshold gate. Under such a scenario, the activation function takes the form of a threshold function where:

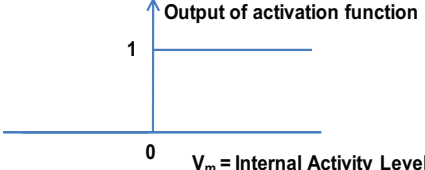
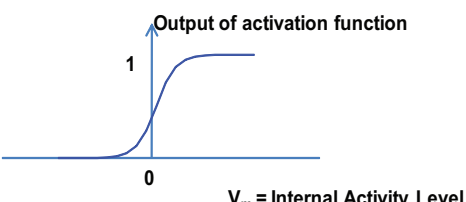
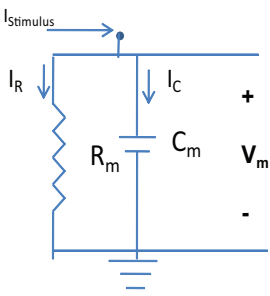
Generation I: McCulloch-Pitts Threshold Neurons	Activation Functions, Ψ
	$\text{Out}(v) = \begin{cases} 1 & \text{if } V_m \geq 0 \text{ (or threshold)} \\ 0 & \text{else} \end{cases}$
<p>Generation II: Sigmoidal non-linearity</p> 	$\text{Out}(v) = 1/(1 + \exp(-a \cdot v))$
<p>Generation III: Integrate-and-fire</p> 	$\frac{dV_m}{dt} = -V_m/(R_m \cdot C_m) + I_{\text{Stimulus}}$ <p>if $V_m \geq \text{Threshold}$, Neuron emits a spike</p>

Figure 1.3: Three generations of neural networks. The first generation of neural networks comprised McCulloch-Pitts (MP) threshold gates, where the output of the neuron is ONE if the membrane potential exceeds zero, and the output is ZERO otherwise. Due to the discontinuous nature of the MP neurons, the second generation of models utilized sigmoidal functions as activation functions. Sigmoidal functions are universally differentiable, and neural networks comprised of sigmoidal neurons may be trained with error backpropagation. Finally, the integrate-and-fire model defines the membrane potential through a first order linear differential equation.

$$\begin{aligned} Out_j &= 1 && \text{if } Net_j \geq 0 \\ Out_j &= 0 && \text{otherwise} \end{aligned} \tag{1.3}$$

The classic perceptron was conceived based on the threshold activation function defined above. The perceptron consists of a single neuron with adjustable synaptic weights and threshold, and can be used for the classification of linearly separable patterns. The algorithm used to adjust the free parameters of the perceptron first appeared in a learning procedure developed in [11] and [12]. The method of steepest descent and the least-mean-square algorithm [13] further formalized the procedure used to train a perceptron.

The next generation of neural network models formulated the activation function as a smooth continuous sigmoid function. A specific example of a sigmoid function is the logistic function which has the general form

$$Out_j = \frac{1}{1 + \exp(-a * Net_j)} \tag{1.4}$$

where Net_j represents the internal activity level of a neuron. In contrast to the threshold function defined in the McCulloch-Pitts model, sigmoid functions are smooth and continuous, and thus differentiable. The smoothness property of the sigmoid function ushered in a new era of neural networks with the development of multilayer perceptrons and the back-propagation algorithm [14] and [15], which shall be discussed further in section 1.2.3. We now turn to the third generation of neural network modeling and the integrate-and-fire neuron.

1.2.2 The Integrate-and-Fire Neuron

The history of neural networks reveals that whenever neurons are modeled to reflect greater biological relevance, the neural network itself becomes capable of performing increasingly complicated tasks. The integrate-and-fire model was experimentally conceived by Louis Lapicque in 1907 [16]. Through stimulating the sciatic nerve of frog muscle, Lapicque found that the soma of a neuron acts much like a temporal integrator. Over a century after Lapicque's experiments, the integrate-and-fire model has become a commonly utilized neural model within neural networks. It has been shown

that neural networks comprised of spiking neurons are more computationally effective than their non-spiking predecessors [17]. The generic integrate-and-fire neuron is depicted in figure 1.4.

The integrate-and-fire model represents the third generation of neural network modeling, in that it strives to model the neural membrane potential through a simple one-dimensional differential equation. The integrate-and-fire neuron is comprised of a parallel resistor-capacitor circuit; thus the application of an input current charges the circuit, and when the membrane potential reaches some threshold value, the neuron emits a spike. The amount of time it takes for the circuit to reach the threshold voltage, is utilized to encode information.

In figure 1.4, a current $I_{stim}(t)$ is defined as the sum of the products obtained by multiplying the input signal vector with the synaptic weights. Thus I_{stim} in this instance is equivalent to Net_j in panel (B) of figure 1.1. The current $I_{stim}(t)$ charges the parallel resistance-capacitance circuit. The voltage across the capacitor, $v(t)$ - which reflects the membrane voltage of the neuron, is compared to a threshold φ . If the membrane voltage exceeds the threshold at some time, $t_i(f)$, an output pulse $\delta(t - t_i(f))$ is generated. Subsequently, the voltage across the circuit is reset to some value, V_{reset} , that is below the threshold. Thus integrate-and-fire neurons are also known as spiking perceptrons. The leaky integrate-and-fire model can be mathematically summarized by the following:

$$\begin{aligned}
 I_{stim}(t) &= I_R + I_C \\
 \Rightarrow I_{stim}(t) &= \frac{v(t)}{R} + C * \frac{dv}{dt} \\
 \frac{dv}{dt} &= -\frac{v(t)}{R * C} + \frac{I_{stim}(t)}{C} \\
 \Rightarrow v &\rightarrow V_{peak} \quad \text{if } v > \varphi \\
 \Rightarrow v(t+1) &< - - V_{reset}
 \end{aligned} \tag{1.5}$$

where C refers to the membrane capacitance, R refers to the resistance, and φ refers to the threshold value of the neuron.

Let us first consider the natural response of the integrate-and-fire neuron, i.e. with $I_{stim}(t) = 0$. Thus we have:

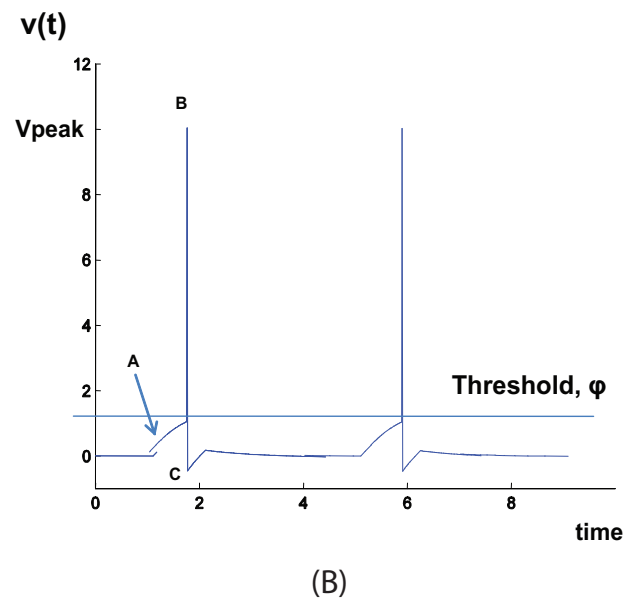
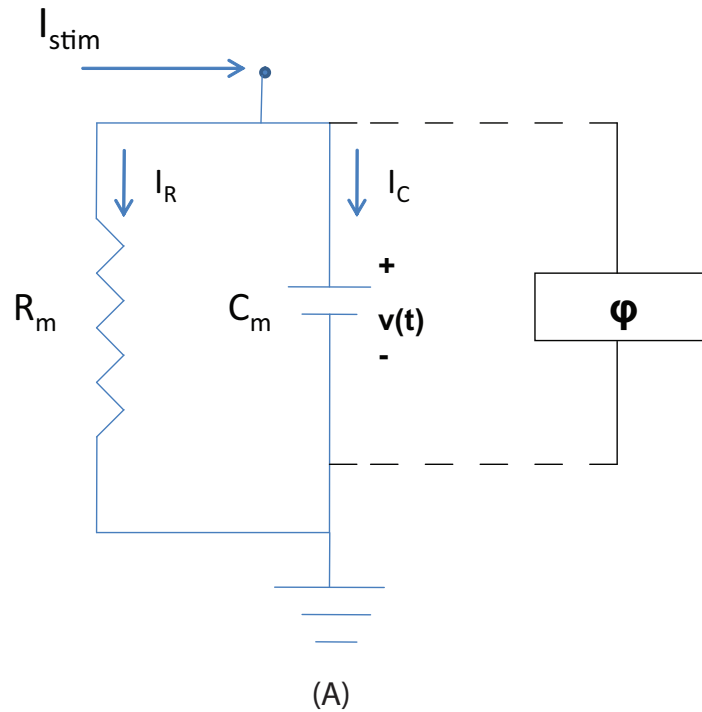


Figure 1.4: The integrate-and-fire neuron. (A) Equivalent circuit model of integrate-and-fire (IAF) neuron. The stimulus current I_{stim} is comprised of the products of the synaptic weights and signal vector \vec{x} , as depicted in figure 1.1. Since the IAF model is a parallel combination of a resistance and capacitance, the neuron is defined by a first order linear differential equation. The stimulus current charges the circuit, and any time the membrane potential $v(t)$ exceeds the threshold ψ , the neuron emits a spike. (B) Representative evolution of IAF membrane potential in time in response to excitatory stimuli. 'A' represents the charging phase, 'B' depicts the spike emission, and 'C' refers to the reset phase of the IAF model.

$$\dot{v} = \frac{dv}{dt} = -\frac{v(t)}{R * C} \quad (1.6)$$

The natural response of the integrate-and-fire neuron is thus:

$$v(t) = V_o * \exp\left(\frac{-t}{R * C}\right) \quad (1.7)$$

where V_o refers to the initial voltage at time = 0.

A complementary approach to analyze the integrate-and-fire model is through the phase space. An introduction to phase space analysis is attached in appendix C and depicted in figure 1.5 is the phase-space representation of the integrate-and-fire neuron.

Since the integrate-and-fire neuron is defined by a one-dimensional differential equation, the evolution of the neuron's membrane potential may be viewed as a flow on a line. The steady state value of the membrane potential corresponds to the instance when $\dot{v} = 0$, or where the v-nullcline in figure 1.5 intersects the x-axis. A positive stimulus current has the effect of shifting the v-nullcline upwards, thus shifting the steady-state potential to the right, and hence increasing the membrane potential of the neuron. Once the potential crosses the threshold value, the neuron emits a spike, thus causing the voltage to reach V_{peak} , and the potential is immediately reset to V_{reset} . Figure 1.6 displays the evolution of the membrane potential in the time domain and in the phase space, in response to a series of depolarizing stimulus current pulses.

In response to negative stimulus currents, the steady-state membrane potential moves towards the left in the phase line, thus implying a decreasing membrane potential. Under such a scenario, the membrane potential never exceeds threshold, and thus the neuron never transmits action potentials in response to inhibitory synaptic currents. However, we know that some biological neurons are capable of generating action potentials even in response to inhibitory currents - a phenomenon known as anode break excitation. Examples of such neurons include thalamocortical relay neurons and mesencephalic V neurons in the rat brain stem [6]. Thus part of our goal through this project is to derive a neuron model that is capable of being responsive even to inhibitory synaptic currents.

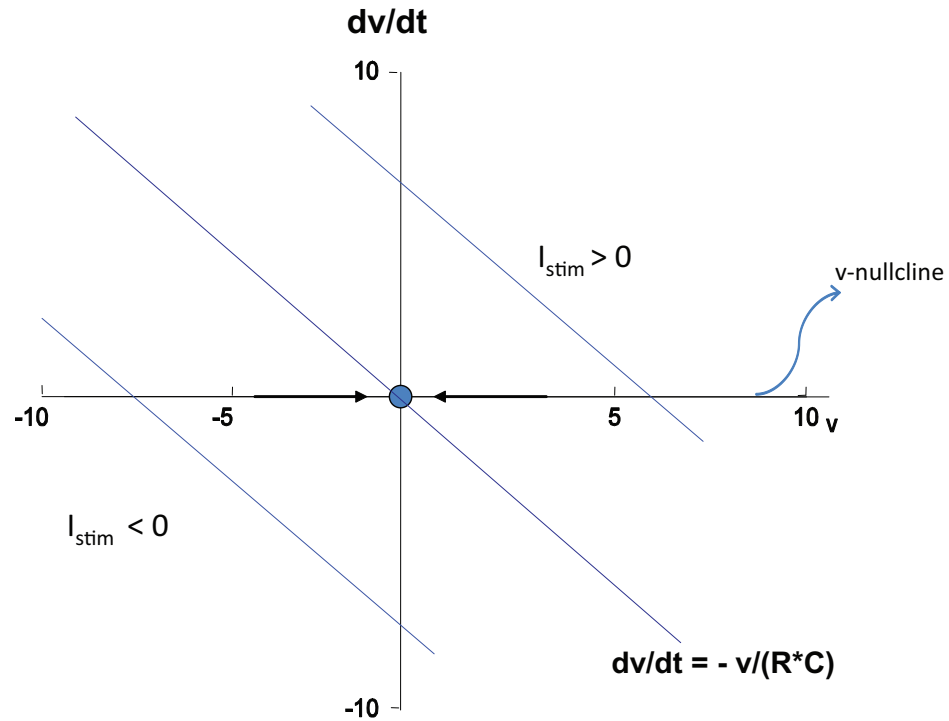


Figure 1.5: A phase space representation of the integrate-and-fire neuron's dynamics. A depolarizing stimulus shifts the steady-state of the system, and the state of the neuron decays back to the original resting potential. Since the integrate-and-fire neuron is defined by a one-dimensional differential equation, the dynamics of the membrane potential is constrained to be a flow on a line.

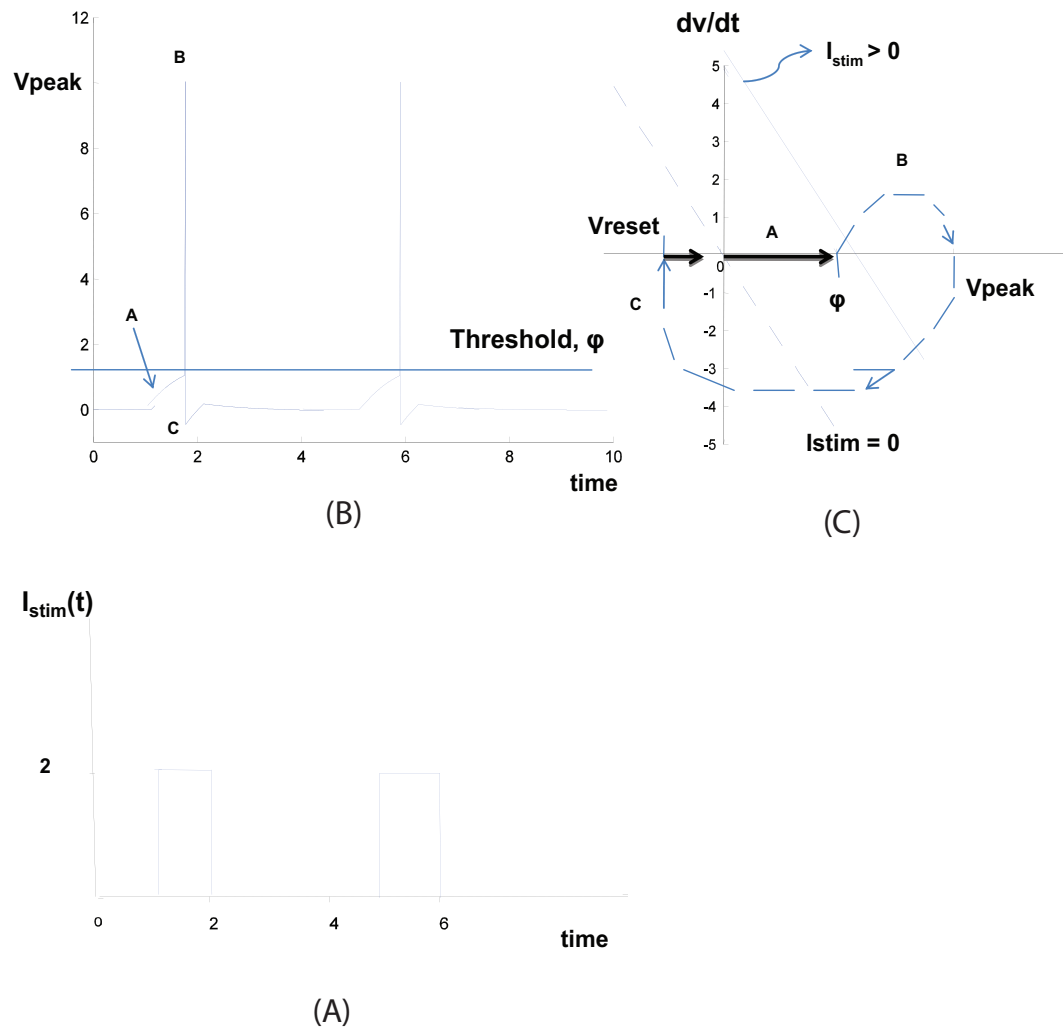


Figure 1.6: (A) Depolarizing current stimuli is applied to the integrate-and-fire neuron, which evokes a super-threshold response. (B) Representation of membrane potential in time. (C) Phase space representation of the integrate-and-fire neuron's dynamics in response to depolarizing stimuli. Since the integrate-and-fire neuron is defined by a one-dimensional differential equation, the dynamics of the membrane potential is constrained to be a flow on a line. Thus the reset phase in panel (C) does not exist in the dynamics of the integrate-and-fire model. Rather, the reset condition is imposed and "hard-coded" into the model

1.2.3 The Learning Procedure

Perhaps one of the most powerful aspects of neural networks is the learning procedure, which is used to train a network to perform some desired task. In panel (C) of figure 1.1, the strength of connections between neurons are referred to as the synaptic weights. Thus the learning procedure entails adjusting the strength of the synaptic weights to yield a certain network response.

Given a set of inputs, we may define a desired response that the network should yield when presented with these inputs. The difference between the desired response and the actual response of the network constitutes an error that we wish to minimize. The error may be defined as follows:

$$e_j = d_j - o_j \quad (1.8)$$

where d_j reflects the desired response that an output neuron j should yield, o_j is the actual response of the output neuron, and e_j is the error. Subsequently, we may derive an energy function which reflects the amount of error that the network yields. A common form attributed to the energy function is:

$$E = \sum_{j \in O} e_j^2 \quad (1.9)$$

where E is the energy, and the set O reflects the total number of output perceptrons in the network. The goal of the learning procedure is to minimize the error surface with respect to the free parameters of the system, i.e. the weights and thresholds applied to each neuron.

Various learning procedures have been developed throughout the history of neural networks. Each procedure has been devised according to the specific neuron model that is utilized to design a neural network. For instance, the classic McCulloch-Pitts perceptron is often trained through the method of steepest descent and the least mean square algorithm. The error backpropagation algorithm is usually employed to train neural networks comprised of sigmoidal-type neurons.

The back-propagation process consists of two different phases. In the first phase, known as the forward pass, an input vector is applied to the network, and its effect is transmitted through the network, layer by layer. Finally, an output is produced as

the actual response of the network. The actual response is subtracted from a desired response, and this result constitutes the error signal. In the second phase, also known as the backward pass, the error signal is propagated through the network, against the direction of synaptic connections. The synaptic weights are adjusted so as to make the actual response of the network approach the desired response [?].

Error backpropagation has emerged as a powerful tool in neural networks. However, the backpropagation algorithm was originally developed for static input patterns, and the derivation of the static backpropagation algorithm is attached in appendix B. Part of our goal in this project is to consider how a temporal backpropagation algorithm could be utilized to adapt a neural network comprised of time dependent neurons. Moreover, traditionally the backpropagation algorithm is implemented to adapt the synaptic structure of the network, i.e. the strength of the synaptic weights. However, we know that the neurophysiological learning process entails enhancements in synaptic transmission, as well as morphological changes in individual nerve cells [1], [18], [19], [20]. Thus through this project we also explore how the internal parameters that characterize individual neurons within a neural network may be adapted alongside the synaptic structure of the network.

The utilization of error backpropagation requires that the activation function be differentiable, and analytically tractable. A further goal of this project is to enhance neural networks to reflect greater biological plausibility, while also retaining the analytical tractability that is required to utilize the backpropagation algorithm. Thus we subsequently explore how methods in biophysical neural modeling could provide the framework for neuron models in neural networks.

1.3 Biophysical Neural Modeling

So far in our discussion we have focused on the neural network approach to neuronal modeling, and thus we have concentrated on the simplest neural models that are depicted in figure 1.2. Through this section we turn our attention towards the opposite extreme of figure 1.2, and concern ourselves with the biophysical realm of neuronal modeling.

The essence of biophysical neural models pertains to the characterization of actual physical processes that occur within a nerve cell. Thus the most elaborate neuronal models are those that segment a nerve cell into discrete compartments, and strive to

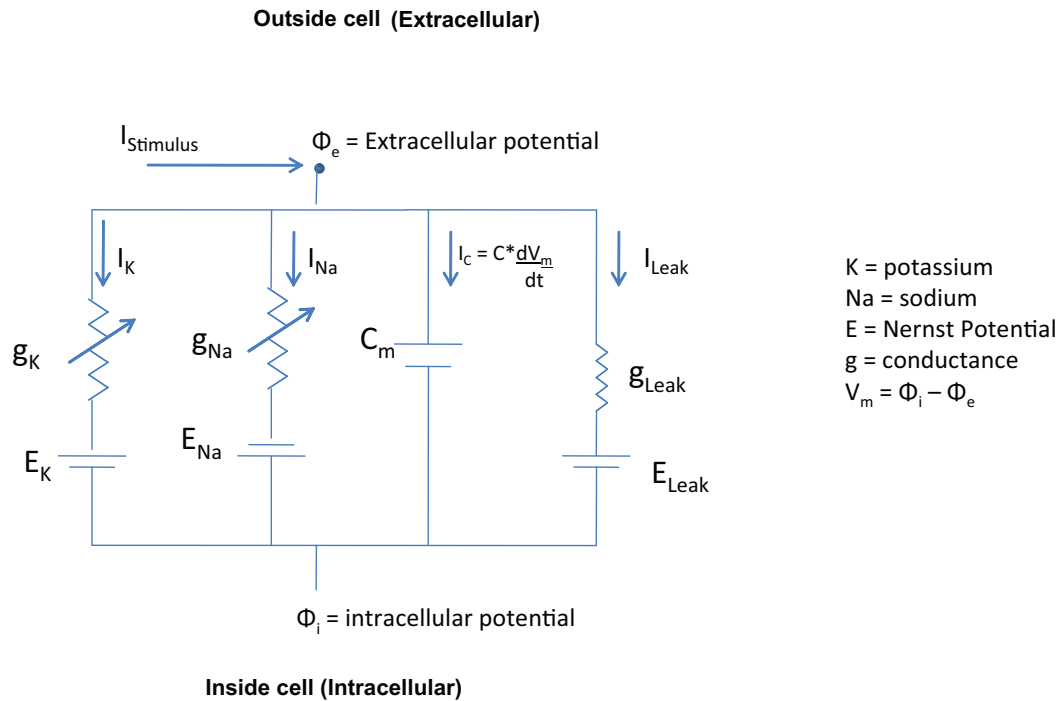


Figure 1.7: The equivalent circuit representation of the Hodgkin-Huxley parallel conductance model. In the squid giant axon, a patch of neural membrane consists of three major types of voltage gated ion channels, which selectively mediate the flow of three types of ionic currents: a sodium current, a potassium current, and a leak Ohmic current. The Hodgkin-Huxley model is a four-dimensional system of coupled nonlinear dynamic differential equations which relate the dynamics of the neural membrane potential to the dynamics of ion channel gating parameters. See text for equations.

model the nature of the currents that flow through each compartment. The procedure utilized to derive compartmental models can be traced back to the space clamped approach to quantitative neural modeling that was originally pioneered by Alan Hodgkin and Andrew Huxley in the early 1950s. The Hodgkin-Huxley (HH) model [21] represents one of the most important and insightful models in theoretical neuroscience, and it is also a relatively simple example of a biophysical model. Through performing experimental studies on the squid giant axon, Hodgkin and Huxley proposed the parallel conductance model depicted in figure 1.7. Hodgkin and Huxley's quantitative description of the phenomenon of action potential generation was awarded the Nobel Prize in physiology/medicine in 1963.

The core idea of the HH parallel conductance model is that within a small patch of neural membrane, there exist three major currents that traverse the squid axon: i) a voltage-gated transient sodium current with three activation gates and one inactivation gate, ii) a voltage-gated persistent potassium current with four activation gates, and iii) an Ohmic leak current. Through performing numerical and experimental analyses, Hodgkin and Huxley characterized the dynamics of the sodium and potassium ion channels that result in the generation of the action potential. Thus the HH model strives to describe the dynamics of the activation function depicted in figure 1.1, using the biophysical details that characterize the neural membrane. The qualitative form of the "activation function" in the HH model is defined by the relative time constants of sodium and potassium ion channels. Sodium channels operate at a much faster time scale than potassium channels, and thus a slight depolarization of the membrane potential initially activates thousands of sodium channels. The influx of sodium ions into the nerve cell creates the upstroke of the action potential. Potassium channels are also voltage-gated, however, they operate at a much slower time scale than do sodium channels. Thus the repolarization of the neural membrane is achieved through the efflux of potassium ions out of the nerve cell, along with the inactivation of the sodium channels. The four dimensional system of coupled differential equations that defines the Hodgkin-Huxley model are:

$$C * \frac{dV_m}{dt} = -g_{Na}m^3h(V - V_{Na}) - g_Kn^4(V - V_K) - g_L(V - V_L) + I_{stimulus} \quad (1.10)$$

$$\tau_n(V_m) * \frac{dn}{dt} = n_\infty(V_m) - n \quad (1.11)$$

$$\tau_m(V_m) * \frac{dm}{dt} = m_\infty(V_m) - m \quad (1.12)$$

$$\tau_h(V) * \frac{dh}{dt} = h_\infty(V_m) - h \quad (1.13)$$

where V_m refers to the potential difference across the neural membrane, and m , h , and n refer to gating parameters. The formulation shown in equations 1.10 to 1.13 differs from the original Hodgkin-Huxley equations, however, the convention depicted above has been adopted in contemporary theoretical neuroscience.

The m and h gates define the dynamics of sodium channels, while the n gate corresponds to potassium channel dynamics. Furthermore, the term $n_\infty(V_m)$ refers to the

steady-state value of the n gate at a particular membrane potential, while $m_\infty(V_m)$ and $h_\infty(V_m)$ correspond to the steady-state levels of sodium channel gating parameters. Moreover, τ refers to the rate constant of each of the gating parameters, C_m corresponds to the membrane capacitance, and g refers to the nonlinear conductance of each of the ion channels as a function of the membrane potential. It is important to note that the rate constants τ , the ion channel conductances, and the steady-state ion channel gating parameters are all nonlinear functions of the membrane potential, V_m .

The four dimensional Hodgkin-Huxley model is analytically intractable, however, we may gain insights into the generic dynamics of the system by considering variables that operate at similar time scales. Thus the membrane potential V_m along with the sodium activation m gate operate at much faster time scales than the n and h variables. Therefore we may reduce the four variable HH model into a fast-slow system comprised of two differential equations, whereby the resulting reduced system qualitatively reflects many of the characteristics of the original HH model [22]. Subsequently, we may analyze the resulting system of differential equations on a two dimensional phase plane. An instance of such a simplification is found in the FitzHugh-Nagumo (FHN) model [23], [24], and we derive the resonate-and-fire neuron from the HH and FHN models in appendix A. The notion of reducing a biophysical neural model, such as the HH model, into a reduced system of equations underlies the phase space approach to nonlinear dynamic neuronal modeling. In figure 1.2 we refer to such models as *Bursting and Spiking Models*. Neuronal models that are derived through the phase space are conceived to exhibit the various different regimes of neural activity, such as quiescence, spiking, and bursting, and these models relate the transitions in neural activity to specific bifurcations that a neural system is capable of undergoing.

The ideas that underlie the HH parallel conductance model have been extended over the years to yield compartmental biophysical models. While the HH parallel conductance model strives to model a small patch of neural membrane, a compartmental model segments a nerve cell into multiple compartments, each with an equivalent circuit representation. Consequently, a compartmental model is typically comprised of several coupled circuits, and one of these circuits resemble the one depicted in figure 1.7. Thus a general method utilized to derive a compartmental model begins with an estimation of the distribution of different ion channels across different segments of the dendritic tree; each ion channel is subsequently represented by an equivalent conductance based model. A similar procedure is performed for the cell body and the axon of a neuron. Subsequently, each compartment is coupled to adjacent compartments by an equivalent conductance term. Due to the intricate nature of neuronal processes, compartmental models are usually comprised of dozens, and even hundreds,

of coupled differential equations. Hence the fundamental objective of a compartmental model is to characterize the flow of ionic currents through the neural membrane, across different modes of neural activity. Instances of compartmental neuron models are found in [25] and [26].

Biophysical neural models prove to be insightful in studying the microscopic properties of nerve cell processes, such as the dynamics of ion channel gating, or the role of genetic mutations in contributing to aberrant neural activity. Consequently biophysical models are of interest to investigators who seek to understand how we may alter neural activity through the administering of drugs, or who strive to model the relation between the molecular and electrophysiological properties of a nerve cell. However, biophysical models are not well suited towards understanding the macroscopic aspects of nervous system function, such as the means through which the nervous system learns, or stores memory.

The sheer complexity of biophysical neural models entail that they are not analytically tractable, and thus cannot be trained using the learning procedures utilized in neural networks, such as error backpropagation. The implementation of the backpropagation algorithm requires that there exist a specific input-output activation function that defines each neuron. Therefore, through our project we are striving to study a neural model that incorporates some of the insights gained from biophysical neural modeling, while also being analytically tractable, and thus implementable in a neural network.

1.4 Neuronal selectivity and the derivation of tuning curves

One of the most classic neurophysiological experiments pertains to the derivation of *tuning curves* for individual nerve cells. In general, a tuning curve relates the activity level of a nerve cell to an "input parameter". The activity level is usually quantified as the average number of action potentials that an individual neuron generates per second, and is therefore the average frequency of action potential generation. Thus as the input parameter is varied, the activity levels of individual neurons are measured. The input parameter may assume a number of different forms, such as the frequency of sound or the orientation of visual stimuli. In an experiment where we wish to derive a tuning curve, a neural pathway is exposed to the particular input stimuli, and the

activity levels of different nerve cells are measured. Thus when the input stimuli is visual stimuli, the response of neurons in the visual cortex are usually monitored, and when the organism is exposed to various frequencies of sound, the response of cells in the auditory cortex are quantified. The general idea of a tuning curve is depicted in panel (A) of figure 1.8.

The method of deriving tuning curves dates back to the late 1950s, when David Hubel and Torsten Wiesel performed their seminal studies regarding information processing in the visual system [27], [28], [29]. Hubel and Wiesel exposed anesthetized cats to various orientations of a straight line, and they monitored the activity level of individual neurons in response to different angular orientations. They found that some cells were selectively responsive to particular orientations of the line, while these same cells became silent for other orientations. Similarly, they also observed that some neurons were most responsive when the line was displaced, while others were most responsive for static lines. These studies suggested that specific neurons of the visual system are responsive to input stimuli that assume specific forms. Moreover, Hubel's and Wiesel's studies indicated that the process of vision formation is critically dependent on the selective responsiveness of individual nerve cells to specific forms of input stimuli. A key figure from Hubel's and Wiesel's experiments are depicted in panel (B) of figure 1.8.

The idea of neuronal selectivity was explored by Hubel and Weisel in a spatial context, i.e. the orientation of lines in space were utilized to derive tuning curves for individual neurons in the cat visual cortex. One of the ideas that we wish to explore through this thesis is how neuronal selectivity may manifest itself in a temporal context in the nervous system. Under such a framework, the input stimuli would constitute the timing of input stimuli relative to each other, and the output would correspond to the activity level of individual neurons. Thus we wish to consider a neuron model whose activity level is dependent upon the timing of individual input stimuli, and how such a neural model could be utilized in neural networks.

Furthermore, as is evident from figure 1.8, an input stimulus results in receptive neurons generating a pattern of action potentials, rather than a single impulse. One of the implications of Hubel's and Weisel's findings is that the nervous system encodes information in the frequency of action potential generation, over a period of time. However, the notion of representing information through a neuron's temporal activity pattern is a novel idea in neural networks. Therefore, through this project we explore how such a time-dependent framework of information encoding could be implemented in neural networks.

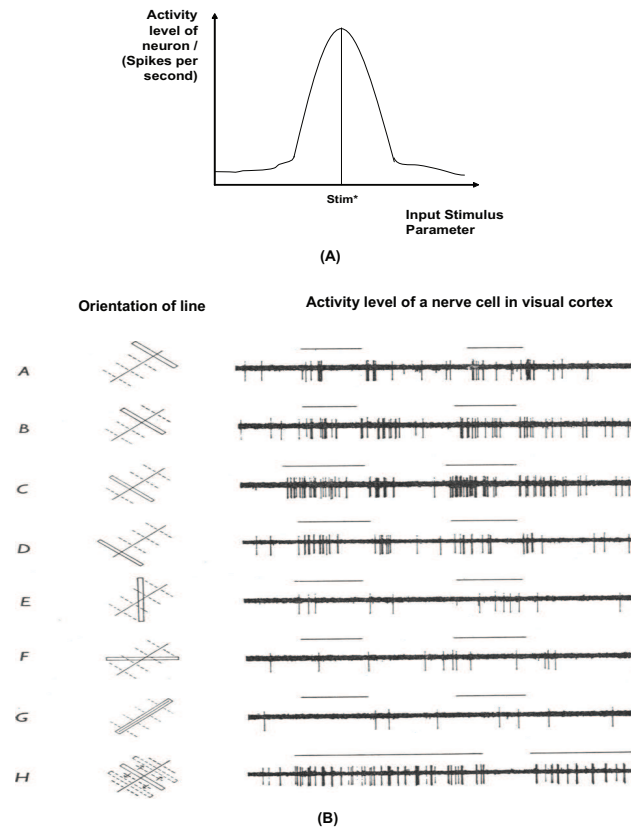


Figure 1.8: (A) The general method of deriving tuning curves. An input stimulus parameter is varied, and for each value of stimulus, the activity level of an individual nerve cell is quantified. The activity level is measured in the average number of action potentials generated by the cell per unit time, and thus reflects the average frequency of action potential generation. In this instance, the cell is most responsive when the stimulus Stim^* is applied. However, tuning curves may come in a variety of shapes, where a given nerve cell may be maximally responsive to more than one value of the input parameter. (B) The classic Hubel and Weisel experiment: adopted from fig. 4 of [27]. Through a narrow slit opening an anesthetized cat is shown various configurations of a line (left panel). The right panel depicts the activity level of a representative nerve cell in response to the specific orientation and position of the line.

1.5 Frequency and Time Dependence in Neural Networks

The notion of encoding information in the frequency of action potentials is yet to be explored in neural networks. The dynamics of the integrate-and-fire (IAF) model entail that the neuron is most responsive to high frequency inputs. Thus the frequency of an input stimulus is directly correlated to the frequency of action potential generation. Moreover, the dynamics of the IAF neuron is characterized by a monotonic function, and thus the membrane potential cannot undergo subthreshold oscillations.

However, biophysical neural models suggest that there exist two major modes of neuronal dynamics - that of i) integrators, and ii) resonators. Integrators are not capable of displaying subthreshold oscillations, while resonators are. Due to the presence of these subthreshold oscillations, resonators are most sensitive to inputs within a certain frequency range, while integrators are most sensitive to high frequency inputs. The frequency selectivity of resonator neurons provides a novel framework for the derivation of decision boundaries within a neural network, and thus expands the possible repertoire of problems that the network is capable of classifying.

Moreover, experimental and theoretical studies have demonstrated that resonator neurons are capable of generating action potentials in response to inhibitory inputs - a phenomenon known as anode break excitation [6], [30], [31]. On the other hand, integrators are not capable of responding to inhibitory stimuli. In the context of neural networks, the IAF model captures some of the qualitative features of integrator neurons in the brain. However, there does not exist a neural model in neural networks that captures the features of resonator neurons.

Furthermore, much of the contemporary literature suggests that neural subthreshold oscillations are a crucial aspect of the mechanisms through which the brain encodes information [32] and [33]. For instance, the oscillatory behavior of a neuronal network in the brain could provide a "background field", and a specific pattern of action potentials could be superimposed on this background field to carry relevant information. A specific instance where the presence of a background field has been experimentally shown arises in the brain of monkeys. It is thought that the relative phase of action potentials with respect to the background field enables the monkey to know its location in space [32]. Thus while the background field encodes a general location, such as a room, the phase of action potentials in relation to this field indicates the monkey's position within the room.

1.6 Project Motivation

One of the motivations of this project is to consider how frequencies may be utilized to encode information in the brain. The notion of a frequency code that encodes relevant information in the brain has become a central tenet of neuroscience, and it has gained relevance in theoretical as well as experimental studies. Thus through this project we strive to explore how frequency and time-dependent neuronal models may be introduced to the field of neural networks. As we have discussed in the previous sections, biophysical neural models suggest that some nerve cells are most sensitive to input stimuli that are applied within a certain frequency range. Furthermore, the derivation of tuning curves along with the insights gained regarding the information processing of the nervous system suggest that the frequency of action potential generation plays a functional role in conveying relevant information in the brain. Therefore, through this thesis we seek to analyze a neural model that displays frequency sensitivity, and is sensitive to the timing of input stimuli. Furthermore, we also consider how such a neural model could be utilized in neural networks, and how the learning procedure could incorporate and reflect the distinction between integrator and resonator neurons. Thus in the subsequent chapter, we analyze the properties of the Resonate-and-Fire neuron.

Chapter 2

The Resonate-and-Fire Neuron

A prevalent notion in contemporary neuroscience suggests that biological nervous systems encode information in the frequency of action potentials. Thus the timing and pattern of multiple action potentials conveys information in the brain. However, the current emphasis in neural networks lies in the ability of a neuron to generate a single action potential, where the state of the neuron is either "on" or "off". In order to implement frequency encoding in neural networks, we must derive an "activation function" that is itself frequency sensitive. Through this chapter we seek to analyze the capabilities of the circuit model depicted in figure 2.1, which we refer to as the resonate-and-fire (RAF) neuron. The RAF model is a specific form of a time-dependent activation function. Thus we are retaining the activation function-based approach to neuronal modeling, while augmenting the neuron models that are currently utilized in neural networks. The RAF model is an extension of the model analyzed by Izhikevich in [34]: while the model in [34] implements a resonator neuron, our model may be tuned to be an integrator or a resonator depending on the choice of parameters. Also, since the RAF model is defined by a linear system of differential equations, the synaptic weights that comprise the input to the perceptron may be trained using the conventional learning procedures of neural networks, such as error backpropagation. Finally, since the RAF neuron is a frequency sensitive model that is dependent on the timing of input stimuli, we may introduce the notion of frequency encoding in neural networks.

2.1 System definition of the resonate-and-fire neuron

The resonate-and-fire neuron may be characterized as follows:

$$Stimulus(t) = I_R + I_C + I_L \Rightarrow Stimulus(t) = \frac{v}{R} + C * \frac{dv}{dt} + \frac{1}{L} \int v dt \quad (2.1)$$

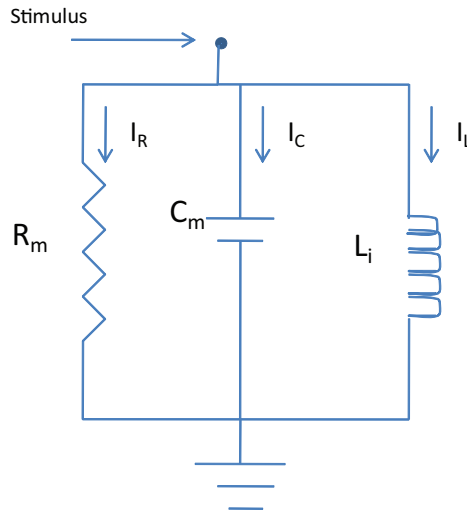


Figure 2.1: The circuit description of the resonate-and-fire neuron. The stimulus refers to a perturbation of the membrane potential (see text). The resistor represents the Ohmic leakage impedance of the neural membrane, and the capacitance models the lipid bilayer that comprises the membrane. The inductor introduces a relatively large time constant to represent the role of long-term currents in neuronal processes

where R_m refers to the membrane resistance, C_m is the membrane capacitance, and L_i represents a memory storage element which operates at a much longer time-scale than the time constant introduced by the capacitance. The notion that underlies the resonate-and-fire neuron is that a biological neuron's internal activity level is determined by the interplay of fast and slow membrane currents. We are mimicing this dynamic interplay of membrane currents by considering time constants that operate at different time scales. Thus the time constant introduced by the capacitance operates

on a short time scale, while the time constant introduced by the inductor functions at a much longer time scale. The resonate-and-fire neuron essentially augments the integrate-and-fire (IAF) model by considering an extra term - the inductance.

The inductor mimics the presence of currents that are dependent on long-term processes, such as the intracellular calcium concentration, The role of calcium currents is known to be crucial in mediating biological learning, and calcium conductances are largely implicated in forms of learning such as long term potentiation (LTP) [1] and [35]. The inclusion of the inductor in the RAF model thus allows us to represent the role of long term currents in the neural membrane through a longer time constant. The interplay of the different time constants introduced by the inductor and capacitor also entail that the RAF neuron is a frequency selective model. We subsequently consider how the choice of parameters in the RAF neuron alter the dynamics of the model.

Upon differentiating, and dividing by C, we may represent the natural response of the resonate-and-fire model as follows, where $I_{stim}(t)$ is zero:

$$\frac{d^2v}{dt^2} + \frac{1}{R * C} \frac{dv}{dt} + \frac{v(t)}{L * C} = 0 \quad (2.2)$$

$$\Rightarrow 2 * \delta * \omega_n = \frac{1}{R * C} \quad \text{and} \quad \omega_n^2 = \frac{1}{L * C} \quad (2.3)$$

Since the resonate-and-fire model is defined by a linear time-invariant system, we may utilize a differential operator in equation 2.2 to obtain the natural roots of the system. Implementing the differential operator is equivalent to taking the Laplace Transform of equation 2.2, with all initial conditions set to zero. Thus we have:

$$V(s) * [s^2 + s * \frac{1}{R * C} + \frac{1}{L * C}] = 0 \quad (2.4)$$

If we solve for s in equation 2.4, we obtain the natural roots of the resonate-and-fire model. Since the resonate-and-fire neuron is defined by a second order differential equation, the system has two natural roots, which may be expressed as:

$$s_{1,2} = \frac{-1}{2 * R * C} \pm \sqrt{\frac{1}{4 * (R * C)^2} - \frac{1}{L * C}} \quad (2.5)$$

$$\Rightarrow s_1 = -\delta * \omega_n + \sqrt{(\delta * \omega_n)^2 - \omega_n^2} \quad (2.6)$$

$$\Rightarrow s_2 = -\delta * \omega_n - \sqrt{(\delta * \omega_n)^2 - \omega_n^2} \quad (2.7)$$

Consequently, the solutions of the resonate-and-fire model may be represented as:

$$v(t) = A * \exp(-t * s_1) + B * \exp(-t * s_2) \quad (2.8)$$

where the constants A and B are determined by the initial conditions of the system.

The roots in equation 2.5 may be distinct and real, where $\delta * \omega_n > \omega_n$, in which case the system response is said to be overdamped. The overdamped response is qualitatively similar to the integrate-and-fire model, in that the system does not display transient oscillations; in response to depolarizing inputs, the membrane potential monotonically increases towards the threshold. Alternatively, if the roots of the system are complex conjugates, where $\delta * \omega_n < \omega_n$, then the system is said to be underdamped. In the instance where the RAF model is underdamped, the neuron is capable of displaying subthreshold oscillations of the membrane potential, as well as postinhibitory rebound spikes in response to inhibitory inputs. Thus the resulting differential equations of the RAF neuron enable us to combine the two different modes of neural activity, namely that of integrators and resonators (see previous chapter), in a single model. Moreover, by adapting the term $\delta * \omega_n$ in our neuron model, we may alter the internal dynamics of the neuron so that it either exhibits the characteristics of integrator or resonator neurons. We shall delve further into the adaptation of internal variables when we derive the learning procedure for resonator neural networks in chapter 4.

Currently subthreshold oscillatory behavior, or postinhibitory rebound spiking, has not been extensively studied in neural networks. However there exists a large body of evidence that suggests that neural subthreshold oscillations have a functional role in the nervous system [32]. Also, the experimentally observed phenomenon of anode break excitation may be understood in the context of subthreshold oscillations, as we show subsequently through phase space analysis. A short introduction to the method of phase space analysis is attached in appendix C.

2.2 Dynamics of the Resonate-and-Fire Model

Since the resonate-and-fire model is characterized by a second order differential equation, we may rewrite it as a system of two differential equations. Thus, we adopt the following convention:

$$\dot{v} = y \tag{2.9}$$

$$\Rightarrow \dot{y} = -\frac{1}{R * C}y - \frac{1}{L * C}v \tag{2.10}$$

$$\tag{2.11}$$

We may evaluate the equilibrium points of the system shown above by considering the two nullclines, defined by $\dot{v} = \frac{dv}{dt} = 0$, and $\dot{y} = \frac{dy}{dt} = 0$. The intersection of the nullclines of an N-dimensional dynamical system correspond to equilibrium points of the system, i.e. points where the differential rate of change is zero. Thus the nullclines of the resonate-and-fire model are:

$$\dot{v} = y = 0 \tag{2.12}$$

$$\dot{y} = 0 = -\frac{1}{R * C}y - \frac{1}{L * C}v \tag{2.13}$$

$$\Rightarrow y = -\frac{R}{L}v \tag{2.14}$$

$$\Rightarrow y = -\frac{\omega_n * v}{2 * \delta} \tag{2.15}$$

Depending on the duration of an external stimulus, $I_{stim}(t)$, the effect on the membrane potential may be regarded as either: i) a prolonged stimulus shifts the y-nullcline, and thus changes the equilibrium of the system, or ii) a momentary stimulus perturbs the membrane potential of the system, thus altering the phase space representation of the system. In the first instance, the prolonged stimulus may be viewed as a current stimulus, which transiently charges and discharges the capacitor and the inductor. On the other hand, the second instance refers to the case where we apply an instantaneous perturbation to the the membrane potential, resulting in the membrane potential assuming the value of the external stimulus, and subsequently

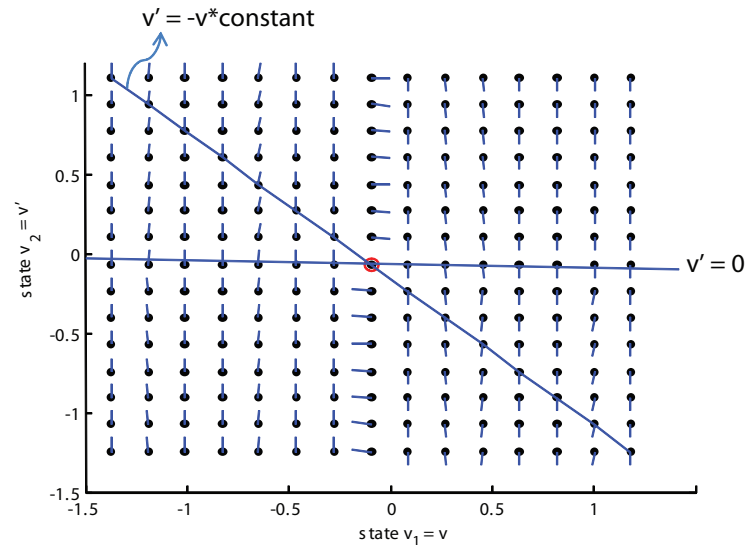


Figure 2.2: Phase portrait of the resonate-and-fire neuron and system nullclines. The resonate-and-fire model's equilibrium is determined by the intersection of the two nullclines defined in equation 2.12.

evolving according to the system defined in equation 2.2. We consider this second case in further detail below. The nullclines of the resonate-and-fire model are depicted in figure 2.2.

2.3 Application of a single stimulus to resonate-and-fire neuron

A momentary perturbation applied to the RAF neuron deviates the membrane potential from the original equilibrium, and subsequently, the potential evolves according to the differential equation defined by equation 2.1. We may analyze the evolution of the membrane potential either in the time domain, or in the phase space. There exist two alternative ways in which we may consider the effect of an excitatory stimulus. We may regard it as an input stimulus current to the RAF circuit, or we may treat it as a perturbation to the membrane potential. We adopt the second alternative, in that an excitatory stimulus perturbs the membrane potential to the magnitude of the stimulus, and subsequently the membrane potential evolves according to equation 2.2.

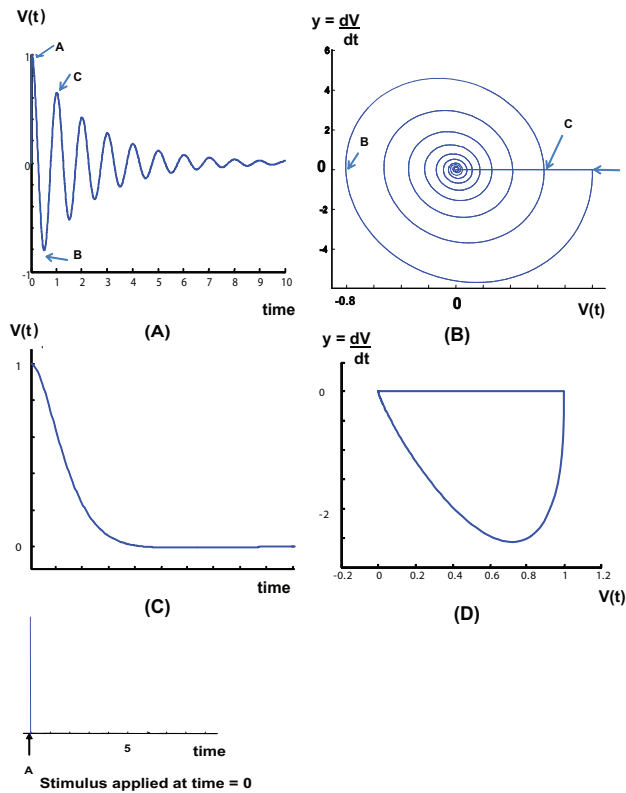


Figure 2.3: The resonate-and-fire dynamics in response to a single excitatory stimulus. An excitatory stimulus is applied at time = 0, labelled as point A in figure. (A) The time response of the resonate-and-fire neuron with $\delta = 0.1, \omega_n = 2 * \pi$. (B) Representation in Phase Space with $\delta = 0.1, \omega_n = 2 * \pi$. Upon application of the first stimulus, the timing of a subsequent stimulus is critical to determining the neuron’s likelihood of firing. Thus a depolarizing stimulus applied at point C is more likely to result in the generation of an action potential than if the same stimulus is applied at point B. We may represent this phenomenon in both the time domain, and in the phase space as is shown in the figure. Threshold omitted for simplicity. (C) The time response of the resonate-and-fire neuron with $\delta = 0.9, \omega_n = 2 * \pi$. Upon application of the stimulus, the membrane potential does not undergo subthreshold oscillations, but rather evolves much like an integrator neuron. (D) Representation in Phase Space with $\delta = 0.9, \omega_n = 2 * \pi$. Therefore the neuron may be tuned to either be an integrator or a resonator through the adjustment of the damping factor

The time domain and phase space representations of the evolution of the membrane potential in response to an excitatory stimulus, are depicted in figure 2.3.

An input stimulus that is applied at time=0 perturbs the membrane potential of the RAF neuron, and this is labelled as point A in figure 2.3. We may represent explicit solutions to this perturbation by considering the following initial conditions:

$$v(0) = Stimulus \quad (2.16)$$

$$v'(0) = 0 \quad (2.17)$$

where Stimulus refers to the magnitude of the perturbation, and $v'(0)$ represents the derivative of the membrane potential at time = 0. We may evoke these conditions in order to solve for the constants A and B in equation 2.8. Thus we have:

$$v(0) = Stimulus = A + B \quad (2.18)$$

$$v'(0) = -s_1 * A - s_2 * B = 0 \quad (2.19)$$

where s_1 and s_2 refer to the roots of the system as defined in equation 2.5. If we utilize the results of equation 2.18, we get:

$$A = Stimulus / (1 - (s_1/s_2)) \quad (2.20)$$

$$B = Stimulus / (1 - (s_2/s_1)) \quad (2.21)$$

If we utilize A and B from the above equation in equation 2.8, we obtain explicit solutions of the evolution of the membrane potential.

The natural dynamics of the RAF entails that the excitatory stimulus evokes damped oscillations of the membrane potential if the damping factor of the neuron is between a value of 0 and 1. In the phase space, we may represent these damped oscillations as a stable focus. Moreover, the timing of a subsequent stimulus relative to the phase of the oscillation is critical in determining the likelihood that the membrane potential exceeds the threshold, and the neuron generates an action potential. Thus a second excitatory stimulus may enhance the effect of the first stimulus, and amplify the evolution of the membrane potential, if the stimulus is presented after a delay equivalent to the natural period of the neuron, labelled as point "c" in figure 2.3.

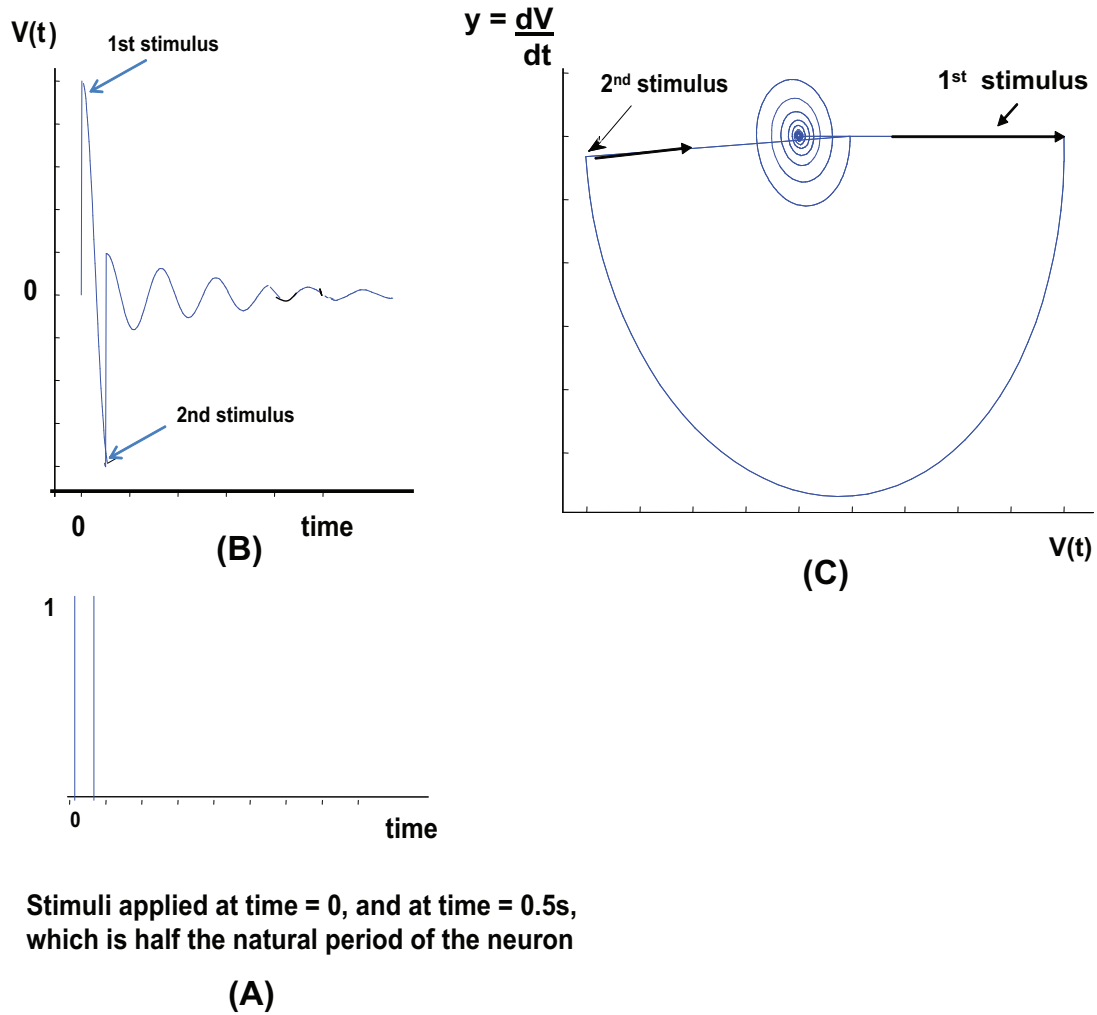


Figure 2.4: The resonate-and-fire dynamics in response to a pair of input stimuli. (A) An excitatory stimulus is applied at time = 0, and then subsequently at time = 0.5s - which corresponds to half the natural period, a second excitatory stimulus is applied (B) The time response of the resonate-and-fire neuron. The second stimulus acts to damp the response evoked by the first stimulus (C) Representation in Phase Space. Parameter values: $\delta = 0.1, \omega_n = 2 * \pi$. Threshold omitted for simplicity. Thus a second excitatory stimulus applied at a specific time reduces the neuron’s likelihood of firing. See text for further explanations.

On the other hand, if the second excitatory stimulus is presented after a delay that corresponds to half the natural period of the neuron, around point "b" in figure 2.3, then the neuron becomes less likely to generate an action potential.

Moreover, as is evident in figure 2.3, the resonate-and-fire neuron may also display the qualitative features of integrator neurons if we adjust the damping factor to be around equal or greater than 1. Under such a scenario, the application of an input stimulus does not evoke subthreshold oscillations of the membrane potential. Instead, the membrane potential monotonically decays toward the original equilibrium point. Therefore, the resonate-and-fire neuron is able to mimic the two major forms of neural activity in nervous systems, i.e. that of integrators and resonators. Through the adjustment of the damping factor that characterizes the neuron, we may alter its activity to either mimic integrator neurons or resonator neurons.

2.4 Application of a pair of input stimuli to the Resonate-and-Fire Neuron

In figure 2.4, a second excitatory stimulus applied in the falling phase of the membrane potential's oscillation reduces the activity level of the neuron, and thus reduces the likelihood of the neuron generating an action potential. This is due to the fact that subsequent to the initial stimulus, the second stimulus is applied after a delay that corresponds to half the natural period of the neuron. Thus the stimulus is applied at the falling phase of the oscillation, and this precise timing damps the membrane potential. If instead of an excitatory stimulus, an inhibitory stimulus was utilized to perturb the neuron subsequent to the initial excitatory stimulus, then the neuron would become more likely to generate an action potential. However, the crucial aspect of the input stimulus is the exact time at which it is presented. Thus subsequent to the excitatory stimulus at time=0, if an inhibitory stimulus is applied after a delay that corresponds to half the natural period of the neuron, then the neuron would become more likely to generate an action potential.

In figure 2.5, we depict the time domain and phase space representations of the membrane potential in response to a pair of stimuli. An excitatory pulse is again applied at time = 0. Subsequently, after a delay that corresponds to half the natural period of the neuron, an inhibitory stimulus is applied. The inhibitory stimulus enhances the neuron's likelihood of firing because it is applied during the falling phase

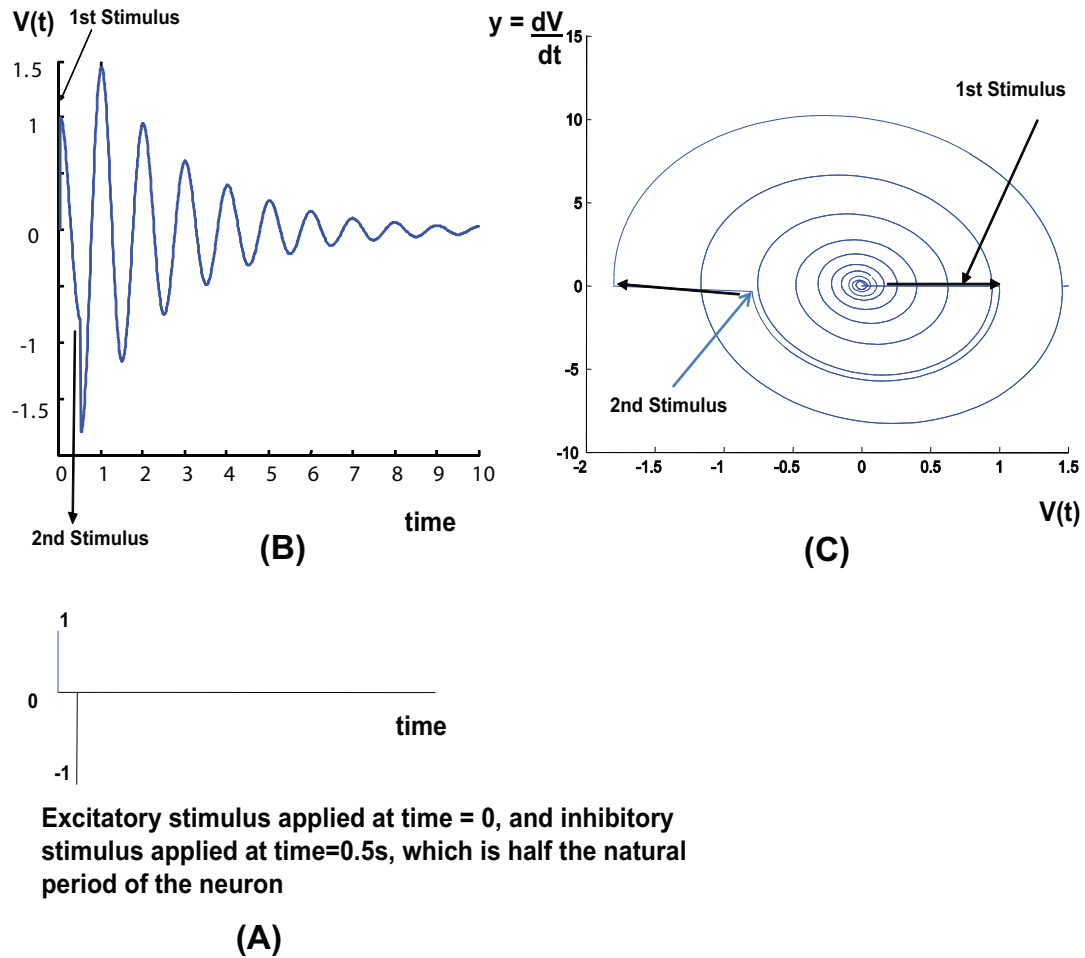


Figure 2.5: The resonate-and-fire dynamics in response to postinhibitory facilitation. (A) An excitatory stimulus is applied at time = 0 and then subsequently at time = 0.5s, which corresponds to half the natural period, an inhibitory stimulus of the same magnitude is applied. (B) The time response of the resonate-and-fire neuron. The second stimulus acts to enhance the response evoked by the first stimulus (C) Representation in Phase Space. The inhibitory stimulus increases the "radius" of the stable focus, thus indicating an enhanced likelihood of firing. Parameter values: $\delta = 0.1, \omega_n = 2 * \pi$. Threshold omitted for simplicity. Thus an inhibitory stimulus applied at a specific time increases the neuron's likelihood of firing. See text for further explanations.

of the oscillation, around the negative peak of the membrane potential. Thus during the rebound oscillation, the neuron evolves through a greater value of membrane potential, in comparison to the case where solely the excitatory stimulus is applied. Therefore, the resonate-and-fire model is capable of displaying subthreshold oscillations of the membrane potential, and as a consequence of this, the neuron may also exhibit postinhibitory rebound spiking. The phenomenon of postinhibitory rebound spiking refers to the instance where a neuron generates action potentials in response to inhibitory stimuli. Through both theoretical and experimental analyses, many neurons have been shown to be capable of exhibiting postinhibitory rebound spikes [6]. One of the limitations of the integrate-and-fire neuron is that the model is not capable of generating action potentials in response to inhibitory inputs. If inhibitory input is applied to the integrate-and-fire model, the neuron will never reach threshold and thus the neuron will never generate action potentials. This limitation of the integrate-and-fire model is depicted in figure 2.6.

2.5 Parameters and Possible Configurations of Resonate-and-Fire Neuron Model

The scenarios that we have considered so far have been limited to instances where we perturb the neuron with a single excitatory stimulus, followed by a precisely timed excitatory or inhibitory stimulus. The general insights gained from these instances suggest that the resonate-and-fire neuron is acutely sensitive to the timing of input stimuli, and that a particular neuron is tuned to be most sensitive to inputs that arrive at a specific frequency. We may extend our approach to the instance where we perturb the neuron with multiple excitatory stimuli through one input, while also applying inhibitory stimuli from a second input. At any given time, the instantaneous input to the neuron constitutes the sum of the instantaneous stimuli received through each input. This specific framework is shown in figure 2.8.

Let us first consider the case where the inhibitory input is set to zero for all time, while the neuron is receiving excitatory inputs at some input frequency. Stimuli that are delayed with respect to each other, by a time period that corresponds to the membrane potential's natural period of oscillation tend to enhance the neuron's probability of firing; on the other hand, stimuli that are applied at delays that are significantly different from the natural period of oscillation diminish the probability of firing. The case where a train of stimuli enhances the likelihood of firing is depicted

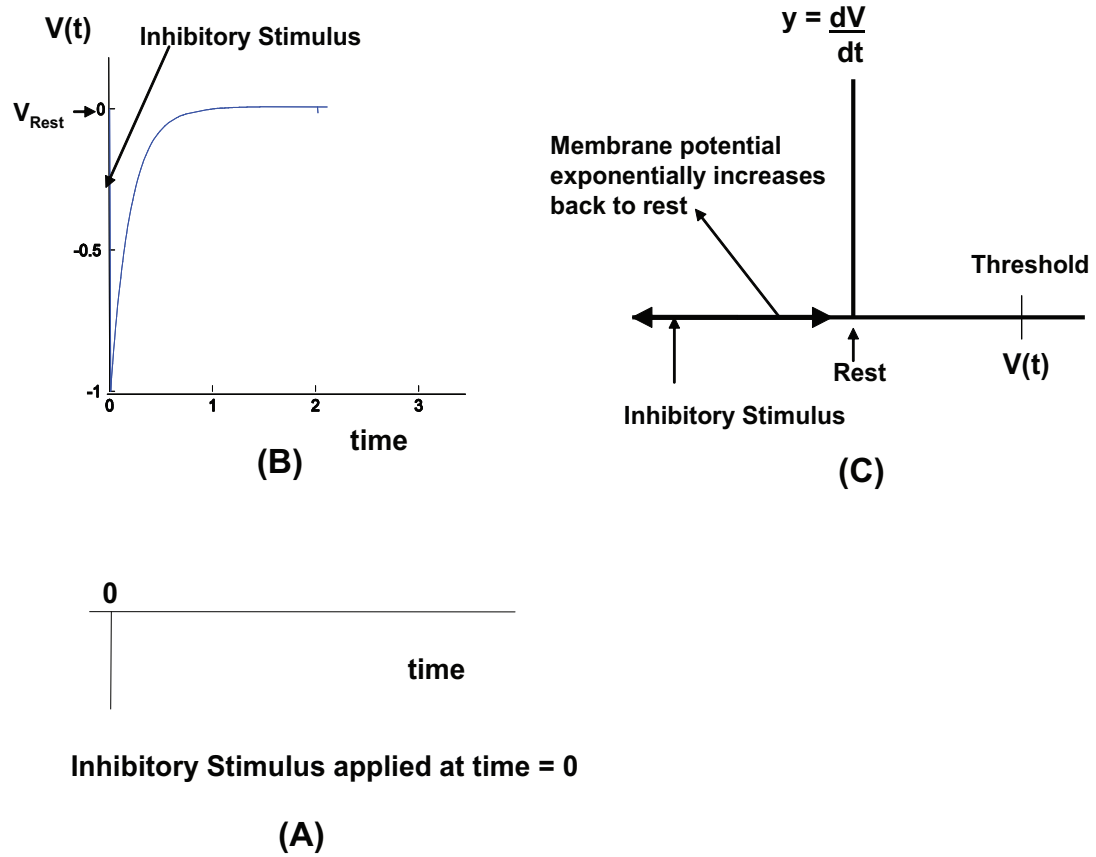


Figure 2.6: The integrate-and-fire (IAF) neuron in response to inhibition. (A) An inhibitory stimulus is applied at time=0 (B) Time domain representation: the membrane potential exponentially rises back to rest subsequent to being inhibited. Since the integrate-and-fire neuron cannot display subthreshold oscillations of the membrane potential, it cannot generate action potentials in response to inhibitory inputs. (C) Phase space representation of IAF neuron's response to inhibition. Parameter value: $R \cdot C = 0.2$.

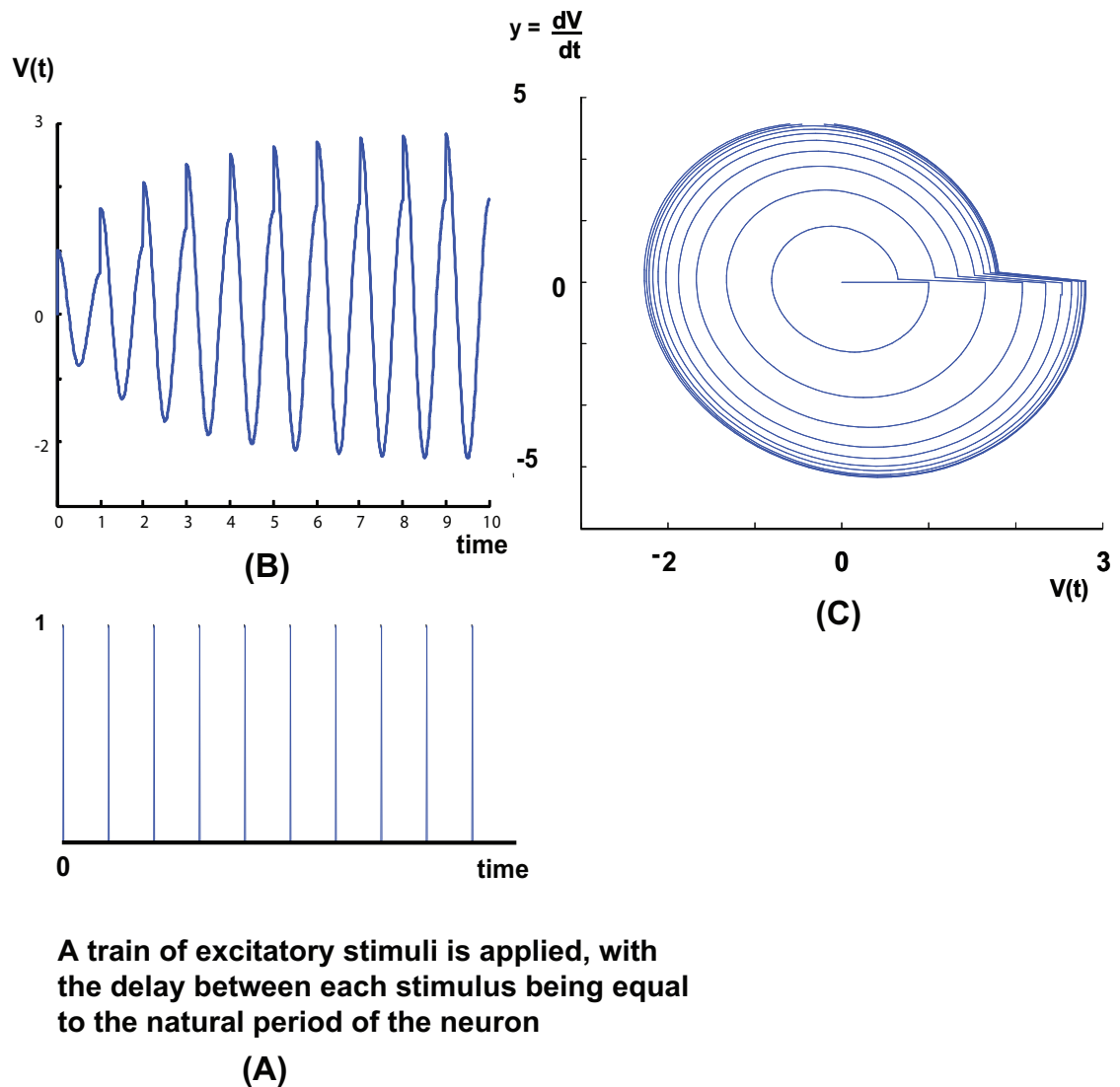


Figure 2.7: The evolution of the resonate-and-fire neuron's membrane potential in response to a train of excitatory stimuli. The input train is applied at a frequency that corresponds to the neuronal membrane potential's natural frequency of oscillation. (A) Input stimulus train (B) Time response of membrane potential (C) Phase space representation of membrane potential's evolution. Parameters: $\delta = 0.1, \omega_n = 2 * \pi$. Threshold omitted for simplicity.

in figure 2.7. We may alter this probability of firing at particular instants in time by applying a second input at precisely timed intervals. Thus in the case where an excitatory and inhibitory input are applied at the same time, they effectively cancel each other out. When the inputs are applied at a certain phase relative to each other, we may manipulate the neuron's membrane potential by controlling the phase between the inputs. This general framework is depicted in figure 2.8.

From our discussion in this chapter, we may apply a certain pattern of input stimuli through two generic methods: i) we apply input stimuli to the RAF neuron through a single input, with each input stimulus applied at a certain delay relative to one another; this particular instance corresponds to the case where we have a single input to the RAF neuron, and each of the input stimuli are applied at a relative delay with respect to each other through the same input line, and ii) there exist multiple inputs to the resonate-and-fire neuron, and each of the input patterns applied are related to each other through some difference in timing. The first instance represents temporal summation in that there exists a single input, and each stimulus is delayed by a certain amount relative to the previous stimulus. The second instance corresponds to spatial summation since the neuron receives input stimuli through multiple presynaptic channels. These two general approaches are depicted in figure 2.9.

In the subsequent chapter we characterize the RAF neuron's response to different profiles of spatial and temporal input patterns, and we subsequently analyze the neuron's capabilities in classifying logic gates, and in solving the XOR problem.

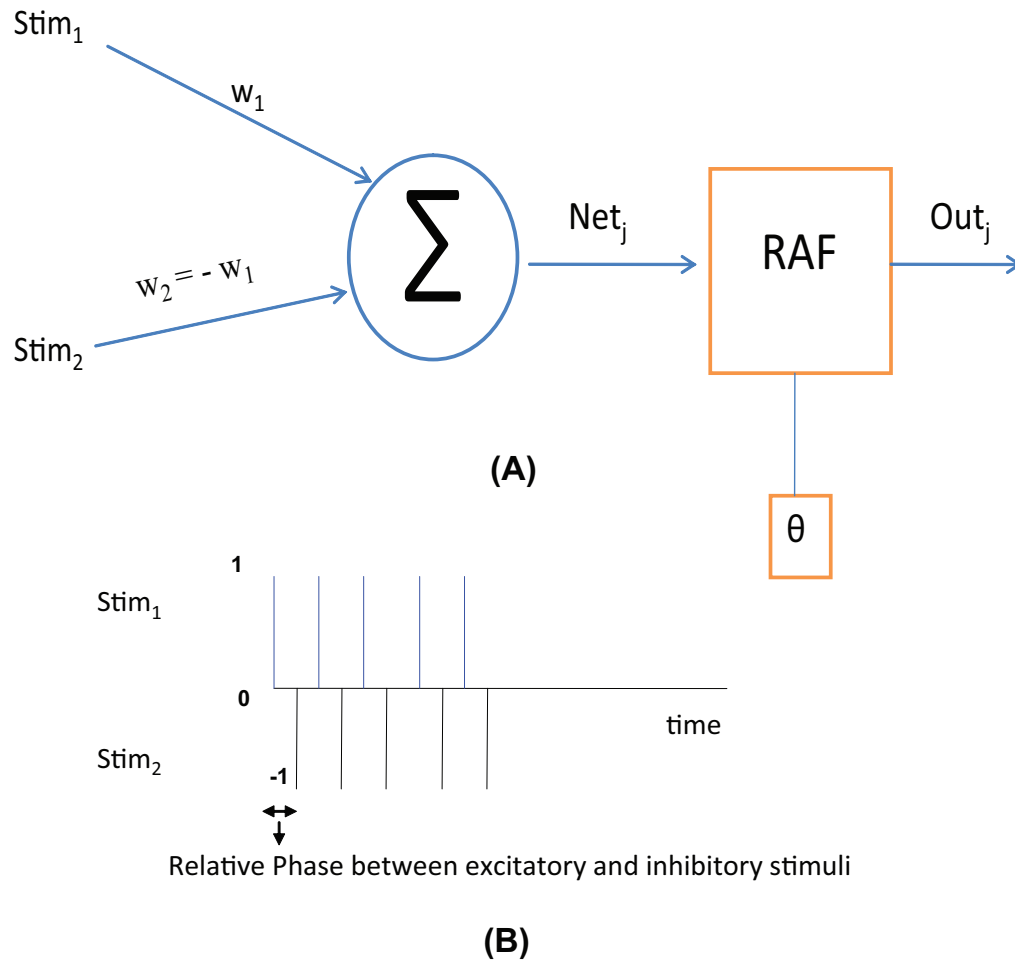


Figure 2.8: The resonate-and-fire neuron with two inputs, one excitatory, and the other inhibitory. (A) Schematic representation of the resonate-and-fire neuron receiving two inputs of equal magnitude, but of opposite sign. (B) The relative delay between the excitatory and inhibitory inputs may be manipulated to either enhance or inhibit neural activity.

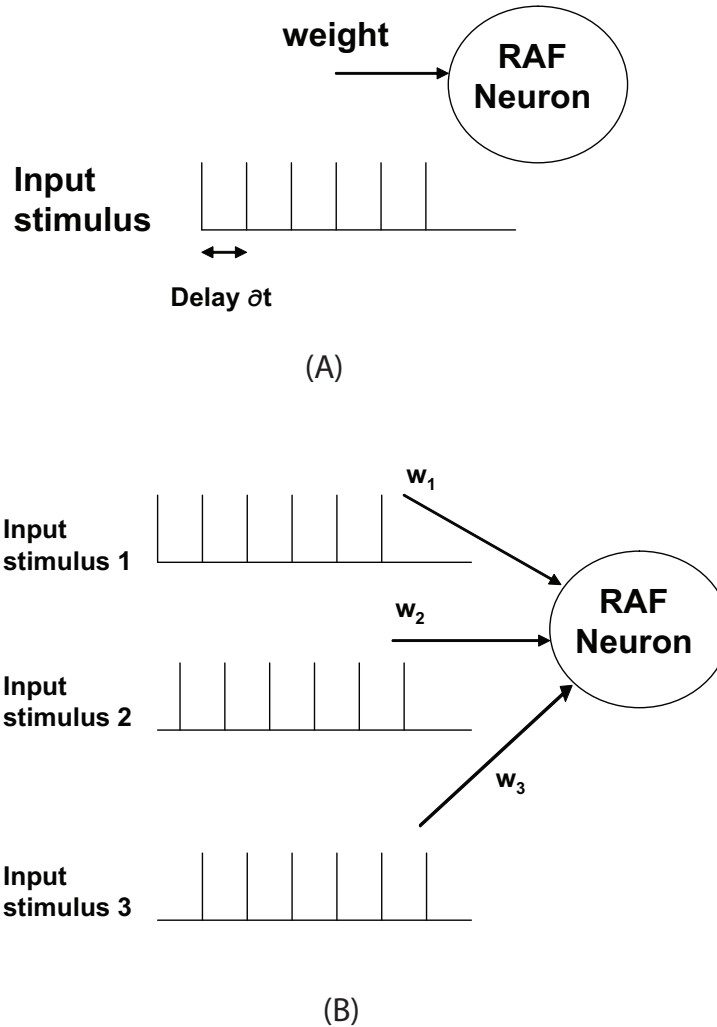


Figure 2.9: Possible input configurations to the resonate-and-fire (RAF) neuron. (A) The RAF neuron receives inputs through a single input channel, and each input stimulus is offset in time from the previous stimulus by a certain delay. (B) The resonate-and-fire neuron receives inputs through multiple input channels, and the stimuli applied to each input are delayed with respect to one another.

Chapter 3

The Classification of Logic Gates and The XOR Problem

The classification of different input patterns as belonging to a specific "class" was one of the earliest problems explored in the field of neural networks. The inspiration underlying the classification problem is that biological nervous systems are extremely adept at recognizing patterns, such as the letters of an alphabet, or the facial profile of different individuals. The classification problem was originally studied in the context of binary logic gates. The McCulloch-Pitts (MP) neuron that we reviewed in our introductory chapter, has been shown to be adept at correctly classifying simple logic expressions, such as the "OR" and "AND" gates. However, the MP model is not capable of classifying linearly inseparable patterns such as the XOR problem. The limitations of the MP neuron in the context of the XOR problem served to become the basis of many of the criticisms that neural networks faced in the early years of the field [36], and the XOR problem continues to be an important benchmark problem in neural networks. Through this chapter we explore the capabilities of the resonate-and-fire (RAF) neuron in classifying logic expressions, and we show that the RAF model is capable of solving the XOR problem. Subsequently, we investigate how the process of frequency encoding could be achieved in a neural network comprised of RAF neurons.

3.1 The "OR" and "AND" problems

Let us consider the logic that defines the OR and AND gates - depicted in figure 3.1. We may derive a decision boundary which separates the input space into two distinct regions - if a point lies above the boundary, then the output is considered to be "one", and conversely, if a point lies below the boundary, then the output is "zero". In the case of the classic McCulloch-Pitts neuron, the output is defined as:

$$out = w_1 * x_1 + w_2 * x_2 - \theta \quad (3.1)$$

and thus the decision boundary is defined by:

$$0 = w_1 * x_1 + w_2 * x_2 - \theta \quad (3.2)$$

If we solve equation 3.2 for x_2 , we obtain a decision boundary in the form of a straight line, that segments the input space into two different classes. Thus the decision boundary is:

$$x_2 = -\frac{w_1}{w_2} * x_1 + \frac{\theta}{w_2} \quad (3.3)$$

Moreover, we may adjust the slope and y-intercept of the decision boundary by adjusting the free parameters of the system, i.e. the weights and threshold. We next explore how these two elementary logic problems may be classified using the resonate-and-fire model.

3.1.1 Single Stimulus Solution to the "OR" and "AND" problems with the Resonate-and-Fire Neuron

The "OR" and "AND" logic classifications may be solved through a variety of different approaches in the resonate-and-fire (RAF) model. Let us consider the simplest instance, where the inputs are presented to the RAF neuron at a single time instant. For simplicity, let us also assume that the inputs are excitatory and of equal magni-

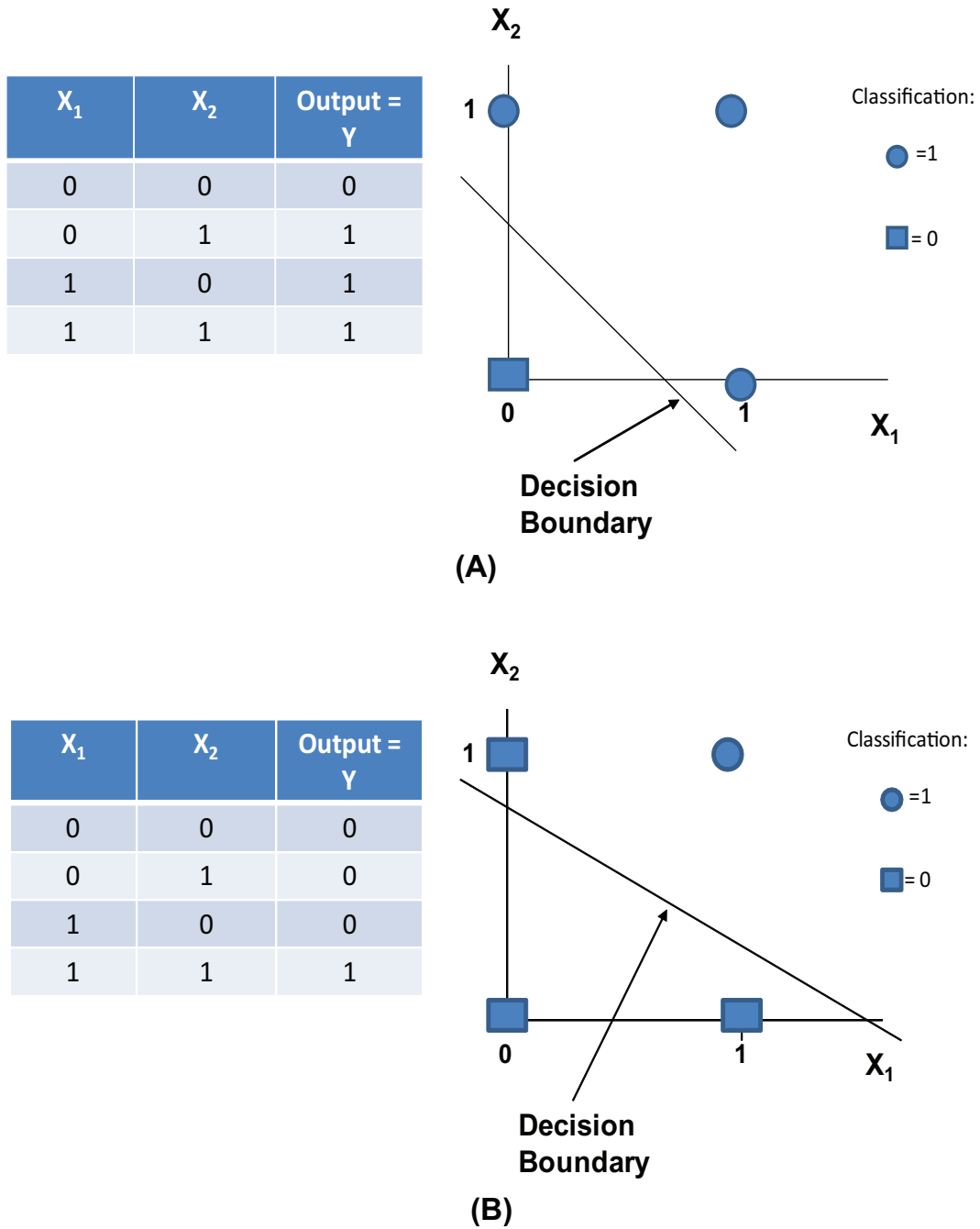


Figure 3.1: (A) The "OR" Problem. If either of the inputs are a logical "one", then the output of the neuron should also be "one". (B) The "AND" Problem. If both inputs are logical "one", then the output is "one". The decision boundary segments the input space into two different classes: "zero" and "one". If the input lies above the decision boundary, then the output is considered "one", otherwise the output is "zero".

tude - thus the synaptic strengths are both positive and of equal magnitude. This general framework is depicted in figure 3.2.

Let us consider the response of the resonate-and-fire model:

$$\dot{v} = y = Stimulus_1(t) + Stimulus_2(t) \quad (3.4)$$

$$\Rightarrow \dot{y} = -\frac{1}{R * C}y - \frac{1}{L * C}v \quad (3.5)$$

$$(3.6)$$

where v refers to the membrane potential, and R, C , and L are the membrane resistance, capacitance, and inductance respectively. Furthermore,

$$\begin{aligned} Out(t) &= 1 && \text{if } v \geq V_{Threshold} \\ Out(t) &= 0 && \text{otherwise} \end{aligned} \quad (3.7)$$

where $Out(t)$ refers to the output of the neuron for classification purposes, and $V_{Threshold}$ is a constant.

In the instance where we wish to solve the "OR" problem, we must satisfy the condition that when the input to the perceptron is $(0,0)$, our output must be zero. For all other inputs, the output must be one. We may satisfy these conditions by applying a very small threshold - for instance, a threshold of zero. Thus when the input is $(0,0)$, there does not exist a net input to the perceptron. When the input is $(1,0)$ or $(1,1)$, the output of the perceptron exceeds the threshold, and thus our binary output is considered "one". The resonate-and-fire solution for the "OR" problem with single stimulus input $(1,0)$ is depicted in figure 3.3.

We may adopt a similar method to classify the "AND" problem with the resonate-and-fire neuron. In the context of an "AND" gate, our output should be "one" only when both our inputs assume a binary value of "one". In this instance, we may set the threshold of the resonate-and-fire neuron between the analog values of one and two. Consequently, the inputs $(0,0)$, $(1,0)$ and $(0,1)$ will not cause the membrane potential to exceed threshold, while only the input $(1,1)$ will result in the neuron generating action potentials. The solution to the "AND" problem with the resonate-and-fire model is depicted in figure 3.4. We subsequently review the resonate-and-fire neuron's solution for the XOR problem.

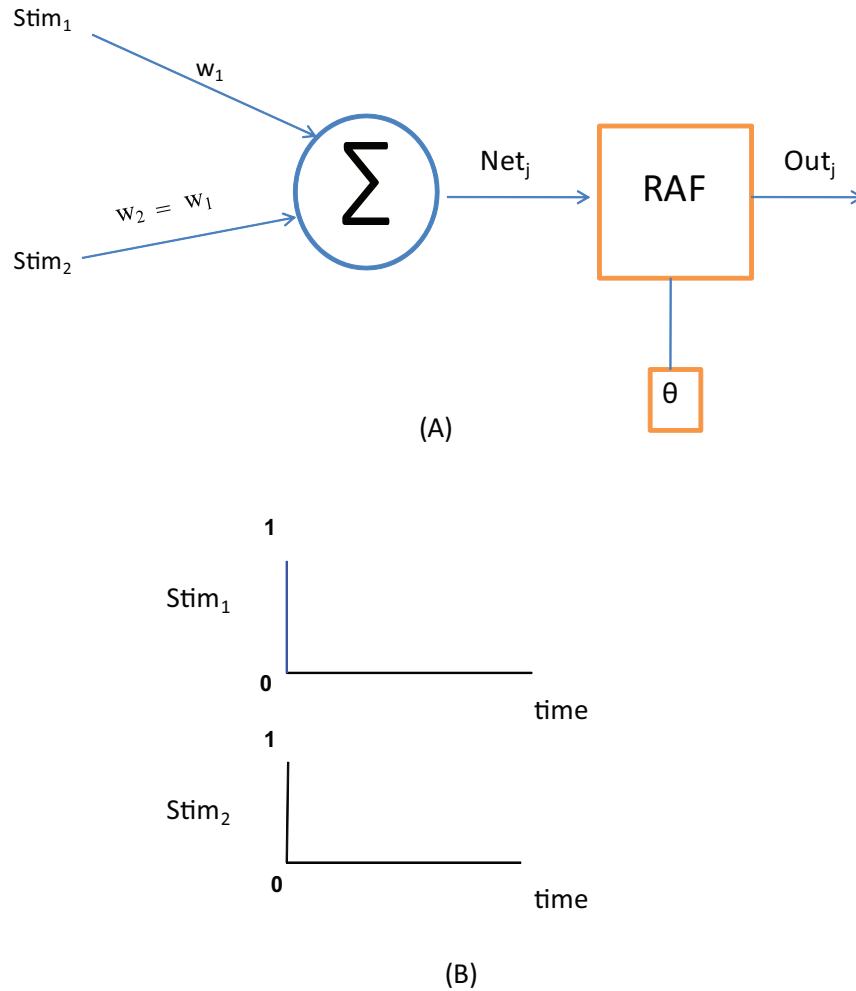
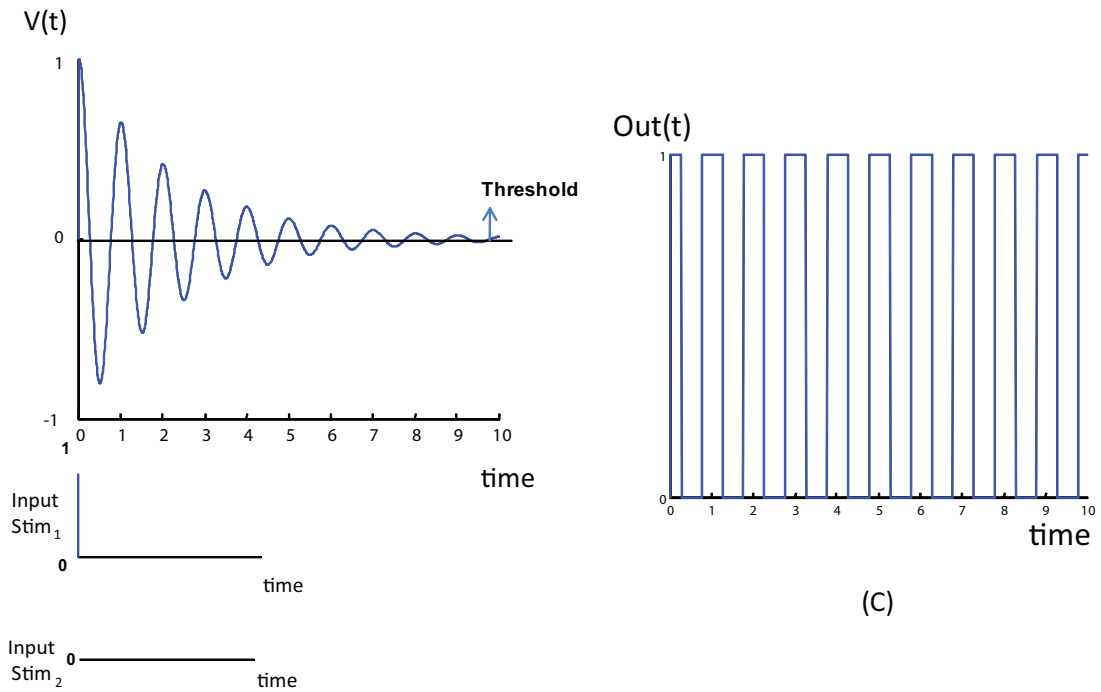
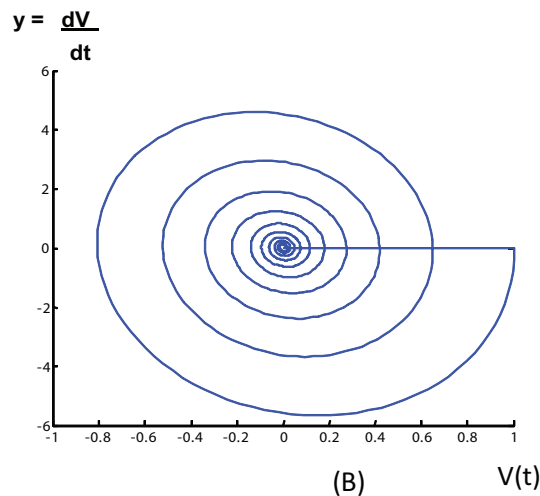


Figure 3.2: The resonate-and-fire model for the solution of the "OR" and "AND" problems. (A) The framework through which we apply input stimuli to the resonate-and-fire model for the "OR" and "AND" problems. Both weights are assumed to be excitatory and equal in magnitude. (B) Timing of input stimuli. In this instance, we present the input pattern (1,1). Thus an input stimulus is applied on both inputs at the same time instant.



(A)



(B)

(C)

Figure 3.3: The "OR" Problem solution with the resonate-and-fire neuron. (A) An input stimulus is applied to the resonate-and-fire neuron at a single time instant. In this instance we consider the input (1,0). The results are qualitatively similar for inputs (1,1) and (0,1). Also depicted in this panel is the time response of the resonate-and-fire neuron. We assign a threshold value of zero, thus each time the membrane potential exceeds the threshold, the binary output (shown in panel (C)) is "one". (B) Phase space representation of resonate-and-fire model in response to input stimulus (1,0). (C) Binary output is "one" each time membrane potential exceeds the threshold value. Parameters: $\delta = 0.1, \omega_n = 2 * \pi$.

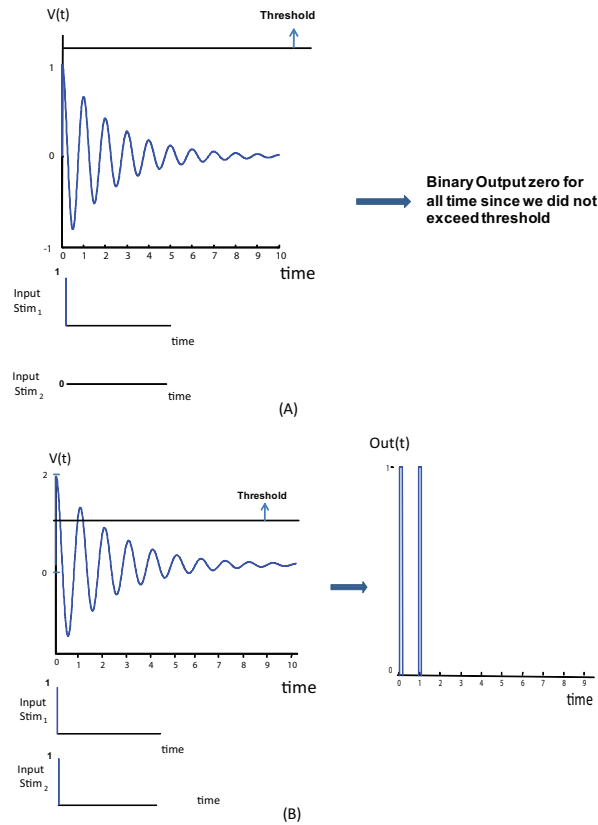


Figure 3.4: Solution to the "AND" Problem with the resonate-and-fire neuron. (A) An input stimulus is applied to the resonate-and-fire neuron at a single time instant. In this instance we consider the input (1,0). Also depicted in this panel is the time response of the resonate-and-fire neuron. We assign a threshold value of 1.2, and thus the membrane potential never reaches threshold. Consequently, the binary output is "zero" for all time. This is the correct classification when the inputs to an "AND" gate are (1,0), (0,1), or (0,0). (B) When the input stimulus (1,1) is applied, the membrane potential exceeds threshold for certain values of time. Thus the binary output is "one" each time the membrane potential exceeds threshold. Parameters: $\delta = 0.1, \omega_n = 2 * \pi$.

3.2 The XOR Problem

One major drawback of the elementary McCulloch-Pitts perceptron is that it is capable of only classifying input patterns that are linearly separable. However, nonlinearly separable patterns are ubiquitous, and a specific example arises in the Exclusive OR (XOR) problem. The XOR problem is a very special instance of a more general problem - that of classifying points within a unit hypercube as belonging to either of two classes - class 0 or class 1. In the specific case of the XOR problem, we are only concerned with the four corners of the unit square. Thus the four input patterns are (0,0), (0,1), (1,0), and (1,1) as is shown in figure 3.5. The elementary perceptron model is not capable of solving the XOR problem, and thus the solution is to utilize a multilayer perceptron network. More recently, Rowcliffe, Feng and Buxton have proposed a solution [37] that utilizes a combination of the integrate-and-fire neuron, and diffusion coefficients. The method proposed in [37] derives a decision boundary as a function of synaptic diffusion parameters, and subsequently adjusts these coefficients to yield the appropriate decision boundary. Here, we propose a much simpler solution to the XOR problem, that exploits the time-dependent capabilities of the resonate-and-fire neuron. The XOR truth table and the required classification in input space is shown in figure 3.5.

In the instance of the resonate-and-fire neuron, we may view the XOR problem in one of two ways:- i) an input is presented to the neuron at a single time instant, and the neuron produces an output based on its internal activity level, or, ii) the XOR problem may be recast into a time-dependent framework, where we perturb the neuron with a train of stimuli, that correspond to the inputs. We consider the first instance in this section, where the input stimuli are presented at a single time instant.

In the instance, where we apply the input stimuli once, there exists a simple way in which we may correctly classify the XOR problem using the properties of inhibition. This method constitutes setting the synaptic weights to be of equal magnitude, but of opposite signs, as is depicted in figure 2.8; however contrary to figure 2.8, we consider the application of a single stimulus. Thus we have:

$$\dot{v} = y = w_1 * Stimulus_1(t) + w_2 * Stimulus_2(t) \quad (3.8)$$

$$\Rightarrow \dot{y} = -\frac{1}{R * C}y - \frac{1}{L * C}v \quad (3.9)$$

$$(3.10)$$

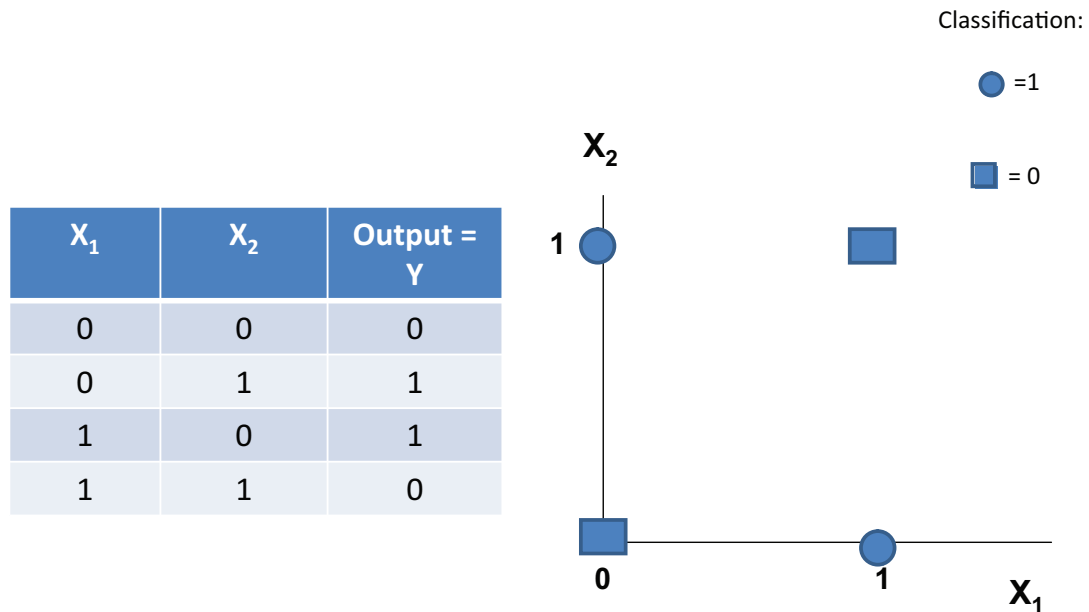


Figure 3.5: The XOR Problem requires that inputs lying on the opposite corners of the unit square be classified as belonging to the same class.

We may introduce a threshold, and apply the decision rule that whenever the membrane potential exceeds the threshold, we consider the binary output of the neuron to be one. In this instance, let us assume a threshold value of zero. When the input is $(1,1)$, the net input to the neuron is zero, since the inputs are weighted by equal weights of opposite signs. Thus the weighted inputs cancel each other. Under such a framework, the inputs $(1,1)$ and $(0,0)$ yield an output of zero, while the inputs $(1,0)$ and $(0,1)$ yield non-zero outputs. The input $(0,1)$ yields an output of "ONE" due to the properties of the RAF neuron with regards to inhibition. Since the net input to the neuron is negative and the model is tuned to operate in the underdamped regime, the membrane potential is capable of undergoing anode break excitation. Thus due to the post-inhibitory rebound of the membrane potential, the RAF neuron is capable of classifying $(1,0)$ and $(0,1)$ as belonging to the same input class "ONE". This is depicted in figure 3.6.

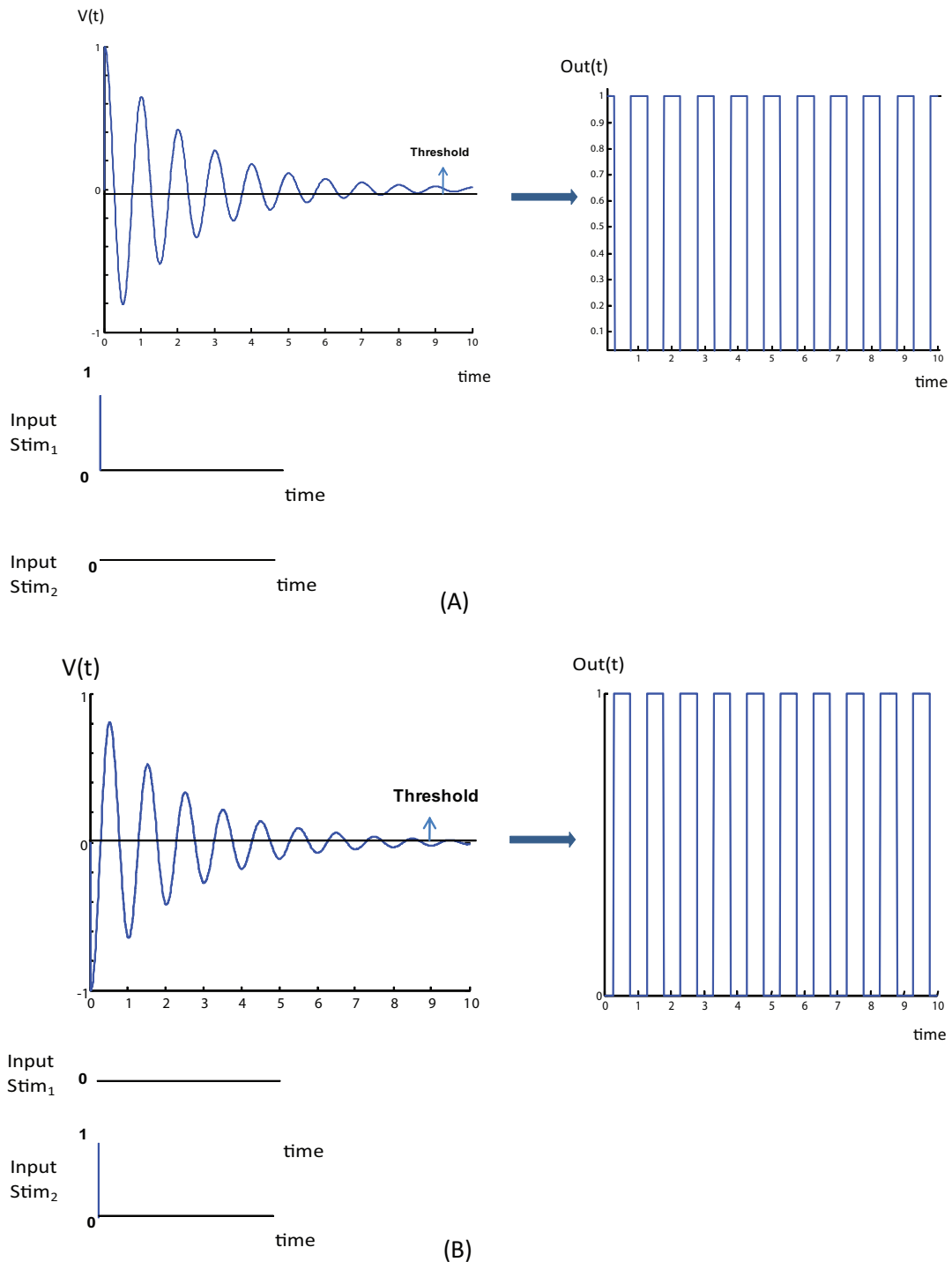


Figure 3.6: A simple solution to the XOR problem with the Resonate-and-Fire Neuron. The inputs are applied at time=0 only, and the input synaptic weights to the neuron are assumed to be of the same magnitude, but opposite signs. In this instance, the threshold is arbitrarily set, and anytime the neuron's membrane potential exceeds the threshold, the binary output is considered to be one. (A) Input pattern (1,0) is correctly classified as "ON". (B) Classification of input stimulus (0,1), which also yields a "ONE". The input cases of (1,1) and (0,0) (not shown in figure) always yield an output of "ZERO" because the net input to the neuron is zero for both cases.

3.3 Application of a train of stimuli to the Resonate-and-Fire neuron

One of the original motivations of this project is to consider how the timing of input stimuli may be utilized by the nervous system to encode information. Thus we wish to consider a time-dependent version of the XOR problem, where an input train of stimuli is applied to the RAF neuron. However, in this section, we elaborate more on the time dependent aspects of the resonate-and-fire model instead of delving into the XOR classification problem. Our objective in this section is to use a time-dependent implementation of the XOR problem's input stimuli in order to highlight the sensitivity of the RAF neuron to the timing of input stimuli. Thus we again have:

$$\dot{v} = y = w_1 * Stimulus_1(t) + w_2 * Stimulus_2(t) \quad (3.11)$$

$$\Rightarrow \dot{y} = -\frac{1}{R * C}y - \frac{1}{L * C}v \quad (3.12)$$

$$(3.13)$$

In the instance of the XOR Problem, once again the inputs (1,1) cancel each other out due to the opposite signs assigned to the synaptic weights. Moreover, the neuron is capable of correctly classifying the inputs (0,1) and (1,0) as belonging to class one if the input stimulus train is presented at the right frequency, i.e. the natural frequency of the RAF neuron. On the other hand, if the same input (1,0), or (0,1), is presented at a frequency that corresponds to half the natural frequency of the neuron, then the RAF neuron classifies the input train as belonging to class "ZERO". This frequency and time-dependence of the RAF model is depicted in figure 3.7.

Therefore, the RAF neuron is extremely sensitive to the timing of input stimuli. As is depicted in figure 3.7, the same input train of stimuli may be presented to the neuron at different frequencies, and the neuron may classify the input as belonging to different classes. Thus the timing of individual stimuli relative to the phase of the evolution of the membrane potential determines the output of the resonate-and-fire neuron. We explore this relationship between the resonate-and-fire neuron's natural frequency and the timing of input stimuli further in the subsequent chapter.

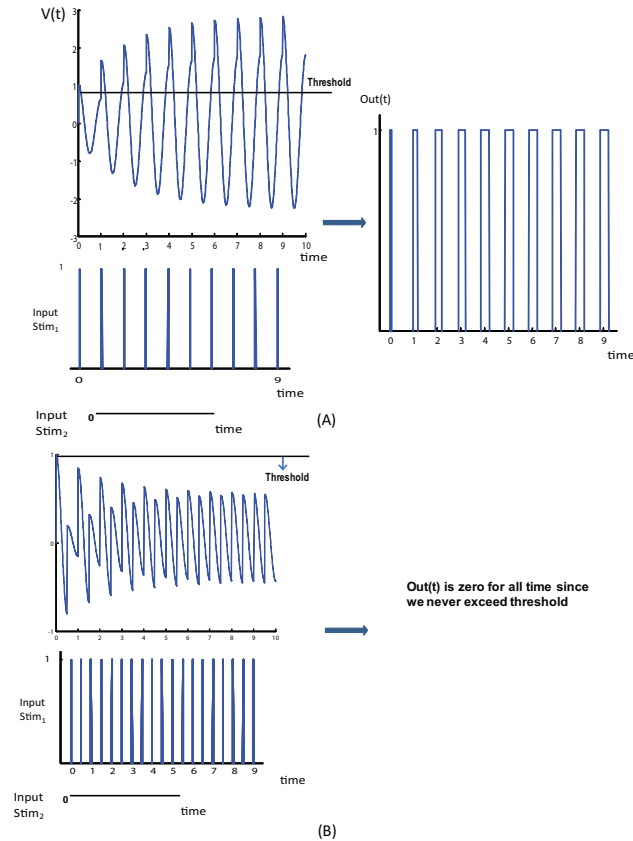


Figure 3.7: The XOR problem with the application of a train of stimuli. The threshold is arbitrarily set, and anytime the neuron’s membrane potential exceeds the threshold, the binary output is considered to be one. (A) In this panel, we depict the input (1,0) being applied to the RAF neuron at the neuron’s natural frequency of 1 Hz. As would be the correct classification in the XOR Problem, the neuron generates a train of action potentials, and thus classifies the input as belonging to class "ONE". (B) The same input (1,0) may be applied at a frequency of 0.5 Hz, which is half the neuron’s natural frequency. In this instance, the membrane potential never exceeds threshold, and thus the neuron classifies the input as belonging to class "ZERO". Therefore, the RAF neuron may classify the same input as belonging to different classes depending on the timing of input stimuli.

Chapter 4

The Temporal Backpropagation Algorithm

The learning procedure represents one of the most important paradigms in neural networks, as well as neurobiology and biophysical neural modeling. The notion of *learning* connotes a wide variety of mechanisms and principles of nervous system function, and the specific interpretation of learning that we adopt depends on the specific approach to neural modeling that we are interested in. However, a generic definition of learning regards the ability of the nervous system to change and reorganize itself in response to external stimuli. Inherent in the process of learning are structural alterations in nervous system architecture, as well as morphological changes in individual neurons [1].

As we reviewed in chapter 1, the learning procedure in traditional neural networks entails the adjustment of synaptic weights according to the static error backpropagation algorithm, whose derivation is attached in appendix B. The static backpropagation algorithm assumes static activation functions, and thus time is not represented explicitly in the network architecture. However, the resonate-and-fire (RAF) neuron is defined by a differential equation whose internal dynamics are critically dependent on the timing of input stimuli. Moreover, the dynamics of the RAF model is dependent on the internal parameters of the neuron, since these parameters determine whether the neuron is an integrator or a resonator. Thus through this chapter we explore the learning procedure to adapt a neural network comprised of RAF neurons. We specifically consider how the backpropagation algorithm may be implemented within time dependent neural networks, and how we may adapt: i) the synaptic structure

of a neural network comprised of RAF neurons, and ii) the internal parameters that characterize each RAF neuron model. Further instances of the implementation of a time dependent version of the backpropagation algorithm are found in [38], [39], and [40]; however each of these applications are significantly different than the adaptation procedure that we derive through this chapter.

Depicted in figure 4.1 is a model for an individual neuron j . The neuron receives input signals from presynaptic neurons i , through p synapses. The input signal on each synapse is weighted by the synaptic weight w_{ij} , and delayed by a time amount τ_{ij} . Thus the net input to the neuron at time t is given by:

$$x_{net,j}(t) = \sum_{i=1}^p w_{ij} * x_i(t - \tau_{ij}) \tag{4.1}$$

where w_{ij} refers to the synaptic weight from neuron i to neuron j , τ_{ij} refers to the time delay that exists on the synaptic connection between presynaptic neuron i and neuron j , and $x_i(t)$ refers to either an externally applied stimulus or the signal originating at presynaptic neuron i at time t . We may model each individual neuron as being a linear time invariant (LTI) system, since the resonate-and-fire neuron is itself such a system. Thus at any time t , the input to the LTI system is the sum of the products of the synaptic weights and input signals to the neuron. Since our model neuron is assumed to be an LTI System, the membrane potential of the neuron, $v_j(t)$ may be obtained through the convolution integral.

Thus we have:

$$\begin{aligned} v_j(t) &= u_j(t) - \theta_j \\ \Rightarrow v_j(t) &= \int_{-\infty}^{+\infty} [h_j(\lambda) * x_{net,j}(t - \lambda)d\lambda] - \theta_j \end{aligned} \tag{4.2}$$

where $h_j(\lambda)$ refers to the impulse response of neuron j with respect to the dummy variable λ , and θ_j refers to an externally applied threshold to the neuron. Since $x_{net,j}(t)$ refers to a summation of weighted input signals, equation 4.2 implicitly includes a summation within the convolution integral. Thus in complete form, equation 4.2 may be represented as follows:

$$v_j(t) = \int_{-\infty}^{+\infty} [h_j(\lambda) * (\sum_{i=1}^p w_{ij} * x_i(t - \tau_{ij} - \lambda))]d\lambda - \theta_j \tag{4.3}$$

Moreover, the output of neuron j is given by:

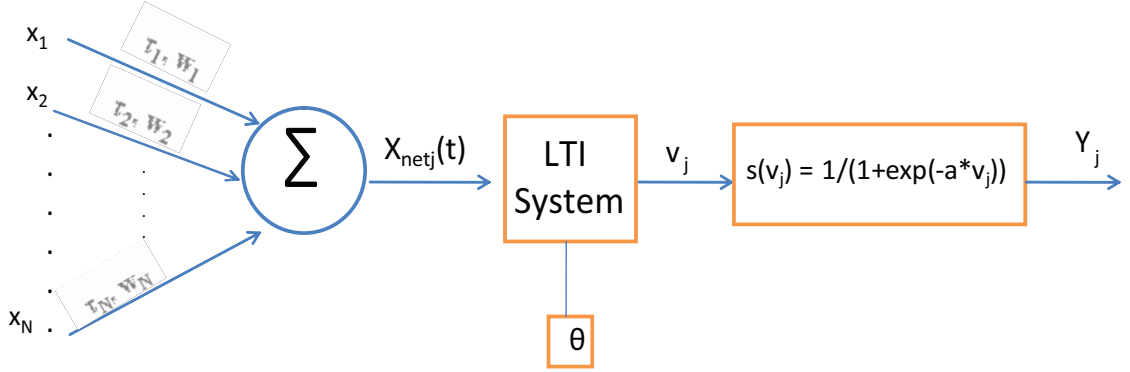


Figure 4.1: Neuron model for temporal backpropagation algorithm. The input to the neuron (LTI System) at any time t is given by $x_{net}(t) = \sum_{i=1}^p w_{ij} * x_i(t - \tau)$, where w_{ij} refers to the synaptic weight from neuron i to neuron j , and τ_{ij} refers to the time delay that a signal experiences as it reaches the neuron j . The output of the LTI system, $v_j(t)$ is passed through a nonlinear mapping function $s(v_j)$ as shown.

$$y_j(t) = s(v_j(t)) = 1/(1 + \exp(-a * v_j(t))) \quad (4.4)$$

where the function $s(v)$ is assumed to have a sigmoidal nonlinear form.

For given inputs in time, we may also consider a desired response, $d_j(t)$, which represents the desired output that an output neuron in the network should yield at time t . Thus the difference between the desired response and the actual response of an output neuron constitutes the amount of error that the specific output neuron yields at time t . Thus we have:

$$E_j(t) = d_j(t) - y_j(t) \quad (4.5)$$

where $E_j(t)$ refers to the error at time t arising at output neuron j , $d_j(t)$ refers to the desired response, and $y_j(t)$ refers to the actual response of output neuron j . Moreover, the ensemble error at any time t is defined as the sum of the errors arising across all output neurons at time t . Thus:

$$\phi(t) = (1/2) * \sum_{j \in \mathcal{O}} [d_j(t) - y_j(t)]^2 \quad (4.6)$$

where $\phi(t)$ refers to the ensemble error across all output neurons at time t . Thus

$\phi(t)$ is obtained by summing the individual errors arising across all output neurons. Finally, we may define a total error surface which considers the ensemble error over all time t . Thus we have:

$$\xi_{total} = \int_{t=-\infty}^{t=+\infty} \phi(t)dt \Rightarrow \xi_{total} = (1/2) * \int_{t=-\infty}^{t=+\infty} \sum_{j \in O} [d_j(t) - y_j(t)]^2 \quad (4.7)$$

where ξ_{total} refers to the error surface of the neural network. The objective of the learning procedure is to minimize the total error surface, ξ_{total} , with respect to the free parameters of the system, i.e. the synaptic weights and delays, and also the parameters that characterize the impulse response of the LTI system itself. Thus the core of the backpropagation algorithm constitutes of how the error surface should be minimized, and how the free parameters of the network should be adapted.

4.1 Segmenting the training procedure in resonator neural networks

The convention in adapting connectionist neural networks is to adjust the synaptic weights of the system until a static input pattern yields a certain desired output. In our instance, we are considering a time-dependent neural network where the synapses are characterized by synaptic weights and transmission delays. Moreover, each individual neuron is defined by the resonate-and-fire (RAF) differential equations, which are in turn dependent on the damping and natural frequency of the neuron. Finally, since each neuron element is comprised of the RAF model and the sigmoidal nonlinearity, we may treat the parameter a in equation 4.4 as the final parameter in the learning procedure. Therefore, the training procedure in this instance is comprised of i) adapting the synaptic structure of the network by adjusting the synaptic weights and delays, and ii) adapting the parameters that define each individual perceptron model, i.e. adjusting the damping, natural frequency, and the slope of the sigmoidal nonlinearity.

The core of the backpropagation algorithm relies on gradient descent, and thus we must derive a gradient descent procedure to adapt each of the free parameters in our neural network. We first turn our attention to how we could adapt the synaptic structure of a resonator neural network.

4.1.1 Adapting RAF neural networks

In order to perform gradient descent, we must compute the partial derivative of the total error surface, ξ_{total} , with respect to the specific parameter that we wish to optimize. Thus in order to adapt the synaptic structure of a resonator neural network, we must compute the following terms:

i) $\frac{\partial \xi_{total}}{\partial \vec{w}_{ij}}$: to adapt the synaptic weights of the network;

ii) $\frac{\partial \xi_{total}}{\partial \vec{\tau}_{ij}}$: to adapt the time delays that characterizes each synaptic link

We may compute the terms above by applying the chain rule of differentiation, and thus we have:

$$\frac{\partial \xi_{total}}{\partial \vec{w}_{ij}} = \int_{t=-\infty}^{t=+\infty} \left(\frac{\partial \phi(t)}{\partial E_j(t)} \right) * \left(\frac{\partial E_j(t)}{\partial y_j(t)} \right) * \left(\frac{\partial y_j(t)}{\partial v_j(t)} \right) * \left(\frac{\partial v_j(t)}{\partial \vec{w}_{ij}} \right) \quad (4.8)$$

and also:

$$\frac{\partial \xi_{total}}{\partial \vec{\tau}_{ij}} = \int_{t=-\infty}^{t=+\infty} \left(\frac{\partial \phi(t)}{\partial E_j(t)} \right) * \left(\frac{\partial E_j(t)}{\partial y_j(t)} \right) * \left(\frac{\partial y_j(t)}{\partial v_j(t)} \right) * \left(\frac{\partial v_j(t)}{\partial x_i(t)} \right) * \left(\frac{\partial x_i(t)}{\partial t - \vec{\tau}_{ij}} \right) \quad (4.9)$$

Moreover, we may also perform gradient descent with respect to the free parameters that characterize each individual perceptron. Towards this end, we must compute the following:

i) $\frac{\partial \xi_{total}}{\partial \alpha_j}$: to adapt the damping factor of each neuron, where α_j refers to the damping factor of neuron j;

ii) $\frac{\partial \xi_{total}}{\partial \Omega_j}$: to adapt the natural frequency of each neuron, where Ω_j refers to the natural frequency of neuron j;

iii) $\frac{\partial \xi_{total}}{\partial a_j}$: to adapt the "slope" of the sigmoidal nonlinearity.

Similarly, we may again apply the chain rule of differentiation to obtain the following:

$$\frac{\partial \xi_{total}}{\partial \alpha_j} = \int_{t=-\infty}^{t=+\infty} \left(\frac{\partial \phi(t)}{\partial E_j(t)} \right) * \left(\frac{\partial E_j(t)}{\partial y_j(t)} \right) * \left(\frac{\partial y_j(t)}{\partial v_j(t)} \right) * \left(\frac{\partial v_j(t)}{\partial \alpha_j} \right) \quad (4.10)$$

and also:

$$\frac{\partial \xi_{total}}{\partial \Omega_j} = \int_{t=-\infty}^{t=+\infty} \left(\frac{\partial \phi(t)}{\partial E_j(t)} \right) * \left(\frac{\partial E_j(t)}{\partial y_j(t)} \right) * \left(\frac{\partial y_j(t)}{\partial v_j(t)} \right) * \left(\frac{\partial v_j(t)}{\partial \Omega_j} \right) \quad (4.11)$$

and finally:

$$\frac{\partial \xi_{total}}{\partial a_j} = \int_{t=-\infty}^{t=+\infty} \left(\frac{\partial \phi(t)}{\partial E_j(t)} \right) * \left(\frac{\partial E_j(t)}{\partial y_j(t)} \right) * \left(\frac{\partial y_j(t)}{\partial a_j} \right) \quad (4.12)$$

Thus the learning algorithm to adapt the free parameters of the neural network may be represented as follows:

$$\vec{w}_{ij}(k+1) = \vec{w}_{ij}(k) - \eta * \frac{\partial \xi_{total}}{\partial \vec{w}_{ij}} \quad (4.13)$$

$$\vec{\tau}_{ij}(k+1) = \vec{\tau}_{ij}(k) - \beta * \frac{\partial \xi_{total}}{\partial \vec{\tau}_{ij}} \quad (4.14)$$

adapts the synaptic structure of the neural network. In order to tune the free parameters of each individual neuron, we must implement the following:

$$\alpha_j(k+1) = \alpha_j(k) - \nu * \frac{\partial \xi_{total}}{\partial \alpha_j} \quad (4.15)$$

$$\Omega_j(k+1) = \Omega_j(k) - \rho * \frac{\partial \xi_{total}}{\partial \Omega_j} \quad (4.16)$$

$$a_j(k+1) = a_j(k) - \gamma * \frac{\partial \xi_{total}}{\partial a_j} \quad (4.17)$$

where η , β , ν , ρ , and γ refer to the momentum rate of the learning procedure, and they may be assigned a constant value between 0 and 1. Thus given a random initialization of the free parameters of the system, we may iterate through the learning algorithm presented in equations 4.13 through 4.17 in order to adapt the network to yield a certain desired response.

4.2 Discrete Time Implementation

We may proceed by approximating the convolution integral as a convolution sum. Thus we may replace the continuous time variable, t , with a discretized version,

where now $t = n \cdot \Delta t$. Furthermore, we may represent equation 4.3 as a combination of two distinct summations, where the first summation runs over the indices of synaptic weights (the spatial summation) and the second summation represents the discretized convolution integral (the temporal summation). Thus we have:

$$v_j(n) = \sum_{i=1}^p \sum_{l=0}^M ((h_j(l)) * (w_{ij}) * (x_i(n - \tau_{ij} - l))) - \theta_j \quad (4.18)$$

In equation 4.18, we have replaced the infinite limits of the convolution integral with a finite upper limit of M . The assumption that underlies this approximation is that our LTI system has finite memory, i.e. the impulse response $h_j(t) = 0$ for $t > T$ and $T = M \cdot \Delta t$. Also, the summation that represents the discretized convolution integral in equation 4.18 has a lower limit of $l = 0$, which implies that the LTI system is a causal system, i.e. $h_j(t) = 0$ for $t < 0$. Thus equation 4.18 has embedded within it a spatial summation which considers presynaptic spatial signals $i=1:p$, and a temporal summation that runs over $l=0:M$. Intuitively, equation 4.18 is considering two forms of synapses: a static spatial synapse, and a dynamic temporal synapse that is defined by the value of the impulse response of the system at discretized time l . Since an input signal is being weighted by a combination of the synaptic weights and the impulse response of the system, we may define an equivalent weight $w_{ij}(l)$ at time l , which constitutes the product of the synaptic weight and the value of the impulse response at time l , $h_j(l)$. Thus we have:

$$w_{ij}(l) = w_{ij} * (h_j(l)) \quad (4.19)$$

By substituting equation 4.19 in equation 4.18, we have:

$$v_j(n) = \sum_{i=1}^p \sum_{l=0}^M (w_{ij}(l) * x_i(n - \tau_{ij} - l)) - \theta_j \quad (4.20)$$

Our objective in implementing the temporal backpropagation algorithm is to minimize an error surface with respect to the free parameters of the system, i.e. the synaptic weights and the form of the impulse response that defines the LTI system, along with the delays of synaptic transmission. Thus we may discretize equation 4.5 to obtain the instantaneous error of an output neuron j :

$$E_j(n) = d_j(n) - y_j(n) \quad (4.21)$$

where $d_j(n)$ refers to the desired output from an output neuron j , and $y_j(n)$ refers to the actual output of neuron j , which is obtained from the discretized version of equation 4.4:

$$y_j(n) = s(v_j(n)) = 1/(1 + \exp(-a * v_j(n))) \tag{4.22}$$

Moreover, we may represent the discretized form of the total ensemble error across all output neurons at discrete time n as follows:

$$\phi(n) = (1/2) * \sum_{j \in O} [d_j(n) - y_j(n)]^2 \Rightarrow \phi(n) = (1/2) * \sum_{j \in O} (E_j(n))^2 \tag{4.23}$$

where index j refers to neurons in the output layer. The objective of the temporal backpropagation learning algorithm is to minimize an ensemble error surface over all time n . Thus the error surface we wish to minimize, ξ_{total} , is defined as:

$$\xi_{total} = \sum_n \phi(n) \Rightarrow \xi_{total} = (1/2) * \sum_n \sum_{j \in O} (E_j(n))^2 \tag{4.24}$$

Our objective is to minimize the error surface, ξ_{total} , with respect to the free parameters of the neural network. Since the method of gradient descent underlies the backpropagation algorithm, we must compute the partial derivative of the error surface, ξ_{total} with respect to the free parameter that we wish to optimize. Upon inspection of equations 4.8 through 4.11, it is evident the computations required for performing gradient descent are similar for all the parameters. Specifically, we must derive a *localgradient* for each neuron within the neural network, and we must backpropagate this gradient in order to adjust the free parameters.

4.3 Computation of the local gradient

The adjustment of the synaptic weights and delays of the network requires that we compute the following:

$$\frac{\partial \xi_{total}}{\partial \vec{x}_{ij}(n)} = \sum_n \left(\frac{\partial \xi_{total}}{\partial v_j(n)} \right) * \left(\frac{\partial v_j(n)}{\partial \vec{x}_{ij}(n)} \right) \tag{4.25}$$

where x_{ij} refers to either the weight or delay that exists on a synapse between input signal i , and neuron j . Similarly, the adaptation of the damping factor and natural frequency of each neuron requires that we compute the following:

$$\frac{\partial \xi_{total}}{\partial x_j(n)} = \sum_n \left(\frac{\partial \xi_{total}}{\partial v_j(n)} \right) * \left(\frac{\partial v_j(n)}{\partial x_j(n)} \right) \tag{4.26}$$

where x_j refers to the internal parameter of the neuron. Thus in both these computations, we must compute the local gradient of a neuron, $\delta_j(n)$, which is defined as:

$$\delta_j(n) = -\frac{\partial \xi_{total}}{\partial v_j(n)} \Rightarrow \delta_j(n) = -\frac{\partial \phi(n)}{\partial v_j(n)} \tag{4.27}$$

Thus the local gradient is obtained by computing the partial derivative of the instantaneous error, $\phi(n)$, with respect to the activity level of neuron j . Moreover, we know that the error of the network at time n is the sum of the errors across all output neurons j , and may be represented as:

$$\phi(n) = 1/2 * \left(\sum_{j \in \mathcal{O}} (E_j(n))^2 \right) \Rightarrow \phi(n) = 1/2 * \left(\sum_{j \in \mathcal{O}} (d_j(n) - y_j(n))^2 \right) \tag{4.28}$$

Thus the partial derivative of $\phi(n)$ with respect to $v_j(n)$ may be computed through the chain rule of differentiation:

$$\frac{\partial \phi(n)}{\partial v_j(n)} = \left(\frac{\partial \phi(n)}{\partial E_j(n)} \right) * \left(\frac{\partial E_j(n)}{\partial y_j(n)} \right) * \left(\frac{\partial y_j(n)}{\partial v_j(n)} \right) \tag{4.29}$$

In order to compute equation 4.29, we have the following:

$$\frac{\partial \phi(n)}{\partial E_j(n)} = E_j(n) \tag{4.30}$$

$$\frac{\partial E_j(n)}{\partial y_j(n)} = -1 \tag{4.31}$$

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \frac{\partial s(v_j(n))}{\partial v_j(n)} \tag{4.32}$$

$$\tag{4.33}$$

where $s(v)$ refers to the sigmoidal nonlinearity. The derivative of the sigmoidal function with respect to the membrane potential may be computed as follows:

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \Psi'(v_j(n)) = a * y_j(n) * (1 - y_j(n)) \tag{4.34}$$

where $\Psi(v_j(n)) = 1/(1 + \exp(-a * v_j(n)))$.

Therefore, the local gradient arising at an output node of the neural network is:

$$\delta_j(n) = E_j(n) * \Psi'(v_j(n)) \tag{4.35}$$

Thus for the case where neuron j is an output node, the computation of the local gradient is relatively straightforward. However, for the case where the neuron in question is a hidden neuron, we must compute the error gradient arising at the output layer of the network, and backpropagate this gradient to the presynaptic hidden layers of the network. Thus let us define a set σ , which refers to the set of all neurons whose inputs are fed by the hidden neuron i in the forward pass. Thus we may define a local gradient at layer i at time n as follows:

$$\delta_i(n) = -\frac{\partial \xi_{total}}{\partial v_i(n)} \tag{4.36}$$

$$\delta_i(n) = -\sum_{j \in \sigma} \sum_n \left(\frac{\partial \xi_{total}}{\partial v_j(n)} \right) * \left(\frac{\partial v_j(n)}{\partial v_i(n)} \right) \tag{4.37}$$

$$\tag{4.38}$$

We may further define the term $-\frac{\partial \xi_{total}}{\partial v_j(n)}$ as the local gradient arising in neuron j of the postsynaptic layer. Thus we have:

$$\delta_j(n) = -\frac{\partial \xi_{total}}{\partial v_j(n)} \tag{4.39}$$

Thus, the local gradient arising at a hidden neuron i at time index n may be represented as:

$$\delta_i(n) = \sum_{j \in \sigma} \sum_n \delta_j(n) * \left(\frac{\partial v_j(n)}{\partial y_i(n)} \right) * \left(\frac{\partial y_i(n)}{\partial v_i(n)} \right) \tag{4.40}$$

From equation 4.40 we know that the term $\frac{\partial y_i(n)}{\partial v_i(n)}$ is the derivative of our sigmoidal nonlinearity function. Furthermore, we also know the local gradient arising at a postsynaptic neuron j . However, we need to compute the partial derivative $\frac{\partial v_j(n)}{\partial y_i(n)}$, which is the derivative of the activity level of a postsynaptic neuron with respect to the output of the hidden neuron i . Thus $v_j(n)$ refers to the membrane potential of postsynaptic neuron j , and it is determined by the outputs of the presynaptic neurons that it receives input signals from. Thus we have:

$$v_j(n) = \sum_{j=0}^p \sum_{l=0}^M w_{ij}(l) * y_i(n - \tau_{ij} - l) \tag{4.41}$$

where the summation from $j=0:p$ refers to the spatial summation, and the summation from $l=0:M$ refers to the temporal summation. Also, τ_{ij} refers to the time delay that exists on synapse ij . Since convolution is a commutative operation, and because we wish to compute the term $\frac{\partial v_j(n)}{\partial y_i(n)}$, we may rearrange equation 4.41 as follows:

$$v_j(n) = \sum_{j=0}^p \sum_{l=0}^M y_i(l) * w_{ij}(n - \tau_{ij} - l) \tag{4.42}$$

Subsequently, we may compute the partial derivative of $v_j(n)$ with respect to the output of a hidden neuron, $y_i(n)$ as follows:

$$\begin{aligned} \frac{\partial v_j(n)}{\partial y_i(n)} &= w_{ij}(n - \tau_{ij} - l) \quad \text{if } 0 \leq n - \tau_{ij} - l \leq M \\ &= 0 \quad \text{otherwise} \end{aligned} \tag{4.43}$$

Thus if time index n is outside the range $l + \tau_{ij} \leq n \leq M + l + \tau_{ij}$, the partial derivative $\frac{\partial v_j(n)}{\partial y_i(n)}$ evaluates to zero. Therefore, for a hidden neuron i , the local gradient we need to compute is:

$$\delta_i(n) = \Psi'(v_i(n)) \sum_{j \in \sigma} \sum_{n=l+\tau_{ij}}^{M+l+\tau_{ij}} \delta_j(n) * w_{ij}(n - \tau_{ij} - l) = \Psi'(v_i(n)) * \sum_{j \in \sigma} \sum_{n=0}^M \delta_j(n + \tau_{ij} + l) * w_{ij}(n) \tag{4.44}$$

Upon inspecting equation 4.44, we notice that it violates causality, because we require future values of the local gradients arising from postsynaptic neurons. We may circumvent this problem by considering a delayed form of the local gradient originating at a hidden node by storing the following vector:

$$\overrightarrow{\Delta_j(n - M)} = [\delta_j(n - M), \delta_j(n - M + 1), \dots, \delta_j(n)] \quad (4.45)$$

Thus the computation of the local gradient arising at a postsynaptic neuron may be effectively made causal by considering previous values of the gradient. Therefore, we may reformulate equation ?? as follows:

$$\delta_i(n - lM) = \Psi'(v_i(n - lM)) * \sum_{j \in \sigma} \overrightarrow{\Delta_j^T(n - lM)} * \vec{w}_{ij} \quad (4.46)$$

where M is the memory of the LTI system, and index l identifies the hidden layer in question. Specifically, l=1 corresponds to the presynaptic layer of neurons that provide inputs to the output layer, while l=2 refers to two layers back from the output layer.

4.3.1 Adapting the synaptic weights

If a neuron j is in the output layer, the weight update algorithm is implemented as follows:

$$\overrightarrow{w_{ij}(k + 1)} = w_{ij}(k) + \eta * \delta_j * x_i(n) \quad (4.47)$$

$$\delta_j(n) = E_j(n) * \Psi'(v_j n) \quad (4.48)$$

Moreover if a neuron i is in a hidden layer, the weight update algorithm for the input synaptic weights, w_{hi} , is:

$$\overrightarrow{w_{hi}(k + 1)} = w_{hi}(k) + \eta * \delta_i(n - lM) * \overrightarrow{x_h(n - lM)} \quad (4.49)$$

$$\delta_i(n - lM) = \Psi'(v_i(n - lM)) * \sum_{j \in \sigma} \overrightarrow{\Delta_j^T(n - lM)} * \vec{w}_{ij} \quad (4.50)$$

where M is the memory of the LTI system, and index l identifies the hidden layer in question.

4.3.2 Adapting the synaptic delays

In order to adapt the synaptic delays that characterize each synapse, we may perform a procedure that is similar to the weight update algorithm. Therefore, if a neuron j is in the output layer, the adaptation algorithm for the synaptic delays that exist on its input synapses is:

$$\overrightarrow{\tau_{ij}(k+1)} = \overrightarrow{\tau_{ij}(k)} + \beta * \delta_j * w_{ij}(n) * \frac{\partial x_i(\vec{t})}{\partial (t - \overrightarrow{\tau_{ij}})} \quad (4.51)$$

$$\delta_j(n) = E_j(n) * \Psi'(v_j n) \quad (4.52)$$

Moreover if a neuron i is in a hidden layer, the update algorithm for the input synaptic delays, $\overrightarrow{\tau_{hi}}$, is:

$$\overrightarrow{\tau_{hi}(k+1)} = \overrightarrow{\tau_{hi}(k)} + \beta * \delta_i(n - lM) * w_{hi}(n - lM) * \frac{\partial x_i(\vec{t})}{\partial (t - \overrightarrow{\tau_{ij}})} \quad (4.53)$$

$$\delta_i(n - lM) = \Psi'(v_i(n - lM)) * \sum_{j \in \sigma} \overrightarrow{\Delta_j^T(n - lM)} * \overrightarrow{w_{ij}} \quad (4.54)$$

where M is the memory of the LTI system, and index l identifies the hidden layer in question.

4.3.3 Adapting the Damping and Natural Frequency of each neuron

Let us consider the discretized form of the internal activity level of a neuron represented in equation 4.3:

$$v_j(n) = \sum_{i=1}^p \sum_{l=0}^M ((h_j(l)) * (w_{ij}) * (x_i(n - \tau_{ij} - l))) - \theta_j \quad (4.55)$$

For the specific case of a resonator neuron, we know that the form of the impulse response will be:

$$h_j(l) = \exp(-\alpha_j * l) * \cos(\Omega_j * l + \varphi) \quad (4.56)$$

which represents an exponential sinusoid. Moreover, we know that if we take the partial derivative of $v_j(n)$ with respect to either parameter α or Ω , we will obtain $v_j(n)$ multiplied by some constant. Therefore, if we were to adapt the internal parameters of a resonator neuron, i.e. the damping factor and the natural frequency, we would have to consider the internal activity level of the neuron in question.

Thus let us consider the scenario where we adapt the damping factor of each neuron, α_j . If the neuron is an output neuron, we may implement the following learning algorithm:

$$\alpha_j(k+1) = \alpha_j(k) - \nu * \delta_j * n * v_j(n) \quad (4.57)$$

$$\delta_j(n) = E_j(n) * \Psi'(v_j n) \quad (4.58)$$

where the index n runs over the time indices of v_j . Moreover, if the neuron i is a hidden neuron, which receives input signals from presynaptic neurons h , and transmits signals to postsynaptic neurons j , we have:

$$\alpha_j(k+1) = \alpha_j(k) - \nu * \delta_i(n-lM) * (n-lM) * v_j(n-lM) \quad (4.59)$$

$$\delta_i(n-lM) = \Psi'(v_i(n-lM)) * \sum_{j \in \sigma} \overrightarrow{\Delta_j^T(n-lM)} * w_{ij}^{\rightarrow} \quad (4.60)$$

where M is the memory of the LTI system, and index l identifies the hidden layer in question.

4.3.4 Adapting the slope of the sigmoidal nonlinearity

In order to adjust the shape of the sigmoidal nonlinearity, we must consider the form of the logistic function, $\Psi(a * v)$:

$$s(a * v) = \Psi(a * v) = 1 / (1 + \exp(-a_j * v_j)) \quad (4.61)$$

where a_j is the parameter that we wish to adapt. Therefore, we must compute the following:

$$\frac{\partial \xi_{total}}{\partial a_j(n)} = \sum_n \left(\frac{\partial \phi(n)}{\partial E_j(n)} \right) * \left(\frac{\partial E_j(n)}{\partial y_j(n)} \right) * \left(\frac{\partial y_j(n)}{\partial a_j(n)} \right) \tag{4.62}$$

If we compute the above derivatives, we have:

$$\begin{aligned} \frac{\partial \phi(n)}{\partial E_j(n)} &= E_j(n) \\ \frac{\partial E_j(n)}{\partial y_j(n)} &= -1 \\ \frac{\partial y_j(n)}{\partial a_j(n)} &= v_j * y_j * (1 - y_j) \end{aligned} \tag{4.63}$$

Therefore, we may define a local sigmoidal gradient, $\delta_{j,sig}$, arising at each output neuron j as:

$$\delta_{j,sig}(n) = -\frac{\partial \xi_{total}}{\partial y_j(n)} \Rightarrow \delta_{j,sig}(n) = E_j(n) \tag{4.64}$$

Thus for an output neuron, we may easily compute the local sigmoidal gradient, $\delta_{j,sig}(n)$, since this is just the error arising at the output neuron j . However, for a hidden neuron i which is located in a hidden layer, we must compute a gradient at the output layer, and backpropagate this gradient against the direction of synaptic communication.

Therefore, if the neuron j is an output neuron, the learning algorithm to update the parameter a in the logistic function is:

$$a_j(k + 1) = a_j + \gamma * \delta_{j,sig}(n) * v_j(n) * y_j(n) * (1 - y_j(n)) \tag{4.65}$$

$$\delta_{j,sig}(n) = E_j(n) \tag{4.66}$$

Moreover, if we are adapting the logistic function of a hidden neuron i , then the adaptation algorithm for the parameter a is:

$$a_j(k+1) = a_j + \gamma * \delta_i(n-lM) * v_j(n-lM) * y_j(n-lM) * (1-y_j(n-lM)) \quad (4.67)$$

$$\delta_i(n-lM) = \Psi'(v_i(n-lM)) * \sum_{j \in \sigma} \overrightarrow{\Delta_j^T(n-lM)} * \vec{w}_{ij} \quad (4.68)$$

where M is the memory of the LTI system, index l identifies the hidden layer in question, and $\Psi'(v_i(n-lM))$ refers to the partial derivative of the logistic function with respect to the internal activity level of the hidden neuron.

4.4 Practical Considerations and Implementation in MATLAB

As is evident through the derivation presented in this chapter, we may adapt the synaptic structure of a neural network comprised of RAF neurons by optimizing the synaptic weights and delays of the network. Moreover, we may also adapt the internal parameters that define each RAF neuron by adjusting the damping factor, the natural frequency, and the shape of the sigmoidal nonlinearity for each neuron in the network. Thus the temporal backpropagation learning procedure for resonator neural networks incorporates: i) the connectionist approach to learning by adapting the synaptic structure of the network, and, ii) the cellular notion of learning, since the algorithm adapts the internal parameters that define each RAF neuron. We explore the implications of such a learning procedure in the subsequent concluding chapter.

An approach to learning that considers the synaptic structure as well as neuronal internal parameters is a novel idea in neural networks, and thus there exist a variety of practical considerations regarding the implementation of the temporal backpropagation algorithm. For instance, the learning procedure may be implemented in a synchronous method, where each parameter is updated simultaneously on each learning iteration. Alternatively, it may be desirable to adapt each parameter asynchronously, and on different time scales.

Also, the algorithm may be implemented in a myriad of software environments. In our instance, we utilize the Biological Neural Network (BNN) Toolbox since the toolbox contains many in-built theoretical neuron models. However the toolbox does not contain the temporal backpropagation algorithm. Thus the derivation presented in this chapter could provide the theoretical basis for the backpropagation algorithm

in the BNN toolbox. In appendix D we have attached program code that implements a resonator neural network within the BNN toolbox; however the adaptation procedure needs to be implemented.

In order to implement a simple form of the temporal backpropagation learning algorithm, we have attached an example of a gene translation problem in appendix D, entitled function $y_k = \text{translate}(x)$. This simple example considers how a time dependent neural network may be implemented and adapted to recognise binary words in time. Thus $x[n]$ specifies a 6 bit binary word, where each of the bits may assume a value of -1 or +1. We assume that each bit codes for a nucleotide base in the genetic code. The objective of the network is to essentially translate $x[n]$, which represents a codon, to a corresponding amino acid represented by the temporal code y_k . A desired temporal pattern $D[n]$ is implicitly defined for each input pattern $x[n]$.

The basis of the algorithm is to assume that each neuron is defined by a certain impulse response, and thus the activity level of the neuron at any time is determined by the convolution of the net input to the neuron, with the impulse response at the specific time of interest. Thus we may define vectors of temporal weights, which are constituted of the product of the input weights to a neuron, with the impulse response of that particular neuron. The temporal backpropagation algorithm performs gradient descent on each of the temporal weight vectors, in order to minimize the ensemble error over the entire training time. In the following chapter, we further explore the implications of the temporal backpropagation algorithm, and we turn our attention towards possible future directions of research that could be pursued with the resonate-and-fire neuron.

Chapter 5

Conclusions and Future Directions

Through this thesis we have explored the resonate-and-fire (RAF) neuron, and how this specific neuron model may be implemented within neural networks. Biophysical neural models along with the methods of phase space analysis gave rise to the distinction between integrator and resonator neurons. The solutions to the differential equation that defines the RAF neuron model display either the characteristics of integrator neurons, or that of resonator neurons, depending on the internal parameters that define the RAF neuron. Moreover, in the previous chapter we explored how a neural network comprised of RAF neurons could be adapted, and made to learn to perform certain desired tasks, through the temporal backpropagation algorithm. Thus while the resonate-and-fire neuron was originally inspired from the insights gained through biophysical neural modeling, the implementation of the resonate-and-fire neuron inherently belongs to the realm of neural networks. Naturally, possible future projects could consider either: i) practical connectionist applications of how resonator neural networks could achieve engineering specifications, or ii) the physiological implications of biological neuronal networks being comprised of resonator and integrator neurons. Therefore through this chapter we explore each of these possible future research directions in turn.

5.1 Resonator Neural Networks and Engineering Design

One of the original ideas that inspired the resonate-and-fire neuron was the question of how to incorporate internal dynamics into a neuron model. The obvious constraint was that the RAF neuron must also have explicit solutions, so that a network of RAF neurons may be implemented, and the resulting neural network be adaptable. The RAF neuron is thus defined by a 2^{nd} order linear differential equation, whose solutions have the form of an exponentially decaying sinusoid. Since the solutions to the RAF model are time dependent solutions, we may represent time explicitly within the neural network, and we may also incorporate time in the learning procedure, as we showed in the previous chapter.

Therefore neural networks that are comprised of RAF neurons could be adept at classifying time dependent input patterns, and at solving time dependent problems. We showed a simple example of how a single RAF neuron could classify time dependent input patterns through our solution to the XOR Problem in chapter 3.5. One of the insights that we gained from our solution was that the neuron's classification is dependent on two factors: i) the strength of the input signal it receives, determined by the synaptic weights, and ii) the internal parameters that define the dynamics of the neuron model. Thus the internal parameters of the neuron govern the characteristics of the evolution of the membrane potential, while the static synaptic weight defines the magnitude of the perturbation. Also, we concluded that in the underdamped limit, the RAF neuron's dynamics is largely dependent on the timing of input stimuli. Therefore RAF neural networks could also be adept at classifying input patterns that have different temporal structures.

The idea that a neuron's response is dependent on its internal parameters as well as the static synaptic weights is a novel idea in neural networks, and it raises the question of what connectionism should really entail. In traditional neural networks, the memory of a network is distributed in the synaptic structure of the network. On the other hand, in resonator neural networks, the notion of "memory" includes synaptic weights and delays, as well as a neuron's internal parameters. Moreover, the learning procedure in resonator neural networks is also a combination of adapting the synaptic structure of the network, as well as a neuron's internal parameters. Thus one of the ideas raised in this thesis that could be pursued further regards the nature of the learning procedure itself, i.e., what should be the adaptation mechanism in a neural network, and to what extent should it involve the internal parameters of individual

neurons? We speculate that the adaptation of neuronal internal parameters could enable a neural network to learn more complex decision boundaries than possible in traditional neural networks.

The adaptation of RAF neural networks could also have potential applications in multi-variable control problems, where it is necessary to maintain desired levels of numerous variables of interest. Moreover, since we may represent time explicitly in our RAF model, a neural network comprised of RAF neurons could potentially ensure that any given variable is of a desired value, at a specific instant of time. For instance, RAF networks could be utilized to maintain desired levels of multiple variables in the design of, for instance, aeroplane controllers or surgical equipment. Multi-variable control problems are ubiquitous in complex systems, and they are an inevitable aspect of engineering as well as biology. Thus we subsequently focus our attention on how the findings of this thesis could potentially provide novel research directions in neurobiology.

5.2 The RAF Neuron and Neurobiology

A central question in contemporary biology regards the relationship between structure and function in living systems. Thus in the context of nervous systems, how are the structure of neuronal networks (i.e. the topology of connections, the synaptic structure, etc.) related to the functions that these networks ultimately perform? This particular question of structure and function could also be asked for individual nerve cells: how are the molecular characteristics of nerve cells, along with the dendritic tree that serves as its input related to the functions that the cell is involved in mediating? In the context of neural networks comprised of RAF neurons, "structure" may be thought of as the synaptic structure of the network, along with the internal parameters that define a neuron. On the other hand, "function" relates to the task that we wish our resonator network, or individual RAF neuron, to perform. Thus the question that arises is whether certain structures are particularly adept at performing specific desired functions. Moreover, the nature of these structure-function relationships could provide insights into how the interplay between the structure of an individual neuron, and the function(s) that the neuron mediates, manifests itself in nervous systems.

A particular instance of the interplay between structure and function is found in the context of *neural plasticity*. The term plasticity refers to the degree to which the nervous system is capable of exhibiting change and adapting itself; thus the process

of learning in nervous systems is a specific instance of neural plasticity. Contemporary research in neurobiology suggests that the nervous system possesses an immense ability to exhibit plasticity throughout development, and we are only beginning to elucidate the neural mechanisms that underlie the learning process. However, what is certain is that the biological learning process is as much a connectionist paradigm (i.e. the adaptation of synaptic structures), as it is a cellular mechanism (i.e. the adjustment of cellular morphological and physiological characteristics).

One of the notions that this thesis explores is how the learning procedure in neural networks could incorporate the adjustment of synaptic structures, as well as the internal parameters of a neuron. Thus the learning procedure in resonator neural networks involves an interplay between the adaptation of synaptic structures and neuronal internal parameters. The question that arises is whether such an interplay between the adaptation of cellular and synaptic structures also exists within biological nervous systems. Moreover, there exists a certain limitation (biological and in engineering) to the extent to which we could optimize synaptic structures and neuronal parameters, and there exists an inherent limit to the extent that error can be minimized in a neural network. The question that arises is whether such a limitation to optimization also exists in biological nervous systems, and the physiological implications of such constraints inherent in the process of learning.

A specific characteristic of the resonate-and-fire neuron is the dependence of its activity level on the timing of input stimuli. Thus one of the capabilities of the RAF neuron is that it may be utilized as a switching device, which transmits information to post synaptic neurons only if input patterns are applied at specific times. The question that arises is how information traverses through nervous systems, and how integrator and resonator neurons are involved in controlling the flow of information. Thus do there exist certain characteristic *network motifs* of integrators and resonators in the nervous system that regulate the flow of information? A network motif refers to a characteristic pattern of interactions that occurs ubiquitously within a network. These motifs are usually detected as patterns that occur significantly more often than would be expected in a randomized network [41].

The notion of motifs in neural networks could be interpreted in several ways. For instance, a motif could refer to a specific neural pathway that is activated when a certain input pattern is applied to the network. Thus the activation of an entire set of interconnected neurons could represent a specific external stimulus within the network. Due to the time dependence of the RAF neuron, distinct temporal patterns of input stimuli would activate different pathways within the network. The specific pathways that are activated could represent the network's *memory* regarding differ-

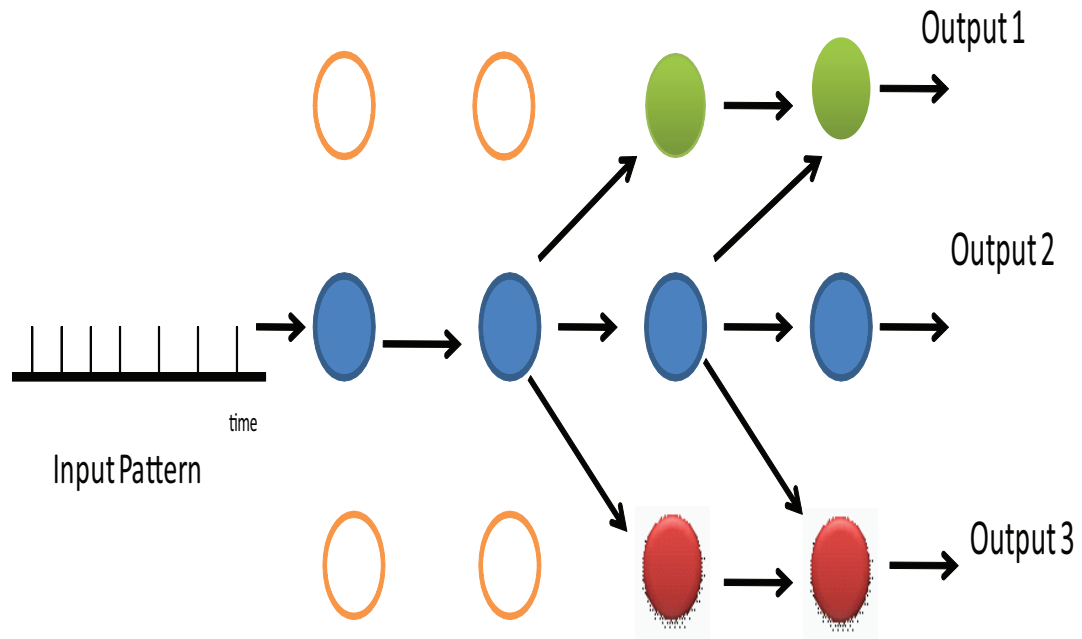


Figure 5.1: The activation of neuronal pathways and the representation of information. An input pattern with a certain temporal pattern activates one neuron in the input layer, which transmits to its postsynaptic partner. Information may flow through any of the pathways shown before activating an output neuron. Thus the pathway of information flow that elicited an output could have functional significance.

ent input patterns. The learning procedure in such networks would consequently need to define desired responses for sets of neurons that are within common neuronal pathways, instead of just output neurons. Figure 5.1 depicts such a framework of information representation. It could be interesting to study naturally occurring network motifs in nervous systems, as well as motifs that arise as a result of the learning procedure in neural networks.

Appendix A

Derivation of the Resonate-and-Fire Neuron

Neural signaling represents an important field in mathematical biology, and the theory of nerve action potential generation provides vital insights into the generic dynamics of biological oscillators. One of the first experimental studies that proposed a theoretical framework for the mechanism of action potential generation was Hodgkin and Huxley's (HH) Nobel Prize winning work on the giant squid axon [21]. The HH model provides an instance of a simple biophysical model that relates the dynamics of the neural membrane potential to the dynamical gating parameters that regulate the flow of ions through the neural membrane. Due to the inherent complexity of the HH model, various simpler phenomenological models have been proposed that capture the key aspects of the full system. One of the earliest and most insightful of these is the Fitzugh-Nagumo (FHN) model [23], [42], [43], [24], which considers a reduced system of the HH model. Through this appendix we begin with the Hodgkin-Huxely model, and subsequently consider theoretical models which represent simplifications of the HH system. The insights that we obtain from these classic models are utilized to derive the differential equation that govern the dynamics of the resonate-and-fire neuron.

A.1 The Hodgkin-Huxley model

Let us assume that the positive direction of current is denoted by positive charge leaving the nerve cell. The current $I(t)$ is comprised of individual ionic currents that pass through the membrane, I_i , and the time variation of the transmembrane potential, $V(t)$, i.e. the contribution due to the membrane capacitance. Thus we have:

$$I(t) = C * \frac{dV}{dt} + I_i \quad (\text{A.1})$$

where C is the membrane capacitance, V is the transmembrane potential, and I_i is the current due to the flow of ions across the neural membrane. Based on experimental and theoretical studies, Hodgkin and Huxley assumed the following convention:

$$\begin{aligned} I_i &= I_{Na} + I_K + I_L \\ \Rightarrow I_i &= g_{Na}m^3h(V - V_{Na}) + g_Kn^4(V - V_K) + g_L(V - V_L) \end{aligned} \quad (\text{A.2})$$

where V is the membrane potential, and I_{Na} is the sodium current, I_K is the potassium current, and I_L is a leakage current that is comprised of all other ions which contribute to the ionic current I_i . Moreover, V_{Na} , V_K , and V_L refer to constant Nernst equilibrium potentials. Also, g refers to a constant conductance, and the variables m , n , and h represent ionic gating variables that are bounded between 0 and 1. Thus the term $g_{Na}m^3h$ in equation A.2 represents the sodium conductance, and it is a measure of the proportion of sodium channels that are "open" at a given membrane potential. Thus the conductances of the sodium and potassium ionic channels are functions of the membrane potential. The gating variables in the Hodgkin-Huxley system are governed by the following differential equations:

$$\begin{aligned} \frac{dm}{dt} &= \alpha_m(V) * (1 - m) - \beta_m(V) * m \\ \frac{dn}{dt} &= \alpha_n(V) * (1 - n) - \beta_n(V) * n \\ \frac{dh}{dt} &= \alpha_h(V) * (1 - h) - \beta_h(V) * h \end{aligned} \quad (\text{A.3})$$

where α and β are functions of V , and were empirically determined to be:

$$\begin{aligned}
\alpha_n(V) &= 0.01 * \frac{10 - V}{\exp(\frac{10-V}{10}) - 1} \\
\beta_n(V) &= 0.125 * \exp(\frac{-V}{80}) \\
\alpha_m(V) &= 0.1 * \frac{25 - V}{\exp(\frac{25-V}{10}) - 1} \\
\beta_m(V) &= 4 * \exp(-V/18) \\
\alpha_h(V) &= 0.07 * \exp(-V/20) \\
\beta_h(V) &= \frac{1}{\exp(\frac{30-V}{10}) + 1}
\end{aligned} \tag{A.4}$$

Therefore, if an external current stimulus $I_{stim}(t)$ is applied, equation A.1 becomes:

$$C * \frac{dV}{dt} = -g_{Na}m^3h(V - V_{Na}) - g_Kn^4(V - V_K) - g_L(V - V_L) + I_{stim} \tag{A.5}$$

The system of equations A.2 to A.3 constitute the 4-variable Hodgkin-Huxley model.

One of the crucial insights provided by the Hodgkin-Huxley model regards the generation and transmission of action potentials. The variable m in equations A.5 and A.3 represents sodium channel gating, and is responsible for the generation of the action potential. Typically when the membrane potential reaches some critical threshold value, thousands of sodium channels are activated resulting in the rapid influx of sodium ions into the nerve cell. The influx of positive charge significantly depolarizes the membrane potential, resulting in the upstroke of the action potential. Subsequently, potassium channels are also sensitive to deviations of membrane potential, and thus the action potential activates potassium channels as well. In the HH model, the gating variable n governs the dynamics of potassium channels. The opening of potassium channels result in potassium ions leaving the nerve cell, and since in this instance positive charge leaves the cell, the membrane potential is repolarized. Moreover, the activation of sodium channels is followed by the inactivation of the same channels through the gating variable h in equations A.3 and A.5.

The time scales for m , h , and n in equation A.3 are significantly separated in that the time scale that defines the evolution of variable m is much faster than the repolarization variables h and n . Thus the convention is to assume that it is sufficiently fast that it relaxes immediately to its steady state value, determined by setting

$\frac{dm}{dt} = 0$. Furthermore, since the variable h evolves at a much slower rate than m and n , we may also set h to be some constant h_o . Through invoking these simplifying assumptions based on the separation of time scales, we may reduce the 4-dimensional Hodgkin-Huxley model into a 2-dimensional dynamical model which retains many of the qualitative features of the Hodgkin-Huxley system. Thus we next turn our attention to the FitzHugh-Nagumo model, which represents such a simplification to the original Hodgkin-Huxley equations.

A.2 The FitzHugh-Nagumo Model

The core idea that underlies the FitzHugh-Nagumo (FHN) model is that we may take projections of the four dimensional HH model onto a two dimensional phase space, and essentially reduce the Hodgkin-Huxley equations into a fast-slow system of two differential equations. Thus the FitzHugh-Nagumo model is a 2-variable model in V and w , and is represented by the following differential equations:

$$\begin{aligned} F(v, w) &= \frac{dv}{dt} = v(a - v)(v - 1) - w + I_{stim} \\ G(v, w) &= \frac{dw}{dt} = b * v - \gamma * w \end{aligned} \quad (\text{A.6})$$

where $0 < a < 1$, and b and γ are positive constants. Thus the variable v qualitatively mimics the evolution of the membrane potential, and the *recovery variable* w represents a reduction of the (m, n, h) system into a single variable. The phase portrait and the nullclines of the FitzHugh-Nagumo Model are depicted in figure A.1.

The nullclines of the FHN model may be determined by considering $\frac{dv}{dt} = 0 = \frac{dw}{dt}$, which results in the following relations in the (v, w) phase space:

$$\begin{aligned} \frac{dv}{dt} = 0 &\Rightarrow w = -v^3 + (1 + a) * v^2 - a * v + I_{stim} \\ \frac{dw}{dt} = 0 &\Rightarrow w = (b/\gamma) * v \end{aligned} \quad (\text{A.7})$$

Thus the equilibria of the FHN model are determined by the intersection of the v and w nullclines in the phase space. Since the v nullcline is a cubic nullcline, the

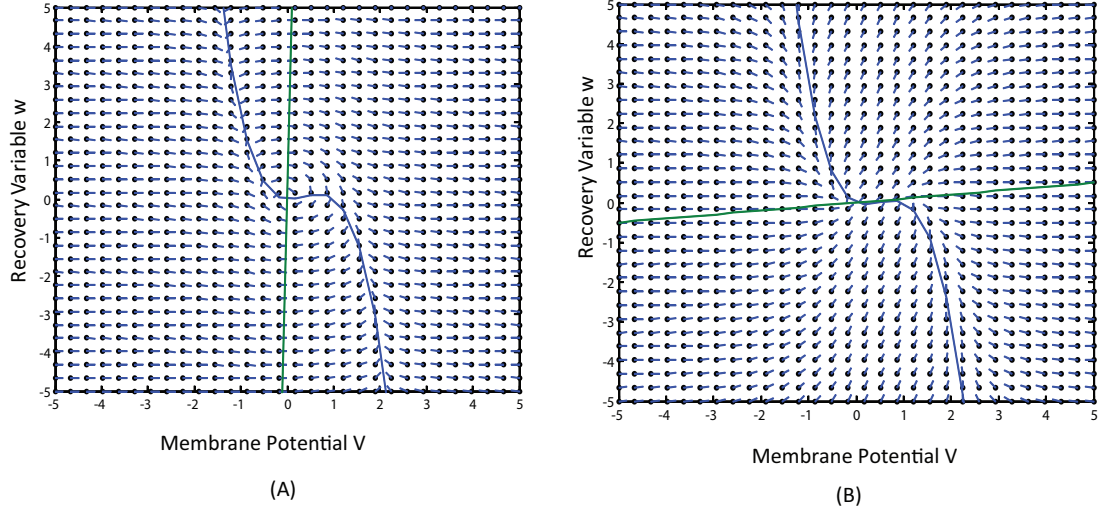


Figure A.1: Nullclines and Phase Portrait of The FitzHugh-Nagumo Model. (A) The characteristic nullclines and phase portrait of resonator neurons. (B) A representative phase portrait of integrator neurons.

FitzHugh-Nagumo model may have upto three equilibria points. If we set $I_{stim} = 0$ in equation A.7, we are guaranteed that an equilibrium point exists at $(v^*, w^*) = (0, 0)$. Moreover, we know that the topological neighbourhood of this equilibrium point may be determined by linearizing around the equilibrium, and considering the Jacobian matrix.

Thus the Jacobian Matrix is:

$$\mathbf{A} = \begin{bmatrix} F_v & F_w \\ G_v & G_w \end{bmatrix}$$

where from equation A.6 $F_v = \frac{dF}{dv}$, $F_w = \frac{dF}{dw}$, $G_v = \frac{dG}{dv}$, and $G_w = \frac{dG}{dw}$. Thus upon evaluating the Jacobian matrix at the equilibrium point $(0, 0)$, we obtain:

$$\mathbf{A} = \begin{bmatrix} -a & -1 \\ b & -\gamma \end{bmatrix}$$

The characteristic roots of the linearization are thus:

$$s_1 = (1/2) * (-a - \gamma + ((a + \gamma)^2 - 4 * a * \gamma - 4 * b)^{1/2})$$

$$s_2 = (1/2) * (-a - \gamma - ((a + \gamma)^2 - 4 * a * \gamma - 4 * b)^{1/2}) \quad (\text{A.8})$$

The case where the characteristic roots around the equilibrium point are distinct and negative correspond to the case where the neuron is an *integrator neuron*. On the other hand, if the characteristic roots are complex conjugates, then the neuron is referred to as a *resonator neuron*. Depicted in figure A.2 are representative phase portraits and trajectories in the phase space of integrator and resonator neurons, along with the evolution of the same states in time. The core idea behind the resonate-and-fire neuron is that we may incorporate the two major modes of neuronal dynamics, i.e. of integrators and resonators, within a single neuron model by selecting appropriate values for the damping and natural frequency parameters.

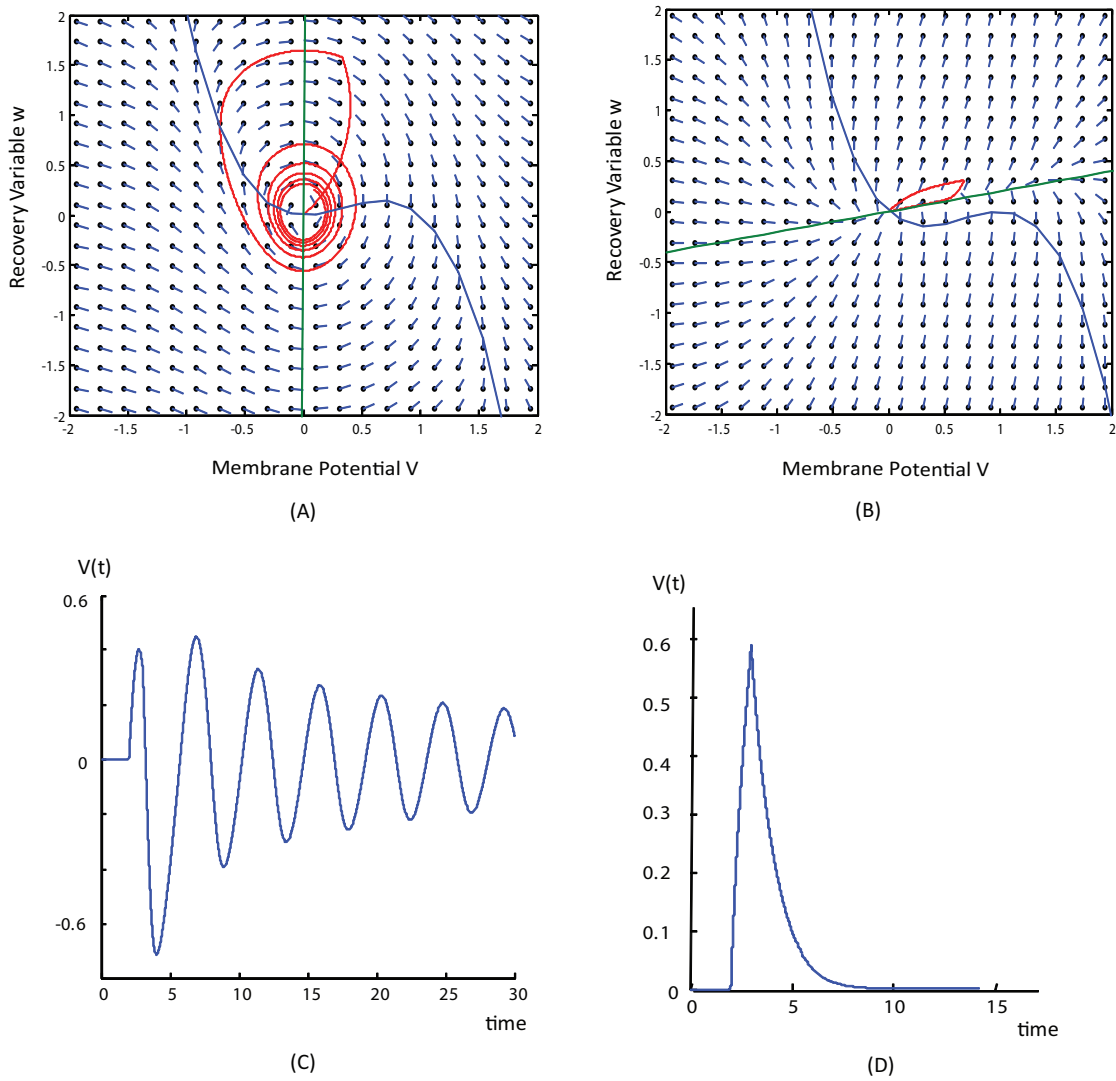


Figure A.2: Phase space trajectories of integrators and resonators in the FitzHugh-Nagumo model. (A) Representative phase space trajectory of a resonator neuron, in response to an excitatory stimulus applied while the neuron is at rest (B) State space trajectory around the equilibrium resting state for an integrator neuron in response to an excitatory stimulus. (C) Time domain representation of evolution of membrane potential for resonator neuron shown in panel (A). The subthreshold oscillatory behaviour is a signature profile of resonator neurons. (D) Evolution of membrane potential in time for integrator neuron. The monotonic evolution of the membrane potential is characteristic of integrator neurons.

Appendix B

Derivation of The Static Backpropagation Algorithm

Through this appendix section, we present the derivation of the static backpropagation algorithm. In this instance *static* refers to the nature of the inputs and outputs, i.e. the objective of the learning procedure is to map a static input vector, \vec{x} , to a desired static output. This is in contrast to the temporal backpropagation algorithm, which considers time varying inputs and time varying outputs. However, as will become evident, the temporal backpropagation algorithm is an extension of the static implementation. The backpropagation algorithm is essentially a paradigm in error minimization, and the mathematical derivation first appeared in the doctoral thesis of Werbos in [14]. The algorithm was adapted for utilization in the field of neural networks in [15], and error backpropagation has become one of the central supervised learning paradigms in neural networks.

Given a neural network, we may derive an error signal that exists at each output node of the network. Thus for each of the output neurons in the network, consider an error signal:

$$e_j(n) = d_j(n) - y_j(n) \tag{B.1}$$

where neuron j is an output node, $d_j(n)$ denotes the desired output, and $y_j(n)$ denotes the actual output. The iteration index n refers to the presentation of the n th training pattern. The output $y_j(n)$ is defined as follows:

$$y_j(n) = s(v_j) = 1/(1 + \exp(-a * v_j(n))) \quad (\text{B.2})$$

where the function $s(v)$ is assumed to be the logistic function, and $v_j(n)$ refers to the net internal activity level of neuron j at iteration n . We know that the internal activity level is the product of synaptic weights and input signals, and thus:

$$v_j(n) = \sum_{i=0}^p (w_{ji}(n) * y_i(n)) \quad (\text{B.3})$$

where w_{ji} refers to the strength of the synaptic connection from presynaptic neuron i to neuron j , p is the total number of inputs, and $y_i(n)$ is the input signal to neuron j from presynaptic neurons in layer i . The synaptic weight w_{j0} corresponds to the threshold applied to neuron j (given that $y_0 = -1$).

Since e_j refers to the error of a particular output neuron j , we may define an instantaneous sum of squared errors for all the output neurons in the network as:

$$\varepsilon(n) = (1/2) * \sum_{j \in C} (e_j(n))^2 \quad (\text{B.4})$$

where the set C refers to the neurons in the output layer of the network. Finally, if N refers to the total number of patterns in the training set, we may sum $\varepsilon(n)$ over all n , and obtain an averaged squared error as follows:

$$\varepsilon_{ave} = (1/N) * \sum_{n=1}^N (\varepsilon(n)) \quad (\text{B.5})$$

The objective of the backpropagation algorithm is to minimize the average squared error, ε_{ave} , with respect to the synaptic weights of the network. Therefore, the algorithm applies a correction $\Delta w_{ij}(n)$ to the synaptic weight $w_{ji}(n)$, where $\Delta w_{ij}(n)$ is proportional to the instantaneous gradient $\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)}$. Also, $w_{ji}(n)$ refers to the strength of the synaptic connection that neuron j receives from presynaptic neuron i at iteration n . According to the chain rule of differentiation, we know that the gradient is:

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = \left(\frac{\partial \varepsilon(n)}{\partial e_j(n)} \right) * \left(\frac{\partial e_j(n)}{\partial y_j(n)} \right) * \left(\frac{\partial y_j(n)}{\partial v_j(n)} \right) * \left(\frac{\partial v_j(n)}{\partial w_{ji}(n)} \right) \quad (\text{B.6})$$

APPENDIX B. DERIVATION OF THE STATIC BACKPROPAGATION ALGORITHM 84

Thus the gradient $\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)}$ represents a *sensitivity factor* that defines the direction of change of the instantaneous error in weight space. If we compute each of the terms in equation B.6, we have:

$$\frac{\partial \varepsilon(n)}{\partial e_j(n)} = e_j(n) \quad (\text{B.7})$$

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \quad (\text{B.8})$$

$$\frac{\partial y_j(n)}{\partial v_j(n)} = s'(v_j(n)) \quad (\text{B.9})$$

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \quad (\text{B.10})$$

where $s'(v_j(n))$ refers to the derivative of the sigmoidal nonlinearity with respect to $v_j(n)$. If $s(v)$ is the logistic function, then we have:

$$y_j(n) = s(v_j(n)) = 1/(1 + \exp(-v_j(n))) \quad (\text{B.11})$$

$$\frac{\partial y_j(n)}{\partial v_j(n)} = s'(v_j(n)) = y_j(n) * (1 - y_j(n)) \quad (\text{B.12})$$

Thus the gradient may be evaluated as:

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = -e_j(n) * s'(v_j(n)) * y_i(n) \quad (\text{B.13})$$

The correction $\Delta w_{ji}(n)$ to a synaptic weight $w_{ji}(n)$ is thus defined by:

$$\Delta w_{ji}(n) = -\eta * \frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} \quad (\text{B.14})$$

where η is a constant referred to as the *learning – rate parameter*. Alternatively, we may represent the weight update algorithm as follows:

$$\Delta w_{ji}(n) = \eta * \delta_j(n) * y_i(n) \quad (\text{B.15})$$

where $\delta_j(n)$ is referred to as the *local gradient* arising at neuron j at iteration n . Thus the local gradient is itself defined as:

$$\delta_j(n) = -\left(\frac{\partial \varepsilon(n)}{\partial e_j(n)}\right) * \left(\frac{\partial e_j(n)}{\partial y_j(n)}\right) * \left(\frac{\partial y_j(n)}{\partial v_j(n)}\right) \quad (\text{B.16})$$

As is evident in equation B.15, the local gradient defines the required changes in the synaptic weights of neuron j . If neuron j is an output neuron, we may explicitly compute an error signal and a local gradient associated with the output neuron. However, if a neuron is a hidden node, we must backpropagate the error signals arising at the output of the network, and subsequently adjust the synaptic weights of the hidden neuron. Thus we subsequently delve into how the local gradient is computed for hidden neurons.

B.1 The computation of the local gradient for hidden neurons

For a hidden neuron j , the local gradient is defined as follows:

$$\delta_j(n) = -\left(\frac{\partial \varepsilon(n)}{\partial y_j(n)}\right) * \left(\frac{\partial y_j(n)}{\partial v_j(n)}\right) \Rightarrow \delta_j(n) = -\left(\frac{\partial \varepsilon(n)}{\partial y_j(n)}\right) * s'(v_j(n)) \quad (\text{B.17})$$

where neuron j is assumed to be a hidden neuron, and $s'(v_j(n))$ represents the differentiation of the sigmoidal nonlinearity with respect to the internal activity level of the neuron. In order to compute the partial derivative $\frac{\partial \varepsilon(n)}{\partial y_j(n)}$, consider again the instantaneous sum of the square errors across all output nodes:

$$\varepsilon(n) = (1/2) * \sum_{k \in O} (e_k^2(n)) \quad (\text{B.18})$$

where neuron k is an output neuron, and the set O refers to the output layer of the network. Thus if we differentiate with respect to the output signal of neuron j , we have:

$$\frac{\partial \varepsilon(n)}{\partial y_j(n)} = \sum_k (e_k(n)) * \left(\frac{\partial e_k(n)}{\partial v_k(n)} \right) * \left(\frac{\partial v_k(n)}{\partial y_j(n)} \right) \quad (\text{B.19})$$

We know that the error at an output neuron k is:

$$e_k(n) = d_k(n) - y_k(n) \Rightarrow e_k(n) = d_k(n) - s(v_k(n)) \quad (\text{B.20})$$

where $s(v)$ refers to the sigmoidal nonlinearity. Thus we have:

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -s'(v_k(n)) \quad (\text{B.21})$$

Moreover, we know that the net internal activity level of an output neuron k is:

$$v_k(n) = \sum_{j=0}^q w_{kj}(n) * y_j(n) \quad (\text{B.22})$$

where q is the total number of synaptic inputs from hidden neurons j to output neuron k , the synaptic weight w_{k0} corresponds to the threshold applied to neuron k , and $y_0 = -1$. Therefore,

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \quad (\text{B.23})$$

Therefore, we may evaluate the partial derivative:

$$\frac{\partial \varepsilon(n)}{\partial y_j(n)} = - \sum_k (e_k(n)) * (s'_k(v_k(n))) * (w_{kj}(n)) \quad (\text{B.24})$$

Therefore, the local gradient associated with hidden neuron j is:

$$\delta_j(n) = s'_j(v_j(n)) * \sum_k (e_k(n)) * (s'_k(v_k(n))) * (w_{kj}(n)) \quad (\text{B.25})$$

where neuron j is a hidden neuron, $s_j(v_j(n))$ refers to the sigmoidal nonlinearity associated with hidden neuron j , and $s_k(v_k(n))$ is the sigmoidal nonlinearity associated with neuron k in the subsequent postsynaptic layer, or the output layer.

B.2 Summary of the static backpropagation algorithm

We may summarize the weight update algorithm of the backpropagation algorithm through the delta rule as follows:

$$\Delta w_{ji}(n) = -\eta * (\delta_j(n)) * (y_i(n)) \quad (\text{B.26})$$

where $y_i(n)$ is the input signal to neuron j , and the local gradient $\delta_j(n)$ depends on whether neuron j is an output node or a hidden node. If neuron j is an output node, the local gradient is:

$$\delta_j(n) = e_j(n) * s'(v_j(n)) \quad (\text{B.27})$$

where neuron j is an output node. Moreover, if neuron j is a hidden node, the local gradient associated with the neuron is:

$$\delta_j(n) = s'_j(v_j(n)) * \sum_k (e_k(n)) * (s'_k(v_k(n))) * (w_{kj}(n)) \quad (\text{B.28})$$

where neuron j is a hidden node, and the index k refers to neurons in the postsynaptic layer which receive inputs from the hidden neuron j .

B.3 Example: A neural network implementation of a square-root finding algorithm

An instance of how a neural network may be adapted to learn a particular problem is shown in appendix D under the function entitled $y_k = \text{sqrtfind}(x)$. In this program \vec{x} is a vector of four numbers between 0 and 1, and the objective of the learning procedure is to adapt the neural network to learn a square root finding algorithm for the values contained in \vec{x} . Thus the target vector and the output vector y_k are also vectors of length four, and each of the elements of the target vector correspond to the square root of the elements in \vec{x} . The network uses one hidden layer comprised of eight sigmoidal neurons, and four sigmoidal output neurons. Since the sigmoidal

*APPENDIX B. DERIVATION OF THE STATIC BACKPROPAGATION ALGORITHM*88

nonlinearities utilized in this function are bounded between zero and one, we only consider input vectors whose elements are within the range 0 and 1. Upon iterating through k iterations (where k is defined in the program code), the network output y_k converges to the target vector.

Appendix C

Introduction to Phase Space Analysis

One of the most common methods utilized to model a natural system is through a system of differential equations. The subject of nonlinear dynamics is a powerful tool in the modeling of natural systems, and an excellent introduction to the subject may be found in [44]. Instances of the application of nonlinear dynamics specifically towards neuronal signaling are found in [3], [45] and [6]. Through this appendix we provide a short introduction to the methods of phase space analysis.

In the simplest of instances, a system has only one variable of interest, and we may thus derive a differential equation that characterizes how our variable evolves with time. As an example, let us consider the one dimensional system shown below:

$$\frac{dv}{dt} = v \tag{C.1}$$

We may picture how the system of equation 1 evolves by considering the phase space, which depicts the state of a variable at a particular time, along with the velocity vector associated with the variable at that point in time. The term velocity in nonlinear dynamics is often used to indicate the derivative of a variable at an instant of time, i.e. the direction in which it is going to evolve. In the simple system defined by C.1, the differential equation $\dot{v} = v$ represents a vector field on the phase line. In order to sketch the vector field, we may graph \dot{v} versus v , and draw representative arrows

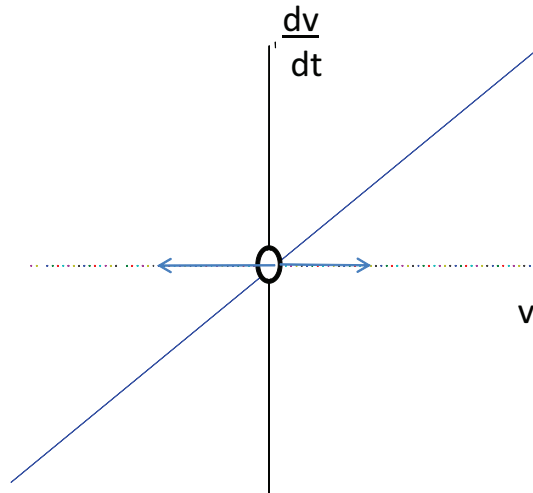


Figure C.1: Phase line representation for $\dot{v} = v$

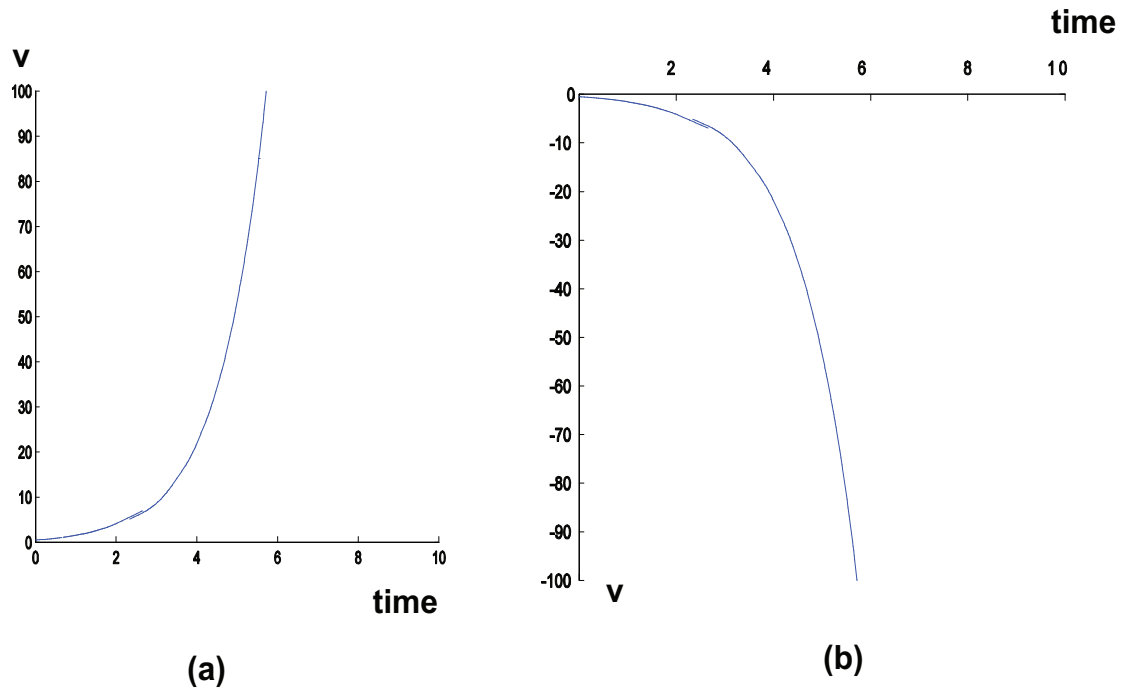
that indicate the velocity vector at each point v , as is depicted in figure C.1.

As is evident from figure C.1, when $\dot{v} > 0$, v increases, and thus the flow is to the right. Conversely, when $\dot{v} < 0$, v is decreasing, and thus the flow is to the left. Since equation C.1 defines a one dimensional linear system, we only have one point in the phase space where $\dot{v} = 0$, and this point corresponds to an equilibrium point. The equilibrium point occurs at $v = 0$, and at this point there is theoretically no flow. However, this is an unstable fixed point since the flow diverges away from this fixed point. Thus any perturbation to the system from the equilibrium point will amplify itself and drive the state variable v to $\pm\infty$. Depicted in figure C.2 are the time responses of the variable v .

As is observed in figure C.2, if the initial value of v is greater than zero, i.e. the initial state of the system is situated to the right of the unstable equilibrium point, v continues to grow exponentially for all $time \geq 0$. Conversely, if the initial state of v is negative, then $v \rightarrow -\infty$ as $time \rightarrow \infty$.

We may extend our method of analysis and consider a general two dimensional system of the form:

$$\frac{dx}{dt} = \dot{x} = f(x, y) \tag{C.2}$$



(a) Exponential growth corresponding to $v(0) > 0$

(b) Exponential fall-off corresponding to $v(0) < 0$

Figure C.2: Time response for $\dot{v} = v$

$$\frac{dy}{dt} = \dot{y} = g(x, y) \quad (\text{C.3})$$

For instance, let us consider the following:

$$\frac{dx}{dt} = \dot{x} = -y \quad (\text{C.4})$$

$$\frac{dy}{dt} = \dot{y} = -x \quad (\text{C.5})$$

Solutions to the two dimensional system in equations (C.4,C.5) are essentially trajectories in the (x, y) space, known as the phase space. The conventional method for viewing the phase space is to plot velocity vectors at each point in the (x, y) plane,

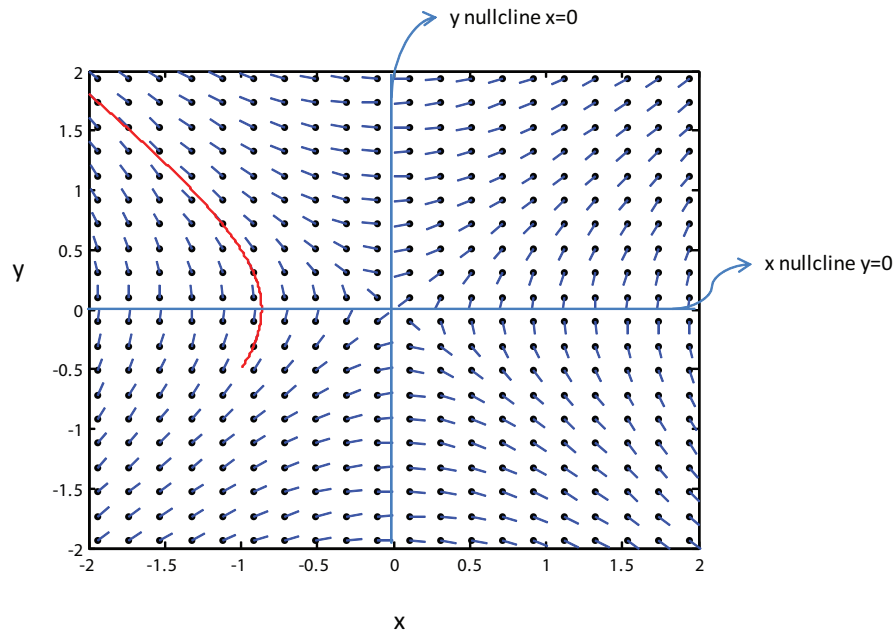


Figure C.3: Phase portrait along with representative state trajectory of system (C.4,C.5)

(x_o, y_o) ; the directions of the velocity vectors are defined by the system of differential equations $\dot{x} = -y$ and $\dot{y} = -x$. Thus depicted in figure C.3 is the phase space of the system (C.4,C.5).

In figure C.3, the vector field assumes different directions in different regions of the phase plane. The set of points where the vector field changes its horizontal direction of flow is known as the x-nullcline, and it is defined by the equation $f(x, y) = 0$. Similarly, the set of points where the vector field changes its vertical direction is known as the y-nullcline, and it is defined by $g(x, y) = 0$. Thus in the simple system depicted in C.3, the x-nullcline is defined by the line $y = 0$, and the y-nullcline is defined by the line $x = 0$. The intersection of these two nullclines defines the equilibrium point of the system, and in this instance, it is $(x^*, y^*) = (0, 0)$.

The nullclines and equilibria define the steady-state characteristics of the system, and they provide a geometrical representation of the system's evolution when released from certain initial conditions. In order to determine the stability of an equilibrium, we must consider the behavior of the vector field in a small region around the equilibrium point. Subsequently, we may utilize some basic concepts of linear algebra to

determine the stability of an equilibrium.

Let us once again consider the general two-dimensional dynamical system, having an equilibrium point (x^*, y^*) . We may linearize the nonlinear functions $f(x, y)$ and $g(x, y)$ near the equilibrium point as follows:

$$f(x, y) = a(x - x^*) + b(y - y^*) + \text{higher order terms} \quad (\text{C.6})$$

$$g(x, y) = c(x - x^*) + d(y - y^*) + \text{higher order terms} \quad (\text{C.7})$$

where $a = \frac{\partial f}{\partial x}(x^*, y^*)$, $b = \frac{\partial f}{\partial y}(x^*, y^*)$, $c = \frac{\partial g}{\partial x}(x^*, y^*)$, and $d = \frac{\partial g}{\partial y}(x^*, y^*)$ correspond to the partial derivatives of $f(x, y)$ and $g(x, y)$ with respect to the state variables x and y , evaluated at the equilibrium point (x^*, y^*) . When evaluating the stability of a two-dimensional system, we may consider the following corresponding linear system:

$$\dot{u} = au + bv \quad (\text{C.8})$$

$$\dot{v} = cu + dv \quad (\text{C.9})$$

where $u = (x - x^*)$ and $v = (y - y^*)$ are the deviations from the equilibrium point. The implicit assumption underlying the linearization method for determining stability is that we may neglect the higher order terms.

The linearization matrix:

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

is known as the Jacobian matrix of the general two-dimensional system at the equilibrium (x^*, y^*) . The eigenvalues of the Jacobian matrix are given by the characteristic equation $\det(\mathbf{A} - \lambda \mathbf{I}) = 0$, where \mathbf{I} refers to the identity matrix. Thus the characteristic equation becomes:

$$\det \begin{bmatrix} a - \lambda & b \\ c & d - \lambda \end{bmatrix} = 0$$

Expanding the determinant, we get:

$$\lambda^2 - \lambda * (a + d) + ad - bc = 0 \quad (\text{C.10})$$

$$\Rightarrow \lambda^2 - \lambda * \tau + \Delta = 0 \quad (\text{C.11})$$

where $\tau = a + d$ and $\Delta = \det(A) = ad - bc$. We may solve for λ as follows:

$$\lambda_1 = \frac{\tau + \sqrt{\tau^2 - 4\Delta}}{2} \text{ and } \lambda_2 = \frac{\tau - \sqrt{\tau^2 - 4\Delta}}{2} \quad (\text{C.12})$$

If we know an initial condition to the system, then the general solution to the system is simply:

$$\vec{x} = c_1 * \exp(\lambda_1 * t) \vec{v}_1 + c_2 * \exp(\lambda_2 * t) \vec{v}_2 \quad (\text{C.13})$$

where \vec{v} refers to a fixed vector to be determined. From linear systems theory, we know that the case where $\lambda > 0$ corresponds to exponential growth, and thus implies instability. On the other hand, $\lambda < 0$ implies exponential decay, so $u(t) \rightarrow 0$ and $v(t) \rightarrow 0$, which implies that $x(t) \rightarrow x^*$ and $y(t) \rightarrow y^*$, and thus the equilibrium is a stable point. Finally, λ defines a pair of complex-conjugates if $4\Delta > \tau^2$. If the real part of the complex number is less than zero, then the equilibrium point is a stable focus, while if the real part of the complex number is greater than zero, then the equilibrium point is an unstable focus.

Appendix D

Computer Simulations Program Code

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Resonate-and-Fire Neuron
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Set model parameters
Damping1 = 0.1; % Resonator case
Damping2 = 0.9; % Integrator case
Omega = (2*pi); %natural freq = 1 Hz

f = Omega/(2*pi); deltaT = 1/f; StimStrength = 100; StimPeriod = 1;
Vth = 0.05; %Threshold
Vpeak = 30; Vrest = 0.0;

dt = 0.01; tend = 10; %End time in seconds
fireflag = 0;

time = 0:dt:tend; %Create time vector in seconds
deltaT = 1/dt; %Declare after how many time steps we apply stim
start_neg = 1;

```

```

%Create stimulus vector

PosStim = zeros(1,length(time));    %positive stimulus
NegStim = zeros(1,length(time));    %negative stimulus
VStim = zeros(2,length(time));      %Overall stimulus

VStim(1,:) = PosStim(1,:); VStim(2,:) = NegStim(1,:);

%plot stimulus
figure(1)
plot(time,VStim(1,:), 'b'); hold on;

plot(time,VStim(2,:), 'k'); hold off;

V = zeros(1,length(time));          %initialize V
W = zeros(1,length(time));          %initialize W

V2 = zeros(1,length(time)); W2 = zeros(1,length(time));

BinaryOut = zeros(1,length(time)); k = 0;

for t = 2:length(time)

    V(t) = V(t-1) + dt*((W(t-1)) + VStim(1,t) + VStim(2,t));
    W(t) = W(t-1) + dt*(-1.0*(2*Damping1*Omega*W(t-1) + Omega*Omega*V(t-1)));
    %Damping*Wn = 1/(R*C), Wn = 1/sqrt(LC); Resonator Simulation

    V2(t) = V2(t-1) + dt*((W2(t-1)) + VStim(1,t) + VStim(2,t));
    W2(t) = W2(t-1) + dt*(-1.0*(2*Damping2*Omega*W2(t-1) + Omega*Omega*V2(t-1)));
    % Integrator Simulation

    if (V(t)>Vth)
        %update binary output if potential exceeds threshold
        BinaryOut(1,t) = 1;
        %increment amount of time active
        k = k + 1;
    end
end

```

```
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%FitzHugh-Nagumo Model Simulations
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function dv=fhnde(t,v)
```

```
%FHNDE State transition function for FitzHugh-Nagumo Model.
```

```
%Program code adapted from course materials: ELEC 472 Digital Signal
```

```
%Processing
```

```
% initialize the output vector
```

```
dv=[ 0 ; 0 ];
```

```
% initialize parameters of FHN model
```

```
b=1; c=5; a=0.99;
```

```
% compute the values of the output vector
```

```
dv(1)=v(1)*(a-v(1))*(v(1)-1)-v(2);
```

```
dv(2)=b*v(1)-c*v(2);
```

```
function [t,v] = fhnsim(odefun,v0,tdelta,tstart,tend)
```

```
%FHNSIM Simulates the behavior of a system with a given state transition function.
```

```
%Code adapted from course materials in ELEC 472: Digital Signal Processing
```

```
t = tstart:tdelta:tend; %time vector
```

```
v = zeros(length(t),length(v0)); %output state matrix
```

```
Istim = zeros(2,length(t)); Istim(1,2000:4000) = 50;
```

```
v(1,:)=v0(:).'; %initial conditions
```

```
%loop through matrix v and evaluate the behavior of system
```

```
for n=2:length(t)
```

```
dv = odefun(t(n-1),v(n-1,:).') + Istim(1,n);
```

```
v(n,:) = v(n-1,:) + tdelta*dv.');
```

```

end

%hold on;
v1 = linspace(-10,10,30);
v2 = linspace(-10,10,30);
phaseport(@fhnde,v1,v2)
title('Phase Portrait for FitzHugh-Nagumo Model')
axis([-10 10 -10 10])
hold on;

%view evolution of states in state space
plot(v(:,1),v(:,2),'r'); hold on;

%plot nullclines

% initialize parameters of FHN model

b=1; c=5; a=0.99;

y1 = v1.*(a-v1).*(v1-1);
y2 = (b/c).*v1;

plot(v1,y1,v1,y2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% SQUARE ROOT FINDING ALGORITHM WITH STATIC NEURAL NETWORKS

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function yk = sqrtfind(x);
% function SQRTFIND(X) finds the square root of the elements contained in
% the vector x, and outputs these values in the vector yk. The objective of
% this program is to implement this square root finding algorithm through a
% neural network adapted through error backpropagation

% obtain the length of the vector x
p = length(x);

```

```
% initialize and define target vector T
T = zeros(p,1);
T = sqrt(x);

% obtain length of target vector
tN = length(T);

% define n as the number of training iterations
n = 2000;

% define output matrix yk, which stores the outputs of output neurons
yk = zeros(tN,n);

% Initialize error matrix Ek, which stores the difference between the target
% value of output neuron k and the actual output of output neuron k
Ek = zeros(tN,n);

% initialize ensemble error matrix phi, which stores the sum of all the
% errors Ek, across all output neurons k

phi = zeros(1,n);

% Define hidden layer comprised of h hidden neurons
h = 8;

% initialize vector vj to store activity levels of hidden neurons
vj = zeros(h,1);

% initialize vector yj to store outputs of hidden neurons, where
% yj=10/(1+exp(-vj))

yj = zeros(h,1);

% initialize input weight matrix wij; this matrix will connect inputs to
% first layer of hidden neurons. Thus we require this matrix to contain p
% columns ( p = length(x), i.e. the number of inputs we receive) and
% h rows (h = number of hidden neurons)

wij = ones(h,p);
```

```
% initialize hidden weight matrix wjk; this matrix connects hidden neurons
% j to output neurons k; Thus we require this matrix to contain tN rows,
% where tN is the length of our target vector; also, we need this matrix to
% contain h columns, where h is the number of hidden neurons

wjk = ones(tN,h);

% initialize vector vk to store activity levels of output neurons
vk = zeros(tN,n);

% define momentum constant eta
eta = 0.5;

% initialize matrix sprime to store derivative of sigmoidal nonlinearity
% for output neurons k

sprimek = zeros(tN,n);
sprimej = zeros(h,n);

temp = zeros(tN,n);

% loop through each training iteration 1 through n
for iter = 1:n

% loop through inputs and weight matrix to compute activity levels vj and
% outputs of hidden neurons yj
    for j = 1:h
        vj(j,1) = wij(j,:)*x;
        yj(j,1) = 1/(1+exp(-vj(j,1)));

        sprimej(j,iter) = yj(j,1).*(1-yj(j,1));
    end

% loop through inputs, obtained from the outputs of hidden neurons, and
% weigh by hidden weight matrix wjk to compute activity levels of output
% neurons, vk, and corresponding outputs yk
```

```

    for k=1:tN
        vk(k,iter) = wjk(k,)*yj;
        yk(k,iter) = 1/(1+exp(-vk(k,iter)));

        % obtain error Ek from each output neuron k
        Ek(k,iter) = T(k,1)-yk(k,iter);

        % Compute derivative of sigmoidal nonlinearity for output
        % neurons

        sprimek(k,iter) = yk(k,iter).*(1-yk(k,iter));
        %temp3(1,iter) = sprimek(k,iter)*wjk(k,j);

        temp(k,iter) = Ek(k,iter)*sprimek(k,iter);
    end

    % compute the ensemble error phi, which sums the errors across
    % all output neurons k
    phi(1,iter) = 0.5*sum((Ek(:,iter)).^2);

    % Implement weight update algorithm for weight matrix wjk,
    % which connects hidden neurons h to output neurons k

    for row = 1:tN
        for col = 1:h

            delta_wjk(row,col) = Ek(row,iter).*(yk(row,iter).*(1-yk(row,iter))).*yj(col,1);

            wjk(row,col) = wjk(row,col) + eta*delta_wjk(row,col);

        end
    end

    % Implement weight update algorithm for weight matrix wij,
    % which connects inputs xi to hidden neurons j

    for inrow = 1:h
        for incol = 1:p

```



```

        wjk_train = sum(wjk(:,inrow).*sprimek(incol,iter));

        delta_wij(inrow,incol) = (sprimej(inrow,iter).*x(incol,1))*wjk_train;

        wij(inrow,incol) = wij(inrow,incol) + eta*delta_wij(inrow,incol);

    end
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% TRANSLATION EXAMPLE WITH TEMPORAL BACKPROPAGATION ALGORITHM

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function yk = translate(x);
% YK = TRANSLATE(X) The objective of this function is to implement a neural
% network, and adapt this network to learn the protein code. Thus x[n]
% specifies a binary word vector in time index n, and the network maps x[n]
% to its corresponding amino acid; The amino acid is also specified by a
% temporal binary code

% obtain length of input vector x, and create time vector that is 5
% times as long as input vector
L = 5*length(x);

% define numpoints to be number of time indices we would like to consider
% for impulse response hj(1) and hk(1)
numpoints = 5*L;

% Specify number of hidden layers
num_hid = 1;

% define time vector n, which runs upto L

```

```
n = linspace(1,L,numpoints);
M = length(n);

% create vector x[n-lM]=x_eff
x_eff = zeros(1,2*M+length(x));

for i = 1:length(x)

    x_eff(i+M) = x(1,i);

end

% Create target vector D[n]; we need the time alignment of D[n] to match
% that of yk[n]. Thus we may allocate a temporal desired vector if we know
% the time memory of a neuron within each layer. Here we assume that each
% layer is comprised of neurons with equal M=L*5=150. Also, we know that we
% will be utilizing one hidden layer

D = zeros(1,2*M + length(x) - 2);
D(1,1:M) = ones;

% initialize output vector yk[n], which should be a [1XL] row vector; We are
% using only one row because we have only one output neuron in this example
yk = zeros(1,2*M + length(x) - 2);

% Define number of hidden neurons h that we would like to use; in this case
% try h=5
h = 5;

% initialize vector hj to store impulse response of a given neuron j;
hj = ones(h,length(n));

% assign an impulse response to each hidden neuron j; in this instance
% assume that we use alpha = 0.5, and OMEGA = 2*pi*50 for all neurons

alphaj = 0.5;
omegaj = 2*pi*50;

% define number of training iterations k
```

```

k = 1000;

% initialize a = 1 for all neurons where y = 1/(1+exp(-a*v))
a=1;

% Define learning rate momentum constant eta
eta = 0.5;

% initialize matrix sprime to store derivative of sigmoidal nonlinearity
% for output neurons k
sprimek = zeros(1,(2*M+length(x))-2);
sprimej = zeros(h,M+length(x)-1);

% initialize matrix shift_sprime to store time shifted gradients and
% derivatives of sigmoidal nonlinearity
shift_sprimej = zeros(h,2*M+length(x));

% Initialize error matrix Ek(n) = D(n) - yk(n)
Ek = zeros(k,2*M+length(x)-2);

% Initialize matrix to store local gradient for output neurons
delta_wjk = zeros(1,2*M+length(x)-2);
shift_deltajk = zeros(1,3*M+length(x));

% Initialize ensemble error matrix which computes the total error for all
% time n
phi = zeros(k,1);

for hrow = 1:h
    for hcol = 2:length(n)

        hj(hrow,hcol) = exp(-(alphaj)*hcol).*cos((omegaj).*hcol);

    end
end

% Initialize input weight matrix wij which connects x[n] to our hidden
% neurons j
wij = ones(h,1);

```

```
% Create temporal weight matrix which stores the effective weight on a
% synaptic connection...i.e. weff(j,1) = wij*hj(1)

wij_eff = ones(h,length(n));

for wijrow = 1:h
    for wijcol = 2:length(n)

        wij_eff(wijrow,:) = wij(wijrow,:).*hj(wijrow,:);

    end
end

%intialize weight matrix wjk that connects hidden neurons j to output
%neuron k

wjk = ones(h,1);

% assign an impulse response to output neuron k; in this instance
% assume that we use alpha = 0.5, and OMEGA = 2*pi*50 for all neurons

hk = ones(1,length(n));

alphak = 0.5;
omegak = 2*pi*50;

for hkcol = 2:length(n)

    hk(1,hkcol) = exp(-(alphak)*hkcol).*cos((omegak).*hkcol);

end

% initialize and create effective temporal weight matrix wjk_eff = wjk.*hk

wjk_eff = ones(h,length(n));

for wjkrow = 1:h
    for wjkcol = 2:length(n)
```

```
wjk_eff(wjkrow,:) = wjk(wjkrow,:).*hk(1,:);

end

end

for iter = 1:k

% Create matrix xnetj which stores the input to neuron j for time index [n]

for row = 1:h
    for col = 1:length(x)

        xnetj(row,col) = x(col);

    end
end

% compute matrix vj[n], which stores the activity levels of hidden neurons
% j; we have that vj[n] is hj[n] convolved with xnetj[n]

for jrow = 1:h

    vj(jrow,:) = conv(xnetj(jrow,:),wjk_eff(jrow,:));

end

%obtain dimensions of matrix v, i.e. obtain number of rows (which we know
%to be five), and number of columns (time length of vj)
[numh,lvj] = size(vj);

% compute outputs of hidden neurons yj = 1/(1+exp(-a*vj))

% initialize hidden output matrix yj, and make it to be of size [numh X
% lvj] (dimensions of vj)
yj = zeros(numh,lvj);

% Compute the matrix yj, and also compute the derivative of the sigmoidal
```

```

% nonlinearity and store the result in matrix sprimej
% Also compute the shifted time delayed matrix shift_sprimej

for yjrow = 1:numh
    for yjcol = 1:lvj

        yj(yjrow,yjcol) = 1/(1+exp(-a*vj(yjrow,yjcol)));

        sprimej(yjrow,yjcol) = yj(yjrow,yjcol).*(1-yj(yjrow,yjcol));

        shift_sprimej(yjrow,yjcol+M) = sprimej(yjrow,yjcol);
    end
end

% initialize matrix xnetk_temp[n], which should have dimensions of [h X lvj]
% we shall use this matrix to store the time indices of yj[n]
xnetk_temp = zeros(h,lvj);

for kcol = 1:lvj
    for krow = 1:h

        xnetk_temp(krow,kcol) = yj(krow,kcol);

    end
end

[numj,lvk] = size(xnetk_temp);
% compute matrix vk_temp[n] which stores the activity level of
% output neuron k; we have that vk[n] is wjk_eff[n] convolved
% with xnetk_temp[n]

for vkrow = 1:h

    vk_temp(vkrow,:) = conv(wjk_eff(vkrow,:),xnetk_temp(vkrow,:));

end

[rowvk, colvk] = size(vk_temp);

```

```

% initialize vector vk[n] to store the activity level of output neuron k
vk = zeros(1,colvk);

for vkcol = 1:colvk

    vk(1,vkcol) = sum(vk_temp(:,vkcol));

end

% compute output matrix yk[n] and also compute derivative of sigmoidal
% nonlinearity....the loop runs from 1 to colvk

for colyk = 1:colvk

    yk(1,colyk) = 1/(1+exp(-a*vk(1,colyk)));

    sprimek(1,colyk) = yk(1,colyk).*(1-yk(1,colyk));

    % Compute the error at the output neuron for this iteration
    E(iter,colyk) = D(1,colyk)-yk(1,colyk);

    % Compute local gradient for output neuron
    delta_wjk(1,colyk) = E(iter,colyk).*sprimek(1,colyk);

    % Obtain time shifted local gradient for output neuron
    shift_deltajk(1,colyk+M) = delta_wjk(1,colyk);
end

% Compute the ensemble error for this training iteration
phi(iter,1) = 0.5*(sum((E(iter,:).^2)));

% Implement weight update algorithm for weight matrix weff_jk that connects
% hidden neurons j to output neurons k

shift_deltajk = shift_deltajk';

for jkrow = 1:h
    for jkcol = 1:length(n)

```

```

wjk_eff(jkrow,jkcol)=wjk_eff(jkrow,jkcol)+eta*delta_wjk(1,jkcol).*yj(jkrow,jkcol);

    end
end

% Implement weight update algorithm for weight matrix weff_ij that connects
% input node to hidden neurons j

for ijrow = 1:h
    for ijcol = 1:length(n)

grad_wij_eff(ijrow,ijcol) =
shift_sprimej(ijrow,ijcol).*(sum((shift_deltajk(ijcol,:)).*wjk_eff(ijrow,ijcol)));

wij_eff(ijrow,ijcol) =
wij_eff(ijrow,ijcol) + eta.*grad_wij_eff(ijrow,ijcol).*(x_eff(1,ijcol+M));

    end
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%BIOLOGICAL NEURAL NETWORK TOOLBOX CODE

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function dydt = raf(t , y)
% Resonate-and-fire neuron state space system definition

%Defining BNN model as a global variable. DON'T CHANGE THIS SECTION.
global net

tstart = 0; tdelta = 1e-3; tend = 10;

t = tstart:tdelta:tend; %time vector
y = zeros(3,length(t)); %output state matrix

```



```

Damping = 0.5; %=1/R*C
NatFreq = 2*pi; %=1/sqrt(LC)

I = zeros(2,length(t)); I(1,2:3) = 200;

dv1 = zeros(1,length(t));
dv2 = zeros(1,length(t));

for i = 2:length(t)
    % Define RAF Differential equations
    dv1(1,i)    = y(2,i-1);
    dv2(1,i)    = (-Damping)*(y(2,i-1)) - (NatFreq^2)*(y(1,i-1)) + I(1,i);

    y(1,i) = y(1,i-1) + tdelta*(dv1(1,i));
    y(2,i) = y(2,i-1) + tdelta*(dv2(1,i));

    y(3,i) = 1/(1+exp(-(y(1,i))));
end

figure(1); plot(t,I(1,:)); figure(2); plot(t,y(1,:)); figure(3); plot(t,y(2,:));
figure(4); plot(t,y(3,:)); figure(5); plot(y(1,:),y(2,:));

%Implement RAF Neuron Model
function NeuronModel = rafmodel(neuron_fun , model_type ,
state_num , model_param , spike_det_fun ,
spike_det_fun_param , reset_fun , reset_fun_param)
% Define and build Resonator Neuron model

Damping = 0.5; NatFreq = 2*pi; vth=0;

% neuron_fun refers to the diff. eqns. of neuron model;
neuron_fun      = 'raf';

% model_type is CDE by default;
model_type      = 'CDE';

% state_num refers to the number of state variables in the neuron model;
state_num       = 3;

```

```
%model_param refers to the parameters defined within the neuron model;
model_param      = [Damping; NatFreq];

%spike_det_fun refers to the function that detects the condition to spike;
spike_det_fun    = 'threshold';
spike_det_fun_param = 'vth';

%reset_fun defines the functions that are called upon firing and reset
reset_fun        = 'none';
reset_fun_param  = 'def';

NeuronModel = newneuron(neuron_fun , model_type , state_num , model_param , ...
    spike_det_fun , spike_det_fun_param , reset_fun , reset_fun_param);

return

%Build neural network multilayer perceptron architecture

function NetArch = rafarch(neuron_num, input_num, input_type,
    neuron_type, weight_type, delay_type)

%creates a feedforward multilayer architecture, i.e. neurons are divided
%into groups called layers.

%neuron_num defines the number of neurons in each layer of the
%neural network [N_1,N_2,...N_k] with N_i being number of neurons in layer
%i;
neuron_num = [1,5,5,1];

%input_num is the number of inputs to the network: M
input_num = 1;

%input_type is a vector of size(input_num), i.e. of length M,
%and defines the type of inputs for each of the inputs defined in input_num;
%analog input is -1 and spiking input is +1
input_type = -1;

%neuron type is a vector of size N, where N=neuron_num; each element of the
%vector corresponds to a certain type of neuron; +1 is excitatory, -1 is
```

```
%inhibitory, and 0 is hybrid
neuron_type = ones(length(neuron_num),1);

% weight_type is a string indicating how to initialize the weight matrixes:
% random for a random uniform distribution, normal for a random normal
% distribution, one for all weights equal to 1, and zero for all weights
% equal to zero. Default is random.
weight_type = 'random';

% delay_type is a string indicating how to initialize the delay matrixes:
% random for a random uniform distribution, one for all delays equal to 1,
% and zero for all delays equal to zero. Default is zero.
delay_type = 'zero';

NetArch = newarchff(neuron_num, input_num, input_type,
neuron_type, weight_type, delay_type);

return
```

References

- [1] E. Kandel, J. Schwartz, and T. Jessell, *Principles of Neural Science* (Cambridge, MA: MIT Press, 1991).
- [2] D. Johnston, S. M.-S. Wu, and R. Gray, *Foundations of Cellular Neurophysiology* (Cambridge, MA: MIT Press, 1995).
- [3] P. Dayan and L. Abbott, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems* (Cambridge, MA: MIT Press, 2001).
- [4] J. V. Tranquillo, *Quantitative Neurophysiology* (San Rafael, CA: Morgan and Claypool Publishers, 2008).
- [5] J. Rinzel, *Models in Neurobiology in 'Nonlinear Phenomena in Physics and Biology'* R. Enns, B. Jones, R. Miura, S. Rangnekar (eds.) (New York: Plenum Press, 1981).
- [6] E. Izhikevich, *Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting* (Cambridge, MA: MIT Press, 2007).
- [7] S. Haykin, *Neural Networks - A Comprehensive Foundation* (New York: Macmillan College Publishing Company, 1994).
- [8] S. Samarasinghe, *Neural Networks for Applied Sciences and Engineering: From Fundamentals to Complex Pattern Recognition* (New York: Auerbach Publications, 2007).
- [9] *Methods in Neuronal Modeling: From Synapses to Networks - C. Koch and I. Segev (eds.)* (Cambridge, MA: MIT Press, 1989).
- [10] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, **5**, 115 (1943).

- [11] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, **65**, 386 (1958).
- [12] F. Rosenblatt, *Principles of Neurodynamics* (Washington D.C.: Spartan Books, 1962).
- [13] B. Widrow and M. Hoff, "Adaptive switching circuits," IRE WESCON Convention Record, pp. 96–104 (1960).
- [14] P. Werbos, *Beyond regression: New tools for prediction and analysis in the behavioral sciences, PhD. Thesis: Harvard University* (MIT Press, Cambridge, MA, 1974).
- [15] G. Hinton, D. Rumelhart, and R. Williams, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition (D.E. Rumelhart and J.L. McClelland, eds.)* (Cambridge, MA: MIT Press, 1986).
- [16] L. Lapique, "Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation," *J. Physiol. Pathol. Gen.*, **9**, 620 (1907).
- [17] S. Bohte, *Spiking Neural Networks, PhD. Thesis: Universiteit Leiden* (Universiteit Leiden, 2003).
- [18] A. Kleinschmidt, M. Bear, and W. Singer, "Blockade of 'nmda' receptors disrupts experience-dependent plasticity of kitten striate cortex," *Science*, **238**, 355 (1987).
- [19] C. H. Bailey, M. Chen, F. Keller, and E. R. Kandel, "Serotonin-mediated endocytosis of apcam: An early step of learning-related synaptic growth in *Aplysia*," *Science*, **256**, 645 (1992).
- [20] S. J. Kim and D. J. Linden, "Ubiquitous plasticity and memory storage," *Neuron*, **56**, 582 (2007).
- [21] A. Hodgkin and A. Huxley, "A quantitative description of membrane current and application to conduction and excitation in nerve," *Journal of Physiology*, **117**, 500 (1952).
- [22] J. D. Murray, *Mathematical Biology* (New York, NY: Springer Verlag, 1993).
- [23] R. FitzHugh, "Impulses and physiological states in theoretical models of nerve membrane," *Biophysics Journal*, **1**, 445 (1961).

- [24] J. Nagumo, S. Arimoto, and S. Yoshizawa, "An active pulse transmission line simulating nerve axon," *Proceedings IRE*, **50**, 2061 (1962).
- [25] U. Bhalla and J. Bower, "Exploring parameter space in detailed single neuron models: simulations of the mitral and granule cells of the olfactory bulb," *Journal of Neurophysiology*, **69**, 1948 (1993).
- [26] P. C. Bush and T. J. Sejnowski, "Reduced compartmental models of neocortical pyramidal cells," *Journal of Neuroscience Methods*, **46**, 159 (1993).
- [27] D. Hubel and T. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *J. Physiol.*, **160**, 106 (1962).
- [28] D. Hubel and T. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *Journal of Physiology*, **148**, 574 (1959).
- [29] D. Hubel and T. Wiesel, "Receptive fields of cells in striate cortex of very young, visually inexperienced kittens," *Journal of Neurophysiology*, **26**, 994 (1963).
- [30] R. Guttman and L. Hachmeister, "Anode break excitation in space-clamped squid axons," *Biophysical Journal*, **12**, 552 (1972).
- [31] B. R. Jones and S. H. Thompson, "Mechanism of postinhibitory rebound in molluscan neurons," *American Zoologist*, **41**, 1036 (2001).
- [32] E. Salinas and T. Sejnowski, "Correlated neuronal activity and the flow of neural information," *Nature Neuroscience*, **2**, 539 (2001).
- [33] E. Izhikevich, "Resonance and selective communication via bursts in neurons having subthreshold oscillations," *Biosystems*, **67**, 95 (2002).
- [34] E. Izhikevich, "Resonate-and-fire neurons," *Neural Networks*, **14**, 883 (2001).
- [35] G. Murphy and D. Glanzman, "Mediation of classical conditioning in aplysial californica by long term potentiation of sensorimotor synapses," *Science*, **278**, 467 (1997).
- [36] M. Minsky and S. Papert, *Perceptrons* (Cambridge, MA: MIT Press, 1969).
- [37] P. Rowcliffe, J. Feng, and H. Buxton, "Spiking perceptrons," *IEEE Transactions on Neural Networks*, **17**, 803 (2006).
- [38] S. Day and M. Davenport, "Continuous-time temporal back-propagation with adaptive time delays," *IEEE Transactions on Neural Networks*, **4**, 348 (1993).

- [39] D. Lin, J. Dayhoff, and P. Ligomenides, "Trajectory production with the adaptive time-delay neural network," *Neural Networks*, **8**, 447 (1995).
- [40] P. Werbos, "Backpropagation through time: What it does and how to do it," *Proceedings of the IEEE*, **78**, 1550 (1990).
- [41] U. Alon, *An Introduction to Systems Biology: Design Principles of Biological Circuits* (London, UK: Chapman and Hall/CRC, 2007).
- [42] R. FitzHugh, 'Mathematical models of excitation and propagation in nerve' in *Biological Engineering* (H.P. Schwan ed.) (New York: McGraw-Hill, 1969).
- [43] R. FitzHugh, "Thresholds and plateaus in the hodgkin-huxley nerve equations," *The Journal of General Physiology*, **43**, 867 (1960).
- [44] S. Strogatz, *Nonlinear Dynamics and Chaos* (Readings, MA: Addison-Wesley, 1994).
- [45] H. Tuckwell, *Introduction to Theoretical Neurobiology* (Cambridge: Cambridge University, 1988).