# Computing with Spiking Neurons

Thesis by

Timothy S. Frank

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1998

(Defended May 27, 1998)

# Acknowledgements

I owe a great debt to my advisor, Joel Burdick, for his invaluable support and encouragement during my years of graduate school. His fresh ideas inspired much of this work, and with his guidance and insight I was able to overcome numerous obstacles.

Many thanks go to my VLSI lab instructors, Chris Diorio and Jeff Dickerson, for their helpful advice in circuit design and layout techniques. Without their assistance, it is doubtful that any of my neural network circuits would have ever been designed or tested. I also want to thank my lab partner, Ren Wu, who helped me struggle through many tedious experiments. Together, we were usually able to bridge the wide gaps between our theoretical and experimental results.

To my parents, Bud and Barbara Frank, and my sister, Vicki, I want to thank you for your love and encouragement throughout my life. You have taught me the virtues of hard work and dedication when striving towards a goal.

To my wife, Karen, words can not begin to describe the gratitude that I have for you. Your constant love and support carried me through the bad times and gave meaning and purpose to the good times. Your own perseverance through medical school has been a great source of inspiration.

To my daughter, Kristen, who was born three years ago, thank you for giving me a living, breathing neural network to study. When my own research seemed to indicate that the study of neural networks was a hopeless pursuit, you provided a wonderful and joyous counterexample. Your rapid mental development has been nothing short of miraculous, and makes all my efforts at developing intelligent, autonomous machines seem rather pathetic. In the truest sense, you have provided me with the ultimate standard by which my research must be measured.

This work was supported in part by a Department of Defense Fellowship. Without their generous stipend, my attendance in graduate school would have been doubtful.

# Preface

It is usually customary for a Ph.D. thesis to contain a summary of all of the research done by the student while in graduate school. This thesis does not. In fact, all of the research described within this thesis was done within the last two years, while I have actually been a graduate student in the applied physics department for over seven years.

I originally entered Caltech in the fall of 1990 with the idea of doing research in optoelectronics. I was primarily interested in developing optical computers. While taking several optoelectronic courses during my first year at Caltech, I realized that the primary advantage optical computing devices have over their silicon counterparts is in their ability to perform parallel computations. This caused my interest to turn to artificial neural networks. In the meantime, I had joined the research group of Dr. Kerry Vahala, and was starting to do research with quantum dot semiconductor lasers. Quantum dots can be created by applying a high magnetic field to quantum well material; however, we were having difficulty obtaining the quantum well material and a magnet of sufficient strength. Dr. Vahala suggested that I consider another project. Since I was already interested in neural networks, I decided to change my area of research, and I joined the robotics research group led by Dr. Joel Burdick. While his primary focus was not on neural networks, I felt his interest in developing neural controllers for robotics was compatible with my background in engineering.

My initial focus was in learning algorithms for fully-recurrent artificial neural networks. Fully-recurrent networks are believed to be superior over their feed-forward counterparts, since they offer the possibility of learning time-dependent phenomena. Some gradient descent learning algorithms had been developed [81, 82, 83, 96, 108, 112], but they tended to be very slow in converging to a solution (see [5] for a general overview). Also, they only seemed to work for training small networks with simple problems. A typical network might have 10 neurons with 2 outputs, and only after

10,000 learning iterations could the network accurately reproduce a figure 8. While I successfully modified the learning algorithms to significantly reduce the number of learning iterations required (from 10,000 to 500), I was unable to train larger networks on more complex tasks.[1]

As I worked on training recurrent networks, it became apparent that the use of neural networks will always be severely limited as long as they are only simulated on serial computers. The primary advantage of neural networks lies in their parallelism, and it will never be feasible to simulate large networks on serial computers in real time. To learn how to design neural networks in hardware, I enrolled in Dr. Carver Mead's analog VLSI design classes. For my class projects, I designed two neural network circuits, which were sent out to MOSIS and fabricated. With these circuits, I was able to correctly train a three neuron network to duplicate the logic functions of AND, OR, and XOR; however, the adjustable weight values needed to be stored off chip as voltages. With a limited number of pins available for input and output signals, the storage of adjustable weights on chip became an obstacle for creating large networks of neurons. I designed a total of seven chips (including the two class projects). While I made some progress in storing weights on chip, I decided to let Dr. Mead's students finish working out the details (Chris Diorio and Paul Hasler were researching methods for storing voltages by using electron tunneling through silicon dioxide), and return to my research on learning algorithms.

To avoid some of the problems that I had previously encountered with learning algorithms, I decided to take a different approach and follow biology more closely. The research presented in this thesis is a summary of my efforts to duplicate the computational power of biological nervous systems. Unlike the neurons in my previous networks, biological neurons communicate via spikes, not analog signals. For them, learning is a continual process which is accomplished through the formation of new synapses and changes in the strengths of existing ones. Although the connections made by even one neuron are too numerous to map, there is clearly an overall pattern.

[1]To my knowledge, no one has yet been able to train a fully-recurrent network to accurately draw a clover figure.

For many of the different sections within the brain, neurobiologists have been able to determine their functions, and these have not been found to vary significantly between individuals. This indicates that the brain is not just a large fully-recurrent neural network, but is divided into different regions which perform specialized tasks, and much of the topology between these regions is genetically determined. Based on these facts about neurobiology, I entered this new phase of research with two initial assumptions:

1. Biology uses spikes, so network designers should as well. Besides reducing noise, spikes provide an efficient method for learning and storing spatial-temporal patterns.

2. When a network is to accomplish a specific computational task, its topology should be chosen to incorporate as much information about the task as possible. Fully-recurrent neural networks should not be viewed as a general solution.

I started by developing a spiking neuron model, and testing its properties to determine how well it was able to duplicate the functions of real neurons. Although the original model has undergone some minor modifications, it still functions as I first intended: only the neuron's internal voltage and its input neurotransmitter are modeled. Spikes are treated as discrete events with only the times of the spikes conveying any information.

While I realize that this thesis only represents a small fraction of the total work I have done at Caltech, I feel that it indirectly builds upon my past efforts. My previous research has helped point me in many useful directions, and optimized my Caltech experience by providing me with a broad background in the relatively new field of artificial neural networks.

# Abstract

This thesis explores methods for computing with spikes. A spiking neuron model (SNM) is developed, which uses relatively few variables. A neuron's state is completely determined by the amount of neurotransmitter at its input synapses and the time since it last produced a spike. A spike is treated as a discrete event, which triggers the release of neurotransmitter at the neuron's output synapses. Neurotransmitter affects the voltage potentials of postsynaptic neurons.

The SNM is able to duplicate many of the properties of biological neurons, including: latency, refractory periods, and oscillatory spiking behavior, thus indicating that it is sufficiently complex for duplicating many of the computations performed by real neurons. Although the inspiration for the SNM comes from biology, the purpose of this research is to develop better computational devices.

Several single neuron building blocks are designed to perform useful functions, such as: a high gain response, a memory oscillator, a bounded threshold response, and an identity or inverse response. These single neuron building blocks are then used in larger networks to accomplish more complex tasks including: synchronizing input stimuli, recognizing spiking patterns, evaluating Boolean logic expressions, memorizing spike patterns, counting input spikes, multiplexing signals, comparing spike patterns, and recalling an associative memory.

When using the SNM, there are several possible methods for encoding information within a spike train. With synchronous spike patterns, each spike can encode a single bit. The strength of an input stimulus may be retained within the output phase of a spike or logarithmically encoded in the neurotransmitter released at a synapse. And when two sensory neurons receive the same input signal, the time duration of the stimulus can be linearly encoded within their phase differences, while the strength of the input signal is logarithmically encoded in their firing rates.

Learning may also be incorporated into an SNM network. A special feedforward network architecture is presented, in which each neuron has either an inhibitory or excitatory effect on all of the neurons to which it connects. A new learning rule is developed to train this network to respond to any combinations of input spike patterns.

# Contents

# List of Figures

# Chapter 1 Introduction to Artificial Neural Networks

## 1.1 Why Neural Networks?

Essentially, there are two different reasons for studying neural networks. The first is for medical reasons: a better understanding of the nervous system will lead to improvements in the diagnosis and treatment of diseases which afflict neuronal functions. The second reason, which is the motivation behind this research, is to build more robust, autonomous machines with the ability to learn and adapt to different environments. However, before embarking on such a journey, it is useful to first consider what properties are to be extracted from biological organisms.

From a systems standpoint, all parts of an organism are devoted to accomplishing three primary functions: (1) input sensory information; (2) process sensory information in a manner which allows the organism to make decisions about its environment; and (3) respond and interact with the environment. So how does current technology compare with the performance of biological organisms?

From a sensory standpoint, technology well exceeds biology in most cases. Vision systems mounted on satellites are capable of imaging vehicles on the ground, hundreds of miles below their orbit. Telescopes are capable of measuring light over a wide spectrum, instead of only the very narrow range of human vision. And video cameras are available that are smaller than a human eye. For hearing, microphones exist that are capable of monitoring conversations hundreds of feet away. For touch, there are a variety of devices available, including: capacitive sensors, strain gages, piezoelectric devices, and thermocouples for measuring displacement, strain, pressure, and temperature. Thus, the technology exists for accurately duplicating the touch sensing capabilities of humans; however, due to the required signal conditioning circuitry, biology tends to have an edge in sensor density. For smell and taste, biology remains superior. While sensors exist to detect specific chemicals, technology has not yet produced a general purpose "nose" capable of discerning as many different chemicals as the human olfactory system. (Many animals have olfactory systems far superior to humans.) However, there are sensors for detecting many things that biology can not; e.g., carbon monoxide and radon gas are two common sensors commercially available. For taste, humans actually have only four types of sensors: sweet, sour, bitter, and salty. Since this sense is primarily associated with food consumption, there has not been a serious attempt to duplicate it. Overall, technology is clearly capable of providing adequate sensor inputs to any autonomous machines, which use vision and audition as their dominant senses.

Technology also tends to exceed biology in its ability to interact with the environment. Humans are only capable of producing movement, which occurs through the control of muscles. Even speech is just the result of coordinated contractions in the muscles which control of the diaphragm, vocal chords, tongue, and jaw. To produce movement, technology has developed gears, motors, and actuators, and their performance exceeds biology in strength, endurance, and precision. However, biology excels in muscle density, and is capable of producing a variety of moments within structures much too small to contain even a single actuator; e.g., the legs on an ant, or wings on a fruit fly. Although smaller structures are being developed for controlling movement, any machine capable of replicating the wide variety of movements possible in a biological organism will need to be relatively large and require a great deal of clever engineering. Thus, it may be easier to build a mechanical elephant than a mechanical ant! But one advantage technology has over biology is in the variety of ways it is capable of interacting with the environment. Electrical signals can be directly output to other machines or converted into sound or light.

Then what is the limiting factor in the design and construction of useful autonomous machines which are capable of operating within an unknown and changing environment? The answer appears to lie within the elusive property called intelligence, which allows an organism to rapidly process its sensory information and make decisions about the environment. A biological organism has the ability to learn from its past, and the more adept it is, the more intelligent it is considered to be. But how does intelligence work? Biologists have shed some light on the problem, and determined that intelligence is a property of the nervous system, which is composed of vast numbers of interconnected neurons. The intelligence of an organism is largely dependent upon its number of neurons. Thus, artificial neural networks are an attempt to duplicate the basic machinery of biological intelligence.

Computers are still unable to fully duplicate intelligence. Currently, personal computers are capable of performing 400 million operations per second. In comparison, it usually takes between 20-40 ms for one brain cell to fire and communicate its signal to another neuron (25 to 50 operations per second). But biology has a computational advantage with its parallel structure, which allows vast numbers of neurons to process information simultaneously. The information flows freely between connected neurons, with no system clock. Thus, conventional computers are far superior to humans in serial tasks, such as arithmetic, but the human brain is superior in tasks that require parallel processing, such as visual pattern recognition. For a computer to recognize an object in a scene, it must analyze the picture one pixel at a time, while the human visual system is able to process the entire scene simultaneously.

Unlike computers, neural networks encode information in a distributed fashion. This allows them to use noisy and incomplete data and still make "good" decisions quickly. While computers follow the commands in programs, neural networks do not make precise calculations, and it even is difficult to reduce their processing into an algorithm. If there is an error in a computer program, the computer will produce the same incorrect output each time, but a neural network may be able to learn from its past experiences and produce new results. The process of self modification gives neural networks their ability to adapt to new environments, and it has been the primary motivation for research into building more biologically based computational devices.

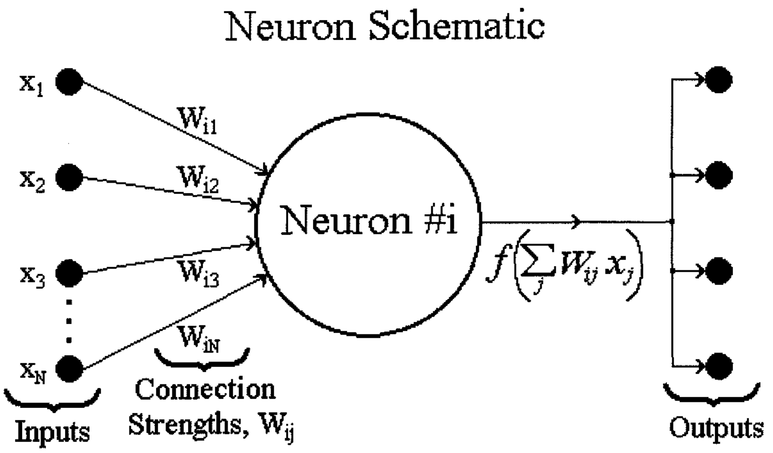


Figure 1.1: **Schematic of a Typical Artificial Neuron.** On the left are the multiple inputs into the neuron, which come from other neurons or external stimuli. Each of the interconnections between the inputs and the neuron has an associated connection strength, $W_{ij}$, where $i$ is the index of the neuron receiving the signal and $j$ is the index of the neuron sending the signal. The neuron performs a weighted sum on the inputs and uses a nonlinear function, $f\left(\sum_j W_{ij}x_j\right)$, to compute its output. The same calculated result is sent to each of the target neurons on the right.

## 1.2 Types of Artificial Neural Networks

There are many different types of artificial neural networks. This section briefly reviews some of the more popular ones to illustrate how they differ from the spiking neuron model developed later in this thesis. In general, all neural networks can be described in terms of: transfer function (discrete or analog), topology (feedforward or recurrent), implementation method (software or hardware), update method (synchronous or asynchronous), behavior (deterministic or stochastic), output signal format (continuous or spiking), and learning methodology (supervised or unsupervised).

The typical artificial neural network is composed of many processing elements called neurons. Each neuron may receive multiple input signals, from other neurons or external stimuli, and produces a single output, which may be sent to numerous other neurons. Each interconnection has an associated connection strength or weight. The neuron performs a weighted sum on the inputs and uses a nonlinear function to compute its output. Figure 1.1 depicts an artificial neuron.

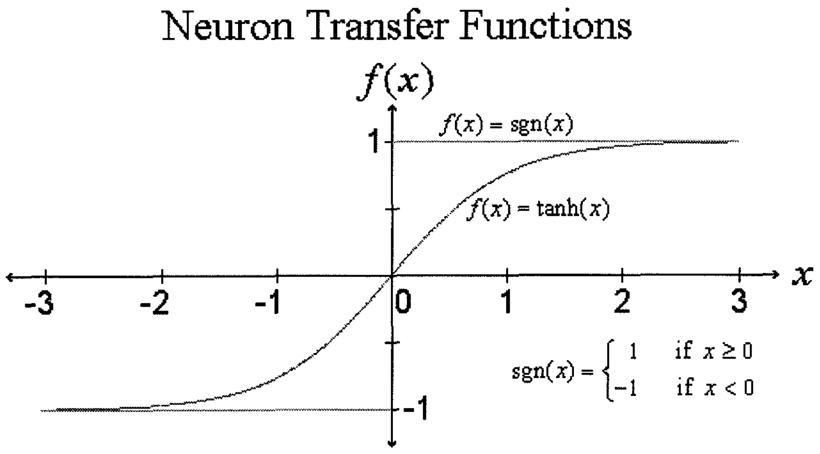


Figure 1.2: **Neuron Transfer Functions.** Two commonly used transfer functions are the binary threshold and the sigmoid. The sgn $(x)$ function is an example of a binary threshold, and the tanh $(x)$ function is an example of a sigmoid.

A neuron's transfer function may be either discrete or analog. An example of a discrete transfer function is the binary threshold, in which a neuron outputs -1 or 1 depending upon whether the input signal is above a threshold level (sometimes 0 is used instead of -1). Historically, this was the first type of artifical neuron used, and it was first proposed in 1943 by McCulloch and Pitts (see [71]). They proved that a synchronous assembly of such neurons with suitably chosen weights can duplicate any computation that can be performed by an ordinary digital computer [42, p. 3]. But currently, the most commonly used transfer function is a sigmoidal analog function. A sigmoid is a bounded differential real function that has a positive derivative everywhere. Its value asymptotically approaches finite upper and lower limits. The central portion of the sigmoid is assumed to be roughly linear, with the average slope of the central portion called the gain [39, p. 106]. An example of a sigmoid is the tanh $(x)$ function. Figure 1.2 depicts these two types of transfer functions.

Often the neurons are arranged in layers, with all of the inputs into a neuron coming from neurons in the previous layer. When the neurons can be arranged in this pattern, the network is considered to have a feedforward topology. Feedforward neural networks, which are also called perceptrons, represent the oldest neural network architecture, and were first developed by Rosenblatt in 1957 (see [94]). The neurons in the first layer receive their inputs from external stimuli, and are called input neurons.

The neurons in the last layer produce the "useful" output signals, and are called output neurons. All of the neurons that do not directly interact with the external environment through input or output signals are called hidden neurons. They usually act as "feature detectors" and respond to particular patterns that may appear in the input layer [17, p. 8]. Sometimes there are also inhibitory connections between the neurons within a layer. The resulting competition between the neurons limits the number that can respond to a particular set of input signals. Such a network can be designed to implement a winner-take-all function, where only the neuron with the maximum weighted input signal has a positive output signal (see [42, p. 219]).

When the neurons can not be arranged in layers, the network is considered to have a recurrent topology. If each neuron connects with all of the others, then the network is fully-recurrent. Unlike feedforward networks, these networks can be used to process and store temporal information; e.g., they can produce an oscillatory or chaotic output even when the inputs are constant [82]. The disadvantage of using recurrent networks is that their interconnections make them very difficult to analyze. Also, most learning algorithms tend to be much slower in recurrent networks [39, p. 189]. One special type of recurrent network is the Hopfield network, which is used for associative memories. The weights are set a priori so that the network converges to one of the stored patterns.

While there has been progress in developing electronic hardware, such as VLSI circuits for artifical neural networks (e.g., [72, 73]), most implementations have been done with standard serial computers simulating the neuron dynamics in software [17, p. 17]. One of the main problems with making neural network chips is that they require a lot of interconnections. The space taken up by the connections usually is the limiting factor in the size of the network [42, p. 9]. Another major problem is storing the adjustable parameters within the chip. (Storing the parameters off-chip usually requires more pins than there are available.) All of the neuron parameters must be represented as currents or voltages, but it is very difficult to accurately store analog signals on chip. One alternative is to use digital-to-analog (D/A) converters, but they require a considerable amount of space – especially if each parameter requires one. Research into the possibility of using floating gates to store analog voltages on chip is currently in progress (see [18, 19, 36, 78]). An alternative to building neural networks on silicon chips is to use optical computers, but current implementations are limited and expensive [42, p. 9]. In the long term, efficient hardware is crucial to take full advantage of the capabilities of neural networks, and there is growing activity in this area; however, it is beyond the scope of this thesis.

When networks are simulated in software, the neuron dynamics can be updated either synchronously or asynchronously. Most algorithms use the synchronous method, with all of the neurons being updated simultaneously at each simulation time step [17, p. 42]. A few algorithms update the dynamics asynchronously, with the neuron outputs being calculated one at a time. (This was first purposed by Hopfield for his associative memory networks [45].) In this method, a neuron is randomly chosen and its output is updated based on its input signals. The new output signal is then available to be used as an input for the next randomly selected neuron.

Networks that use an asynchronous update method often have "stochastic" neurons. In this case, the neuron's transfer function does not actually represent the output of a neuron, but rather, the probability of the output assuming a binary value of 1 or -1. Typically, a stochastic neuron uses a sigmoidal transfer function which is bounded between 0 and 1, e.g., $f(x) = 0.5 \cdot \left(\tanh\left(\frac{x}{T}\right) + 1\right)$.[1] The value of the sigmoid, which is a function of the inputs, represents the probability of a neuron's output being 1. Many networks with stochastic neurons are used for storing associative memories, where the neuron dynamics cause the network to converge to a stored memory. These stable states (attractors) represent the global minima of an energy function. The use of stochastic units actually helps prevent the system from getting trapped in spurious local minima [42, p. 33].

Most neuron models continuously output a signal, which depends only upon their

[1]The parameter of $T$ controls the gain of the sigmoid, and is referred to as the "temperature" since it is analogous to controlling the thermal fluctuations due to noise in a stochastic physical system. The terminology comes from the similarity between the activation probability of stochastic neurons and the Ising model for spins (magnetic moments) within magnetic materials, where the actual temperature does influence the probability of an atom changing its spin. (See [42, pp. 25-32] for more details.)

inputs; however, these models do not reflect the complex properties of real neurons. Consequently, a few researchers have turned to models which output spikes or pulses. Usually, the time between pulses is inversely related to the strength of the signal. When the neuron's output is a spike train, the neural transfer function must also include a dependence upon the neuron's internal state; i.e., after a neuron produces a spike, there must be time lag before another spike can be produced (regardless of the input). Thus, the time of the next output spike must include a dependence upon the time of the previous spike. Because this complicates the neuron dynamics, most artifical neural networks continue to be based on neurons with continuous outputs. However, spiking neuron models are popular with neurobiologists, who have developed sophisticated dynamical systems for replicating action potentials. But these models are usually used for the purpose of studying biology and are too complex to be used within a large artificial neural network as a *computational device.* Most spiking models can be classified as being either of the integrate-and-fire or realistic channel-based type [101]. With the integrate-and-fire models, inputs are integrated until the voltage reaches a threshold level, whereupon a spike is produced, resetting the neuron. The channel based models describe the actual ionic currents within a biological neuron, which lead to the generation and propagation of an action potential. (Section 3.2 presents a novel spiking neuron model and Section 3.6 compares it to other popular spiking models.)

The usefulness of neural networks depends upon their ability to perform desired tasks. While there are some problems where the neuron parameters can be chosen a priori, these problems are the exception rather than the rule. Most problems require the network to learn the desired computation by making iterative adjustments to the neuron parameters. (In many neuron models, the interconnection weights are the only parameters to learn.) There are two methods for training networks: supervised and unsupervised learning. In supervised learning (sometimes called learning with a teacher), the parameter adjustments are made by comparing the network's output with known correct answers. The most common form of supervised learning is back-propagation, which uses gradient descent to adjust the parameters in the direction that reduces the error in the output signal. In unsupervised learning, the network is expected to find correlations between inputs, and produce output signals which divide the input signals into different categories. This method is used when the learning goal is not defined in terms of specific correct examples [42, p. 10].

## 1.3 Why Use Spiking Neurons?

Much of the current research in artificial neural networks has focused on the use of neurons with continuous analog signals. However, it has been well established that biological neurons display spiking behavior. Furthermore, all of the spikes produced by real neurons have nearly identical shapes and sizes. In fact, there is relatively little variation in neuronal spikes even between species [61, p. 4]. This has led most researchers to conclude that information is not carried within individual spikes, but within the spike frequency.[2] Indeed, much of the justification for using continuous analog signals in artificial neural networks is based on the tacit assumption that the output signal of a neuron primarily depends upon its mean spiking frequency, which can be represented as an analog signal (see [46] for example). This raises some obvious questions about the time window over which the average frequencies are computed. With small time windows, neurons become more sensitive to noise, but with larger time windows an organism has difficulty responding to rapidly changing inputs. To overcome this problem, some neurobiologists have suggested that at least some information is carried within the relative phases between spike trains (e.g., [7, 27]). Obviously, a coding scheme based on single spikes or interspike intervals yields a much greater information capacity than one using an average spike rate [21], which is one of the primary motivating assumptions for this thesis. In support of a spike encoding scheme, various groups of neurons have been observed displaying oscillatory patterns (e.g., [30]). It has been suggested that neurons with similar oscillations can

[2]Adrain [2] was one of the first to observe that the output spike frequency of a nerve cell is a measure of the stimulus intensity, with stronger stimuli producing higher frequencies and better maintaining the cell's firing.

be used to solve the labeling problem [98]; i.e., how are the observed properties of an object linked together within the brain so that the object is perceived as a single entity? Other researchers have even suggested that groups of oscillating neurons may even form the basis of consciousness [57].

Research in *artificial* neural networks has often been approached from the view point of artificial intelligence with the primary goal of building better computing devices. Conversely, computational neuroscientists investigate *biological* neurons and use simulations to understand how different neural systems function (e.g., [9, 59]). Obviously, the findings from both approaches are mutually beneficial. The research presented in this thesis attempts to further narrow this gap, by using some of the known information about neurobiology to construct better computational devices composed of spiking neuron-like models. While much of this research was motivated by biology, the ultimate goal was not to contribute to a better understand of neurobiology. Thus, assumptions are occasionally made that have not been experimentally justified in biological systems. But it is hoped that some of the computational ideas presented here may offer reasonable paradigms for biological systems, which stimulate further investigation by neurobiologists.

## 1.4 Overview of Thesis

Since the bulk of this thesis uses biologically motivated neurons with spiking dynamics, Chapter 2 contains a brief description of real neurons, before Chapter 3 develops the Spiking Neuron Model (SNM). Some modifications to the model are discussed to further mimic the biological functions of presynaptic inhibition and synaptic clusters.

In Chapter 4, several properties of the SNM are described and compared with biology, including: latency, refractory periods, endogenous spiking behaviors, and adaptation of firing rate. Chapter 5 then discusses some methods for simulating the SNM, which build upon the neurotransmitter properties presented in Chapter 4.

Chapter 6 is the first chapter showing how the SNM can be used as a computational tool. This chapter presents some single neuron building blocks, which are useful in constructing large networks. The neuron parameters are chosen to perform specific functions, such as: a high gain response, a memory oscillator, a bounded threshold response (spikes are only output when the input is between a lower and *upper* limit), and an identity or inverse response. Each neuron is designed to perform a specific function, and there is no learning involved.

Chapter 7 describes how the SNM can be used in networks to accomplish more complex tasks. It begins by describing a stimulus detector, which senses when there is a large stimulus signal arriving from several sources and synchronizes their spikes. It is used as a preprocessor for other networks which require synchronous inputs. Next, several pattern recognizing networks are developed for both spatial and temporal spike trains. The chapter then shows how a single neuron with synchronized inputs can be used to evaluate any Boolean logic expression. A variety of networks are developed for tasks including: memorizing spiking patterns, counting spikes, multiplexing input signals, comparing spike patterns, and recalling an associative memory.

Chapter 8 discusses some of the different ways that the SNM can encode information in spike trains. For the networks presented in Chapter 7, the meaning of each spike depends upon the task being performed. However, the SNM can also be designed to encode information in either its output spike phase or frequency.

Chapter 9 shows how learning may be incorporated into the SNM. A special feed-forward network architecture is presented, in which each neuron is either inhibitory or excitatory. A new learning rule is developed, which can be used to train this network to solve the pattern recognition and logic tasks discussed in Chapter 7.

The thesis then concludes with Chapter 10, which summarizes the research presented, highlights its significant contributions, discusses some of its computational and biological implications, and outlines some of the possible advantages that may be achieved by implementing the SNM in VLSI hardware.

# Chapter 2 Biological Neurons

## 2.1 Introduction

The existence of a nervous system is a primary distinguishing feature of animals. Its fundamental task is to gather information about the environment, process and store this information, and generate behavior. It can be studied at a number of levels of organization. Biochemists explore the characteristics of molecules within a single neuron. Physiologists study the properties of groups of neurons that are functionally related. Behaviorists investigate patterns in learning and behavior. And finally, connectionists attempt to extract the "intelligent" abilities of a nervous system.

Research into neural computation can be roughly divided into two areas: one is interested in how neural models can illuminate the properties of biological systems, and the other is interested in the functionality that neural computing can provide. This research is based on the latter approach; however, the spiking neuron model extracts many of the salient features of biological neurons. To facilitate the comparisons between biology and the spiking neuron model, it is useful to first describe some of the important properties of a nervous system, and its basic structural unit, the neuron. The Spiking Neuron Model (SNM) presented in Chapter 3 is a simplified mathematical model for the neuronal functions discussed here, and many of the model parameters have a plausible physiological interpretation. Since only the most basic neuron properties are discussed in this chapter, readers wishing to learn more about biological neurons may want to consult [53], [61], or [65].

## 2.2 The Nervous System

Although the entire nervous system is interrelated, it can be divided between the peripheral and the central nervous systems. The peripheral nervous system is composed of the nerves that carry information to and from the central nervous system, and it can be further divided into the somatic system, which carries messages inward from the sense organs and outward to the skeletal muscles, and the autonomic system, which carries information to and from organs, glands, and other muscles such as those of the heart and digestive system. The central nervous system includes the brain and spinal cord, which can also be subdivided into smaller regions according to their appearance and organization. The largest part of the brain is the cerebrum, which contains 70% of the neurons in the central nervous system. The cerebral cortex is the outer layer of the cerebrum, and is responsible for the higher functions of the nervous system, including intellectual processes [20, p. 34, 44].

The brain is the primary organ responsible for interpreting sensory input, coordinating bodily activities, and generating thought. Although the human brain contains approximately a hundred billion ($10^{11}$) neurons, its processing is broken down into groups of neurons that act as functional units. The neurons are densely packed in a highly interwoven mass of tissue, with each neuron having as many as 200,000 interconnections to other neurons (1,000 to 10,000 interconnections is typical). The number of interconnections grows considerably as a human develops from a fetus to an adult, while the actual number of neurons decreases.

The cells within the brain can be divided into two distinct classes: neurons and glia. Glia, which comprise over 90% of the cells in the brain, are not believed to be directly involved in signaling information. They fill the spaces between neurons and provide the necessary structural and nutritional support. Some types of glia form a myelin sheath surrounding the axons of some neurons. It acts as an insulating cover, which aids in signal transmission and allows for faster propagation of action potentials. Except in the case of myelination, the role of glia in the nervous system is not well understood. It is thought that during development, certain classes of glial cells guide the migration of neurons and direct the outgrowth of axons. Glia also appear to be involved removing the neurotransmitters released by neurons during synaptic transmission [53, p. 22].

## 2.3 Neuron Structure

Neurons come in many different sizes and shapes and are usually tailored for a specific function within the nervous system, but most neurons are polarized with inputs arriving at the dendrites and outputs leaving at the axon [50, pp. 1-2].

Typically, a neuron has three major regions: the axon, the soma or cell body, and the dendrites (see Figure 2.1). The axon, which acts as the output mechanism for the neuron, conducts signals away from the soma. The outgoing signals, called action potentials, are usually initiated at the axon hillock, located where the axon connects to the soma. Although there is only one axon for each cell, it can branch tremendously, sending signals to different locations. While providing structural support to the cell, the soma contains the nucleus and other structures required to manufacture the enzymes and molecules needed by the neuron. The dendrites receive incoming signals from other neurons. They can grow from one or more different locations on a cell body, and, like axons, can be densely branched. Their sides may have dendritic spines, which are protruding structures important for receiving incoming signals. The diameters of the dendritic branches determine how incoming signals are summed and processed by the neuron. A signal generated at the tip of a dendrite will excite the axon hillock differently than a signal generated near the soma [17, p. 126].

Surrounding the entire cell is a membrane, whose structure and properties affect the neuron's functions. The membrane contains special channels which allow certain ions to pass through in a controlled way. The neuron integrates the incoming signals and propagates action potentials down the axon by regulating the flow of ions through its membrane [17, p. 137].

Neurons transmit information through specialized contact zones, called synapses. Synapses can be either electrical or chemical. Electrical synapses, which are also called gap junctions, consist of specialized proteins that form intercellular bridging pores through which current can flow from one neuron to the other. Chemical synapses consist of synaptic vesicles in the presynaptic neuron (sending neuron) which release chemical messengers, called neurotransmitters, into the gap between the neurons.

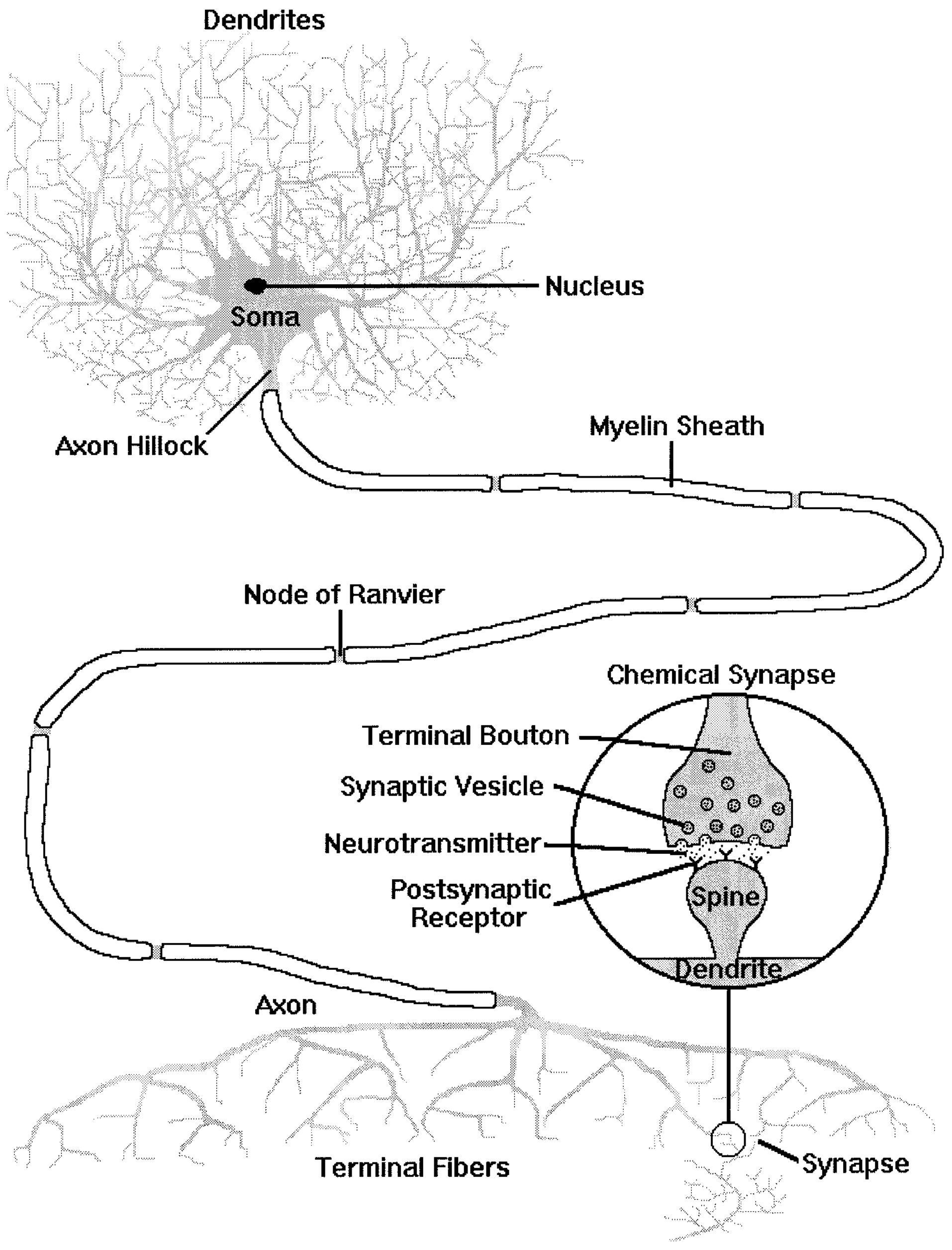


Figure 2.1: **Structure of a Typical Neuron.** Neurons convey information by electrical and chemical signals. Electrical signals propagate from the axon hillock to the terminal fibers in the form of action potentials. When an action potential arrives, it triggers the secretion of neurotransmitters from the terminal boutons. Neurotransmitters bind to receptors and produce changes in the potential of the postsynaptic neuron. If a sufficiently large change in the potential is generated, an action potential is produced in the next neuron.

The synaptic vesicles are stored at the tips of the axon, which are called terminal boutons. The spines along the dendrite of the postsynaptic neuron (receiving neuron) contain neurotransmitter receptors. The actual patterns of synaptic connections in the nervous system are extremely complex. Some neurons make synaptic contacts to nearby neurons, while others have long axons that form synapses on neurons up to a meter away.

## 2.4 Neural Transmissions

While individual neurons have extraordinary diversity in their morphology and biochemical properties, the nervous system relies on only a few basic principles for transmitting signals. Neural signals are transmitted electrically in the interior of a neuron by ionic currents through the membrane, which cause changes in the transmembrane voltage. These ionic currents are controlled by ionic specific channels, which are either open or closed. When a channel opens, the ionic concentration gradient across the membrane drives the flow of ions through the channel. Normally, the interior of a neuron is negatively charged with respect to its surrounding medium, with a resting potential, $V_r$, of approximately -70 millivolts. This potential is a consequence of the different interior and exterior ionic concentrations. The concentrations are maintained by ion pumps whose energy is derived from the hydrolysis of ATP molecules. There are actually four different ions involved in transmembrane currents: sodium ($Na^+$), potassium ($K^+$), calcium ($Ca^{2+}$), and chloride ($Cl^-$) [50, pp. 3-4].

When a neuron's dendrites are stimulated, it triggers a sequence of voltage and time varying conductance changes in the cell membrane, resulting in a voltage change at the soma. The dendrites are stimulated by neighboring neurons through neurotransmitters, which can be excitatory or inhibitory. When a dendrite receives excitatory neurotransmitters, channels in the membrane open through which positively charged ions enter, causing the potential difference between the interior and exterior of the cell to diminish in a process known as depolarization. If the interior of the cell becomes sufficiently depolarized (typically the potential difference across the membrane must increase above -40 mV [35, p. 16]), an action potential is generated at the axon hillock and propagates down the axon.

Action potentials are produced in the axon hillock when special channels in the membrane open allowing sodium ions ($Na^+$) to rush in. The front part of the axon becomes slightly positive relative to the external medium. This causes the channels in the immediate vicinity to open so that the potassium ions can rush out and restore the original resting potential of -70 mV. But before the potential is restored, the adjacent portions of the axon membrane are affected, causing the entire process to repeat itself a little farther down the axon. Thus, the rapid ion exchange moves down the length of the axon, with the sudden shift from negative to positive and back to negative resembling a spike in potential. When a nerve impulse is generated in this manner, it will travel relatively long distances without any distortion or loss of strength, but the action potential is an all-or-nothing response [20, pp. 39-40].

The conduction velocity of the action potential primarily depends upon the rate at which the membrane capacitance ahead of the active region is discharged to threshold by the spread of positively charged ions. Because the ions are able to spread more quickly within larger fibers, they tend to have higher propagation velocities than smaller fibers. In theory, the propagation velocity of an action potential varies directly with the square root of the fiber diameter [61, p. 178].

After a nerve impulse passes through a section of the axon, the membrane gradually recovers its original properties and regenerates its resting potential over a period of several milliseconds. During this recovery period the neuron remains incapable of further excitation, and the neuron is said to be in its "refractory period." After the recovery is completed, the neuron returns to its resting state and can produce another spike. Because the depolarized sections can not immediately become active again, the pulse of electrical activity can only propagate in one direction: away from the cell body. All action potentials have the same shape and magnitude. Thus, neurons are believed to encode the intensity of their signals in the spiking frequency, which can range from about 1 to 100 spikes per second [77, p. 4]. Of course, the time interval between spikes must be longer than the neuron's refractory period.

In addition to the diversity among neurons in their firing frequencies and propagation velocities, the intrinsic patterns for action potential firing vary greatly. While some neurons have a steady unchanging resting potential in the absence of external stimulation, others generate a variety of endogenous spiking patterns. This includes "pacing" or "beating" neurons, which fire repetitively at a constant frequency. Although external stimulation can change the firing rate or inhibit it altogether, the mechanisms that drive their repetitive firing are intrinsic to the cell and do not require any external stimuli. Some neurons that fire spontaneously do not produce action potentials at fixed regular intervals but instead generate bursts of spikes that are separated by periods of hyperpolarization in the cell's potential. Such neurons are called "bursting" neurons and are used to generate rhythmic behaviors, like breathing, walking, and chewing [65, p. 47].

While the generation of action potentials is the primary method used by neurons to transmit information, it is not the only method. Ions flowing into the dendrites and soma that do not stimulate the neuron sufficiently to generate a spike may nonetheless enable the cell to pass information to nearby neurons. The voltage levels between the resting potential and the potential required to generate a spike are called graded potentials. Unlike action potentials, graded potentials rapidly lose their electrical strength with distance. Thus, action potentials are required for long-distance transmission, while graded potentials can be used for local communication. Many of the neurons that rely heavily on graded potentials in order to exchange information don't even have axons [20, pp. 40-41]. These neurons usually communicate via gap junctions, which provide a low-resistance electrical interconnection between neurons.[1]

[1]It is believed that gap junctions may be a primitive form of a synapse since they are much more common in invertebrates than they are in mammals [17, p. 149]. They are of less interest here because they have a much lower degree of adjustability [77, p. 5]. Consequently, neither graded potentials nor gap junctions are considered in the spiking neuron model presented in the next chapter.

## 2.5 Synaptic Transmissions

When an action potential travels down the axon, it must come to a halt at the synapses, since there is not a conducting bridge to the next neuron or muscle fiber. The signal is transmitted across the synaptic gap (also called the synaptic cleft) by neurotransmitters, which are liberated in tiny amounts from vesicles contained within the terminal bouton. The brain uses a variety of neurotransmitters, and several different neurotransmitters can coexist within the same synapse. Their release is initiated by the influx of calcium ions ($Ca^{2+}$) into the presynaptic axon during the depolarization caused by the flow of sodium ions ($Na^{+}$). The amount of neurotransmitter released varies widely between synapses. The molecules diffuse across the synaptic gap and reach the postsynaptic neuron (or muscle fiber) within approximately 0.5 ms. When the neurotransmitters arrive at special receptors, the conductance of the postsynaptic membrane is modified for certain ions, which results in a polarization (inhibitory) or depolarization (excitatory) of the local postsynaptic potential. The neurotransmitters are then quickly broken down by enzymes. While the rate at which the neurotransmitter is released from a terminal bouton is increased enormously with the arrival of an action potential, neurotransmitter is also randomly emitted at a relatively low rate. The random release of neurotransmitter results in small depolarization potentials in the postsynaptic neuron, and can cause the spontaneous generation of an action potential [77, pp. 5-6].

Inhibitory synapses often connect onto other presynaptic axons, and inhibit their ability to release neurotransmitters. This is referred to as presynaptic inhibition. Also, there is some evidence that all the synaptic endings of an axon are either excitatory or inhibitory (Dale's law). Thus, an entire neuron may be referred to as being excitatory or inhibitory.[2] The two types of synapses have some significant

[2]The spiking neuron model presented in Chapter 3 allows neurons to form both excitatory and inhibitory synapses. This departure from real neurons is based on the underlying assumption that biology needs to distinguish neurons as being excitatory or inhibitory due to the different chemical and structural constraints of each type. Obviously, any network with neurons which are capable of both synaptic types should at least have the same computational ability as a network composed of neurons which are only capable of forming excitatory or inhibitory synapses.

structural differences, with excitatory synapses changing the conductance of $Na^+$ and $K^+$ and inhibitory synapses changing the conductance of $Cl^-$ [77, p. 5].

In general, biology use a wide variety of synapses. Often, they differ in the cell parts that participate in the synaptic connection. While the typical synapse is axo-dendritic, where a contact is made from an axon to a dendrite, other types of connections exists, including: axon-somatic (axon to soma); axo-axonic (axon to axon); somato-somatic (soma to soma); and dendro-dendritic (dendrite to dendrite). Often synapses are situated near each other in tightly grouped clusters, called synaptic glomeruli. A connection between two neurons may even be a mixed junction, which has both a chemical synapse and a gap junction. Another surprisingly common type is a reciprocal junction, in which two neurons have synapses onto each other [17, pp. 150-151].

When a postsynaptic neuron receives an excitatory signal at one of its synapses, the neuron can, in principle, be inspired to fire; however, this is rarely the case, especially if the synapse is located at the outer end of a dendrite. Most neurons receive input from thousands of others. The body of the neuron acts as a "summing" device. The effects from an incoming signal tend to decay with a characteristic time of 5-10 ms. But if several signals arrive within this time period, their excitatory effects accumulate, and when the total magnitude of the depolarization exceeds the critical threshold, the neuron fires. Notice that an input neuron with a high firing rate will have a larger effect than one with a low firing rate, which is consistent with having either the *average* firing rate or the inverse of the time between spikes express the intensity of a signal [77, pp. 5-6].

In addition to the repetition rate of the arriving spikes, the influence each synapse has on a neuron depends upon the inherent strength of its depolarizing effect, and its location with respect to the cell body [77, p. 6]. Thus, while the body of the neuron sums the incoming signals, each of the synapses can be thought of as multiplying one of the input signals by a "weight value," which indicates its relative significance in triggering an action potential. There is a great deal of evidence that synaptic strengths are adjusted over time, which is believed to play a dominant role in storing memories and learning. While it is not completely clear how the brain makes the appropriate synaptic adjustments, it has been theorized that an active synapse, which repeatedly triggers the activation of its postsynaptic neuron, will grow in strength, while others will gradually weaken (Hebb's rule [37]).

While neurons transmit information in the form of spikes, the information content of a neuron is determined by its function; i.e., the quality or meaning of a signal depends upon the origins and destination of the nerve's connections. Activity from the various sensory inputs stimulate different regions in the brain, which are strictly determined by the neural connections. Thus, spikes convey information about the intensity of a stimulus and not its quality [61, p. 6].

## 2.6 Summary of Biological Neurons

Nearly all neuron have the same underlying principles for transmitting signals. The Spiking Neuron Model presented in the next chapter is based on a typical neuron, which produces action potentials when its membrane voltage exceeds a threshold level. The spikes travel down the axon to synapses, which connect the neuron to other neurons or muscles. When a spike arrives at a synapse, it causes the presynaptic neuron to release neurotransmitter into the synaptic gap. As the neurotransmitter diffuses across the gap, it is then absorbed by the postsynaptic neuron. Depending upon the neurotransmitter used, the effect on the postsynaptic neuron may be either excitatory or inhibitory. Excitatory signals increase the membrane voltage toward threshold, while inhibitory signals decrease the membrane voltage. The voltage change due to each incoming spike depends upon the strength of the connection, which is determined by the amount of neurotransmitter released by the presynaptic neuron and the synapse's location on the postsynaptic neuron. Also, the effect on the postsynaptic neuron is delayed from the presynaptic neuron's initial spike generation, primarily due to the propagation time required by the axon. When a presynaptic neuron is producing closely spaced spikes, there can be a buildup of neurotransmitter within the synaptic gap, which causes the voltage change on the postsynaptic neuron to be larger than that from a single input spike. Most neurons receive many excitatory and inhibitory inputs from other neurons and in turn supply many others. Some neurons produce an endogenous spike pattern even with no input stimulation. Other neurons have sensory receptors that respond to an external physical stimulus. Stronger stimuli usually produce higher firing frequencies, but the maximum frequency is limited by a neuron's refractory period. The absolute refractory period is the time immediately after a spike is produced when the neuron is unable to produce another output spike, and the relative refractory period is the time after the absolute period when the neuron is able to produce another spike, but its effective threshold level is elevated above normal. The next chapter presents a simple mathematical model to account for all of these phenomena.

# Chapter 3 Spiking Neuron Model

## 3.1 Introduction

This chapter presents the Spiking Neuron Model (SNM) which is able to capture many of the dynamic properties observed in biological neurons. Each neuron receives input signals from other neurons and/or external stimuli. The neurons that receive an external input signal are referred to as sensory neurons. The potential voltage of a neuron represents its internal state. When a neuron produces a spike, it triggers the release of neurotransmitter at its output synapses. Spikes carry no information in themselves; i.e., the spikes' widths and heights are inconsequential. It is the neurotransmitter within the synapses that carries information between neurons.

When a presynaptic neuron fires, the released neurotransmitter affects the voltage potential of the postsynaptic neurons. Associated with each synapse are a time delay, two weight values, and a time constant. The time delay sets the time from the initial spike production in the presynaptic neuron until its effects are felt by the postsynaptic neuron. The weight values set the strength of the presynaptic neuron's influence on the postsynaptic neuron's potential voltage. (One weight value corresponds to the amount of neurotransmitter released by the presynaptic neuron, and the other weight value corresponds to the number of neurotransmitter receptors on the postsynaptic neuron.) The time constant determines the length of time the presynaptic spike is able to influence the postsynaptic neuron's potential voltage.

In addition to its input synapses, the behavior of the SNM also depends upon its resting potential, threshold level, and refractory time constant. The resting potential determines the neuron's potential when there are no input signals, while its threshold value sets the level that its potential needs to exceed before a spike can be produced. The refractory time constant determines the time between output spikes. (The SNM has both an absolute and relative refractory period – see Section 4.3.3.)

The SNM does not model the dendrites and axon of a real neuron. Instead, all of their affects are lumped together and included within the synaptic parameters, which account for the propagation time delay and time constants. The SNM is completely deterministic: the release of neurotransmitter at each synapse only occurs when triggered by a spike; the amount of the neurotransmitter released with each spike is set by the synaptic parameters; and a neuron fires when its potential reaches threshold. While real neurons seem to be somewhat non-deterministic in nature, it is unclear as to the extent of their probabilistic behavior, and whether it offers a computational advantage or is just a consequence of the neurobiological constraints. But the use of probabilistic neurons is difficult to implement in hardware, and it tends to complicate the analysis and makes computations intractable. Notice also that deterministic systems can produce chaotic behavior. Thus, the deterministic nature of the SNM is not believed to significantly weaken its computational abilities.

## 3.2 Dynamic Equations for the SNM

For the variables used, the subscripts refer to neuron labels, while the superscripts refer to spike numbers. The amount of neurotransmitter available in the $j^{\text{th}}$ synapse of $n_i$ (neuron #i) is denoted by $T_{ij} \cdot U_{ij}(t)$, while the potential voltage of $n_i$ is given by $V_i(t)$. When this potential voltage rises above the neuron's threshold voltage, $\Theta_i$, the neuron produces a spike. The SNM equations are given by:

$$\begin{aligned} V_i(t) &= \tanh^2\left(\frac{t - t_i^K}{\tau_i}\right) \cdot \left\{ R_{i0} + \sum_{j=1}^{N_i} R_{ij} \cdot \tanh\left(\frac{T_{ij}}{R_{ij}} \cdot U_{ij}(t)\right) \right. \\ &\qquad \left. + \sum_{x=1}^{P_i} R_{ix} \cdot \tanh\left(T_{ix} \cdot I_{ix}(t)\right) \right\} \end{aligned} \tag{3.1}$$

$$\text{where:} \quad U_{ij}(t) = \sum_{t_m^k \le t - d_{ij}} \left(\frac{t - t_m^k - d_{ij}}{\tau_{ij}}\right) \mathrm{e}^{-\left(\frac{t - t_m^k - d_{ij}}{\tau_{ij}}\right)} \tag{3.2}$$

$$m \equiv \{N_i\}_j \tag{3.3}$$

$$\text{if:} \quad V_i(t) \;\geq\; \Theta_i \quad \Rightarrow \quad n_i \text{ fires: } t_i^K = t \tag{3.4}$$

There are four parameters for each synapse, $R_{ij}$, $T_{ij}$, $d_{ij}$, and $\tau_{ij}$, two parameters for each external input, $R_{ix}$ and $T_{ix}$, and three parameters associated with each neuron, $\tau_i$, $R_{i0}$, and $\Theta_i$. The choice of these parameters determines the behavior of the neuron. The variables in this model are defined as:

| | | |
|---|---|---|
| $n_i$ | = | Refers to neuron #i. |
| $V_i(t)$ | = | Potential voltage of $n_i$. This is an internal state variable. |
| $\Theta_i$ | = | Threshold voltage for $n_i$. When $V_i(t)$ reaches $\Theta_i$, $n_i$ produces an output spike. ($\Theta_i > 0$) |
| $U_{ij}(t)$ | = | The neurotransmitter in the $j^{\text{th}}$ synaptic gap of $n_i$ at time $t$. The total amount is scaled by $T_{ij}$. ($U_{ij}(t) \geq 0$) |
| $t_i^K$ | = | Time of the most recent output spike from $n_i$. |
| $t_m^k$ | = | Time of the $k^{\text{th}}$ output spike from $n_m$. |
| $N_i$ | = | Number of incoming synapses on $n_i$. |
| $\{N_i\}_j$ | = | Refers to the neuron connected to the $j^{\text{th}}$ synapse of $n_i$. The complete set of neurons connected to $n_i$ is denoted by $\{N_i\}$. Notice that a neuron may connect to another neuron through several synapses. |
| $d_{ij}$ | = | Time delay associated with the $j^{\text{th}}$ synapse of $n_i$. A spike produced by $n_m$ can not cause the release of neurotransmitter until an elapsed time of $d_{ij}$ has occurred. ($d_{ij} \geq 0$) |
| $\tau_{ij}$ | = | Synaptic time constant for the $j^{\text{th}}$ synapse of $n_i$. It sets the neurotransmitter diffusion rate across the synaptic gap. ($\tau_{ij} > 0$) |
| $\tau_i$ | = | Refractory time constant for $n_i$. It sets the rate at which the ion channels are able to become responsive to the input neurotransmitter after $n_i$ has fired. ($\tau_i > 0$) |
| $R_{i0}$ | = | "Neuron Resting Potential." $R_{i0}$ represents the value that $V_i(t)$ converges to when there are no input or output spikes. |

| | | |
|---|---|---|
| $T_{ij}$ | $=$ | "Synaptic Transmitter Weight." $T_{ij}$ scales the amount of neurotransmitter released into the $j^{\text{th}}$ synapse, each time $n_m$ fires. When using inhibitory synapses, its value is negative. |
| $R_{ij}$ | $=$ | "Synaptic Receiver Weight." At the $j^{\text{th}}$ synapse of $n_i$, $R_{ij}$ represents the total number of receptors on $n_i$. $(R_{ij} \geq 0)$ |
| $P_i$ | $=$ | Number of external inputs going into $n_i$. |
| $I_{ix}(t)$ | $=$ | The $x^{\text{th}}$ external input signal into $n_i$, at time $t$. It represents the measurement of an environmental parameter. $(I_{ix}(t) \geq 0)$ |
| $T_{ix}$ | $=$ | "Input Scaling Weight." $T_{ix}$ scales the strength of the external input, $I_{ix}(t)$. If the input inhibits $n_i$, its value is negative. |
| $R_{ix}$ | $=$ | "Input Sensitivity Weight." $R_{ix}$ represents the sensitivity of $n_i$ to a particular external input signal. $(R_{ix} \geq 0)$ |

## 3.3 Interpretation of SNM Terms

Every time $n_m$ fires, a spike propagates down its axon. The time required for this spike to reach the $j^{\text{th}}$ synapse of $n_i$ is determined by $d_{ij}$. When the spike arrives, neurotransmitter is released and diffuses across the synaptic gap until it reaches the postsynaptic neurotransmitter receptors on $n_i$ with the functional form[1] of:

$$\left(\frac{t - t_m^k - d_{ij}}{\tau_{ij}}\right) \mathrm{e}^{-\left(\frac{t-t_m^k-d_{ij}}{\tau_{ij}}\right)} \qquad \text{for:} \quad t \geq t_m^k + d_{ij} \tag{3.5}$$

[1]This function is sometimes written as $\left(\frac{t}{\alpha}\right)\mathrm{e}^{-\alpha t}$ and is referred to as the "$\alpha$-function." Some neuron models use this function to account for the induced ionic current through the postsynaptic membrane after a presynaptic neuron fires and releases neurotransmitter. (See [50, p. 100] as an example.) In this model it has a slightly different interpretation: it represents the amount of neurotransmitter available to the postsynaptic neuron. But the probability that a postsynaptic receptor will respond and open an ion channel is given by $\tanh\left(\frac{T_{ij}}{R_{ij}} \cdot U_{ij}(t)\right)$. Thus, when the presynaptic neuron fires repeatedly, there is a buildup of neurotransmitter, but the response of the postsynaptic neuron is bounded by the tanh(−) function.

This function peaks when $t = t_m^k + d_{ij} + \tau_{ij}$, so most of the neurotransmitter is not received by $n_i$ until time $(d_{ij} + \tau_{ij})$ after $n_m$ has fired. Furthermore, the neurotransmitter is assumed to dissipate with time, so that the total amount of neurotransmitter available from each spike is given by:

$$\int_{t_m^k+d_{ij}}^{\infty} \left(\frac{t - t_m^k - d_{ij}}{\tau_{ij}}\right) \mathrm{e}^{-\left(\frac{t-t_m^k-d_{ij}}{\tau_{ij}}\right)} dt = \tau_{ij} \tag{3.6}$$

Since the neurotransmitter does not dissipate immediately, a rapid firing of $n_m$ causes a buildup in the neurotransmitter at its output synapses. (The amount of neurotransmitter released by $n_m$ is assumed to be independent of the current amount of neurotransmitter in the synaptic gap.) This buildup is accounted for by summing over all of the previous output spikes from $n_m$; i.e., the total amount of neurotransmitter in the $j^{\text{th}}$ synapse of $n_i$ is given by:

$$T_{ij} \sum_{t_m^k \leq t-d_{ij}} \left(\frac{t - t_m^k - d_{ij}}{\tau_{ij}}\right) \mathrm{e}^{-\left(\frac{t-t_m^k-d_{ij}}{\tau_{ij}}\right)} \tag{3.7}$$

Here, $T_{ij}$ is a scaling factor to account for variations in synaptic strength.[2] Notice that $t_m^k \leq t - d_{ij}$ represents all of the previous spikes that occurred at or before $t - d_{ij}$. Spikes produced after this time would not yet have reached the synapse.

Although a rapidly firing neuron creates a buildup of neurotransmitter, the response of the postsynaptic neuron to the neurotransmitter must be bounded. This is accomplished in the SNM by using the tanh(−) of the neurotransmitter; i.e.:

$$\tanh\left(\frac{T_{ij}}{R_{ij}} \cdot U_{ij}(t)\right) \tag{3.9}$$

[2]If the synaptic transmitter weight varies with time, $T_{ij} = T_{ij}(t)$, then the amount of neurotransmitter released due to a spike depends upon its value at the time the spike arrives at the synapse; i.e., the amount of neurotransmitter is actually given by:

$$\sum_{t_m^k \leq t-d_{ij}} T_{ij}(t_m^k + d_{ij}) \cdot \left(\frac{t - t_m^k - d_{ij}}{\tau_{ij}}\right) \mathrm{e}^{-\left(\frac{t-t_m^k-d_{ij}}{\tau_{ij}}\right)} \tag{3.8}$$

This term represents the probability of a postsynaptic receptor on neuron $n_i$ receiving any of the neurotransmitter in the synaptic gap. Obviously, as the total amount of neurotransmitter increases, the probability that any of it arrives at a receptor increases, but not above one. Notice that the total amount of neurotransmitter is divided by $R_{ij}$, the number of post-synaptic receptors. This accounts for fact that if the number of receptors is increased while the amount of neurotransmitter remains constant, then the probability of any *one* receptor receiving neurotransmitter is decreased. But, the postsynaptic response to the neurotransmitter depends upon both the number of receptors at the synapse and the probability of receiving neurotransmitter. Thus, the effect each synapse has on the neuron's potential is given by:

$$R_{ij} \cdot \tanh\left(\frac{T_{ij}}{R_{ij}} \cdot U_{ij}(t)\right) \tag{3.10}$$

Since $R_{ij}$ represents the total number of receptors on $n_i$, scaled by the effect each receptor has on the neuron's potential, it will take into account any attenuation due to the synapse's location in the dendritic branches.[3] The total change in a neuron's potential, $V_i(t)$, is given by the sum of the responses at each of its incoming synapses, with the total number of input synapses given by $N_i$. Since there may be more than one synapse between neurons [53, p. 1016], $N_i$ does not necessarily represent the number of input neurons, and the subscript $i$ is used to indicate that the number of inputs varies with each neuron. Notice that $m = \{N_i\}_j$ refers to the neuron connected to the $j^{\text{th}}$ synapse of $n_i$.

If the amount of neurotransmitter at all of its input synapses is sufficient to cause the potential voltage of $n_i$ to reach $\Theta_i$, then the neuron produces an output spike. This spike does not affect the neurotransmitter concentration at its input synapses, but instead triggers the release of neurotransmitter at its output synapses.

Sensory neurons also have external input signals which encode the measurement of

[3]Actually, it is only the neuron's potential at the axon hillock that matters for producing a spike. Thus, in addition to attenuation, there may also be a delay from the time a change in potential is created at the synapse until it propagates to the axon hillock; however, it is not necessary to include a separate delay term in the SNM for this, since it can be included within the $d_{ij}$ delay.

environmental parameters. While most of the neurons in the human brain are used to process sensory information, the actual number of neurons that directly receive sensory signals are relatively few. To account for these special neurons, the SNM includes the term of:

$$\sum_{x=1}^{P_i} R_{ix} \cdot \tanh\left(T_{ix} \cdot I_{ix}(t)\right) \tag{3.11}$$

Unlike the spikes arriving at a synapse, the external input, $I_{ix}(t)$, is an analog signal, which is directly input into a neuron; consequently, the $\tanh(-)$ function does not represent the probability of a receptor receiving neurotransmitter, but simply limits the effective response of a neuron to the stimulus. The input weight of $T_{ix}$ scales the external input, $I_{ix}(t)$, received by $n_i$, and the sensitivity weight of $R_{ix}$ represents the sensitivity of $n_i$ to a particular signal. To understand these parameters, consider an eye focusing on a screen of uniform color and illumination. While the external input signal into all sensory neurons of the retina is the same, there are differences in intensity between the neurons due to their size and position within the retina. The $T_{ix}$ term accounts for these variations. In addition, the neurons also differ in their sensitivity to particular colors, which is controlled by the $R_{ix}$ parameter.

Finally, the production of an output spike is assumed to affect the receptivity of the neuron's ion channels to neurotransmitters or external input signals. (In a biological neuron, the generation of an action potential is followed by a period of residual inactivation in the $Na^+$ channels and the opening of $K^+$ channels. This causes a refractory period, during which the neuron is unable to produce another spike. See [53, p. 110].) The percentage of ion channels able to respond to the input signals and actually work to change the potential voltage of the neuron is given by:

$$\tanh^2\left(\frac{t - t_i^K}{\tau_i}\right) \qquad \text{for:} \quad t \geq t_i^K \tag{3.12}$$

Thus, the percentage of ion channels available depends upon the elapsed time since the neuron fired, and none of the ion channels are able to respond immediately after the neuron fires. This forces a refractory period on the neuron – a time immediately after firing when it is unable to produce another spike (see Section 4.3.3). Notice that since the square of the tangent function is used, the time derivative of the refractory coefficient is zero immediately after the neuron fires.

In this model for neuron dynamics, there is no information concerning the neuron spikes; e.g., the spike's width or height. Instead, only the onset times of the spikes are used to influence the resetting of the neuron's potential voltage and the release of neurotransmitter into the output synapses.

## 3.4 Synaptic Variability

### 3.4.1 Introduction

One shortcoming of the SNM presented in Equations (3.1)-(3.4) is that it is only capable of modeling "typical" axo-dendritic synapses; however, other types of synapses exist. Two of the more important variations for neural computation are synapses with presynaptic inhibition and synaptic clusters. With some minor modifications, the SNM can account for these special synapses, which are depicted in Figure 3.1.

Most inhibitory synapses connect to the dendrites or soma of another neuron and change the potential voltage of the entire neuron. But with presynaptic inhibition, the synapse instead connects to the presynaptic plate of another axon and inhibits its ability to release neurotransmitter. Thus, presynaptic inhibition allows the inputs into a neuron to be selectively suppressed.

In a synaptic cluster, several synapses are closely spaced together. The synaptic cluster shown in Figure 3.1 is a convergent glomeruli, where several axons connect to a dendrite. In this case, the response of the postsynaptic neuron depends upon the total amount of neurotransmitter released from all of the incoming axons. Thus, the neuron's potential changes as if there were only one synapse receiving input from several sources.

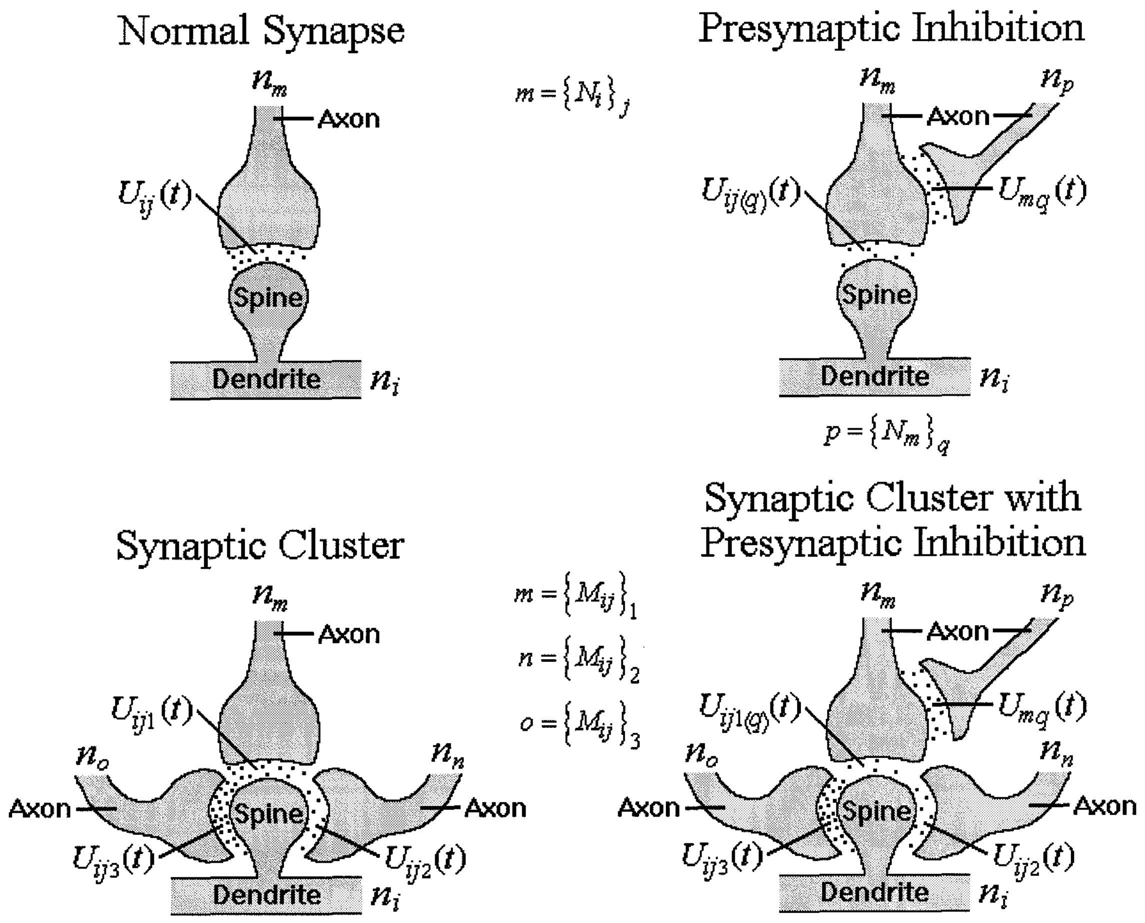


Figure 3.1: **Synapses.** The top left diagram shows a normal synapse where $n_m$ is connected to the $j^{\text{th}}$ synapse of $n_i$ without presynaptic inhibition. The neurotransmitter, $U_{ij}(t)$, is modeled with Equation (3.2). The top right diagram shows a synapse connecting $n_m$ to $n_i$ with presynaptic inhibition from $n_p$. The neurotransmitter released from $n_p$, $U_{mq}(t)$, is modeled as a normal synapse, but it does not effect the potential voltage of $n_m$. Only the neurotransmitter released from $n_m$ to $n_i$, $U_{ij(q)}(t)$, is diminished, and modeled with Equation (3.13). The bottom left diagram shows a synaptic cluster, where the total amount of neurotransmitter received by the dendritic spine comes from three axons. The neurotransmitter released by each presynaptic neuron, $n_m$, $n_n$, & $n_o$, is modeled as a normal synapse, but the response of the postsynaptic neuron, $n_i$, is modeled with Equation (3.14). The bottom right diagram shows a synaptic cluster with presynaptic inhibition on one of the input connections. The response of the postsynaptic neuron is still modeled with Equation (3.14), but the release of neurotransmitter from $n_m$ is modeled with Equation (3.13).

### 3.4.2 Presynaptic Inhibition

In the SNM, the equations for the neuron dynamics can be easily modified to account for presynaptic inhibition. In this case, the equation that governs neurotransmitter concentration becomes:

$$U_{ij(q)}(t) = \sum_{t_m^k \leq t-d_{ij}} \left( \frac{t - t_m^k - d_{ij}}{\tau_{ij}} \right) \mathrm{e}^{-\left[ \left( \frac{t-t_m^k-d_{ij}}{\tau_{ij}} \right) - R_{mq} \tanh\left( \frac{T_{mq}}{R_{mq}} U_{mq}(t_m^k + d_{ij}) \right) \right]} \tag{3.13}$$

Here, the $q$ subscript is put in parentheses to show that the neurotransmitter in the $j^{\text{th}}$ synapse of $n_i$ depends upon $U_{mq}(t)$, which is the neurotransmitter in the $q^{\text{th}}$ synapse of $n_m$. (Remember that $m$ represents the *neuron* that connects to the $j^{\text{th}}$ synapse of $n_i$, $m = \{N_i\}_j$, while the neuron into the $q^{\text{th}}$ synapse of $n_m$, $p$, is given by $p = \{N_m\}_q$.) The value of $U_{mq}(t)$ is calculated as in the case of a normal synapse, using Equation (3.2); however, since it only affects the $j^{\text{th}}$ synapse of $n_i$, it does appear in the formula for the potential voltage of $V_m$. Thus, when $U_{mq}(t)$ is zero, $U_{ij(q)}(t)$ acts as a normal synapse, but as $U_{mq}(t)$ increases, it inhibits the release of neurotransmitter, causing subsequent spike outputs from $n_m$ to have less influence on $n_i$. (Since $U_{mq}(t)$ is assumed to be an inhibitory synapse, $T_{mq} < 0$.[4]) Notice that the attenuating effect only depends upon the value of $U_{mq}(t)$ at the time of neurotransmitter release, $t = t_m^k + d_{ij}$, and does not vary with time itself.

While the above formula includes the effects from a single presynaptic connection, there may be several which influence the neurotransmitter output at a synapse, and each of these connections may form separate synapses or be grouped within synaptic clusters, which are discussed in the next section. In general, any type of connection that a neuron can make onto another neuron to alter its potential voltage, it may also make onto an axon as a presynaptic connection to alter the release of neurotransmitter

[4]In biology, most presynaptic connections are believed to be inhibitory; however, there are also excitatory connections which enhance the release of neurotransmitter at the synapse (see [53, pp. 207-209] for details). Equation (3.13) can also be used to account for such excitatory connections with $T_{mq} > 0$. But since the dependence of the presynaptic synapse upon $U_{ij(q)}(t)$ is exponential, small values for $R_{mq}$ should be used. The remainder of this thesis will use the term presynaptic connection to refer to any synapse, excitatory or inhibitory, of this type.

at a specific synapse.

### 3.4.3 Synaptic Clusters

In the SNM, the equations for the neuron dynamics can be altered to account for convergent synaptic clusters[5] by adding the total amount of neurotransmitter released by each input axon before calculating its effect on the receiving neuron; i.e.:

$$\begin{aligned} V_i(t) &= \tanh^2\left(\frac{t-t_i^K}{\tau_i}\right)\cdot\left\{R_{i0}+\sum_{j=1}^{N_i}R_{ij}\cdot\tanh\left(\frac{1}{R_{ij}}\cdot\sum_{l=1}^{M_{ij}}T_{ijl}\cdot U_{ijl}(t)\right)\right. \\ &\qquad\left.+\sum_{x=1}^{P_i}R_{ix}\cdot\tanh\left(T_{ix}\cdot I_{ix}(t)\right)\right\} \end{aligned} \tag{3.14}$$

$$\text{where:}\quad U_{ijl}(t) = \sum_{t_m^k\le t-d_{ijl}}\left(\frac{t-t_m^k-d_{ijl}}{\tau_{ijl}}\right)\mathrm{e}^{-\left(\frac{t-t_m^k-d_{ijl}}{\tau_{ijl}}\right)} \tag{3.15}$$

Here, the total number of inputs into the $j^{\text{th}}$ synapse of $n_i$ given by $M_{ij}$, and $\{M_{ij}\}$ represents the set of all neurons connected to $n_i$ at its $j^{\text{th}}$ synapse. Notice that $n_m$ represents the $l^{\text{th}}$ input neuron into the $j^{\text{th}}$ synapse on $n_i$, ($m = \{M_{ij}\}_l$).[6]

The bottom right diagram in Figure 3.1 shows a convergent synaptic cluster with presynaptic inhibition. In this case, Equation (3.14) still applies, but the neurotransmitter, $U_{ijl}(t)$, is modeled with Equation (3.13) for those input connections which have presynaptic inhibition or excitation. Notice that the presynaptic connection may in fact be from one of the other neurons which connects to $n_i$. Thus, while

[5]Some synaptic clusters are divergent, where one axon connects to several dendrites. However, since the neurotransmitter received by each dendrite affects its potential independent of the others, this type of cluster does not require any special modification to the original SNM; i.e., the total amount of neurotransmitter released by the axon will be distributed among the dendrites, but each connection can be modeled as separated synapses with the corresponding weight values reflecting the neurotransmitter distribution.

[6]For the remainder of this thesis, the term "connection" will be used to refer to a location where two neurons are joined together, and the term "synapse" will be used to refer to the locations on the receiving neuron that may have one or more connections. At a synaptic cluster, the neurotransmitters, $U_{ijl}$, and transmitter weights, $T_{ijl}$, are associated with the connections, while the receiver weight, $R_{ij}$, is associated with the synapse. When it is clear that there is only one connection at a synapse, the term "synapse" will be used to refer to all of the associated variables.

sending an output signal, a neuron may also inhibit others from transmitting theirs.

Real neurons use different receptors for the inhibitory and excitatory neurotransmitter. Thus, all of the incoming connections tend to be either inhibitory or excitatory within a cluster; however, for the SNM there is no such restriction. In fact, a cluster can have two input connections with different delays from the same neuron, with one being excitatory and the other being inhibitory. And if the inputs into the clusters are not from the same neuron, but are synchronized, then the neuron can be used to evaluate *any* Boolean logic expression (see Section 7.4).

## 3.5 Summary of the Spiking Neuron Model

The SNM is designed to capture much of the interesting dynamics demonstrated by real neurons, while remaining simple enough to simulate and understand. Spikes are discrete events that have no width or height associated with them. Only the interspike time intervals matter, as the spikes reset the neuron's potential voltage and trigger the release of neurotransmitter at the interneuron connections.

A network is composed of neurons, which may receive input from other neurons or external input signals. The sign of the parameters associated with each connection determines whether it has an excitatory or inhibitory effect on the neuron. When a spike arrives at an input connection, neurotransmitter is released into the synaptic gap with the functional form of $te^{-t}$. The arrival time is determined by the time the spike was produced, plus the delay associated with the connection.

The neurotransmitter's effect on the neuron's potential is modeled by the hyperbolic tangent of the neurotransmitter concentration at each synapse. When a group of input connections converges at the same location, it is referred to as a synaptic cluster. All of the cluster's input connections contribute to the total neurotransmitter concentration at the synapse, and their overall effect on the neuron's potential is determined by the tanh $(-)$ of the sum. While most synapses connect directly onto neurons, some connect onto other inputs, forming presynaptic connections, which alter the strength of the input connection onto which they are joined. As with synapses connecting onto neurons, there can be presynaptic clusters, which contain several input connections.

External input connections are very similar to interneuron connections, but they are assumed to have no delay associated with them, and their effect on the neuron's potential is given directly by the tanh $(-)$ of the input signal; i.e., there is no release of neurotransmitter involved. Also, while a neuron may receive several external inputs, this thesis assumes that they do not form input clusters.

A neuron's parameters may be set so that it is capable of producing spikes in the absence of any input signals. Essentially all neurons can be divided into two classifications: (1) neurons with $\Theta_i \geq R_{i0}$; and (2) neurons with $\Theta_i < R_{i0}$. When $\Theta_i \geq R_{i0}$, a neuron can not fire unless there is an excitatory input signal. (Theoretically, if $\Theta_i = R_{i0}$, a neuron will fire once and then its refractory coefficient will inhibit it from ever firing again.) But when $\Theta_i < R_{i0}$, a neuron can produce an output spike pattern without any input signals. Of course, input signals can modify or inhibit the neuron's endogenous output spike pattern. This is discussed in Section 4.3.4.

While a mathematical description is necessary to simulate the neurons, it is also useful to develop symbols for the different network components to illustrate a network's connectivity. Figure 3.2 shows the symbols for the SNM components that are used in network schematics. Obviously, since each symbol has at least one parameter associated with it, a network schematic alone, without the parameters, is insufficient for determining the network behavior; however, it can be a useful tool for understanding the relationships between neurons.

One symbol shown in Figure 3.2 that does not fit into the SNM is the Neuron with Random Output. Occasionally it is useful to have neurons producing randomly generated output spike trains to test the network's behavior. These neurons do not receive any input signals, and the time between their output spikes is determined by:

$$t_m^{K+1} - t_m^K = \lambda - \gamma \cdot \log(1 - \varepsilon) \tag{3.16}$$

Here, $t_m^K$ represents the time of the last output spike, $t_m^{K+1}$ represents the time of the next output spike, $\lambda$ represents the absolute refractory period, $\gamma$ represents the

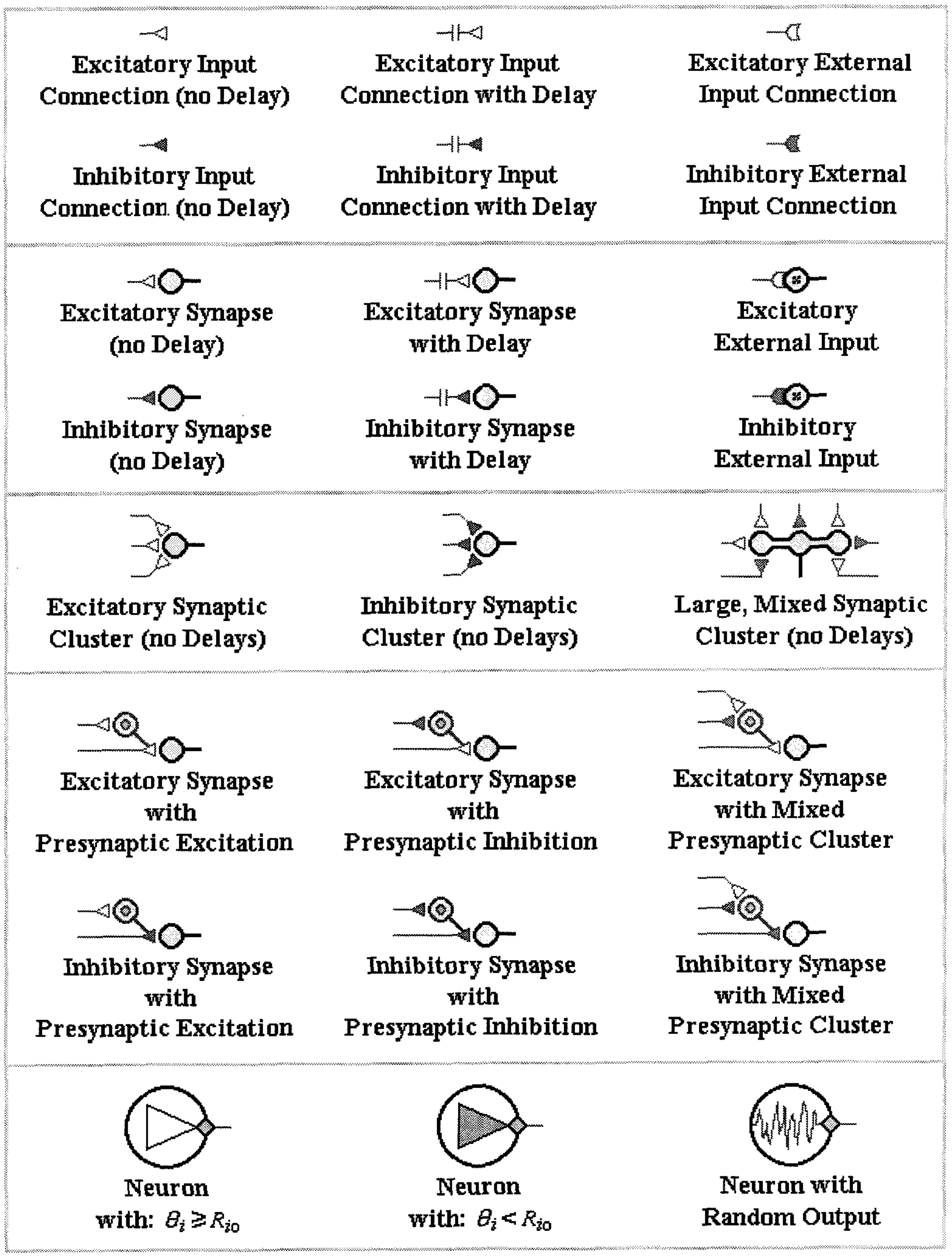

Excitatory Input Connection (no Delay)
Excitatory Input Connection with Delay
Excitatory External Input Connection
Inhibitory Input Connection (no Delay)
Inhibitory Input Connection with Delay
Inhibitory External Input Connection
Excitatory Synapse (no Delay)
Excitatory Synapse with Delay
Excitatory External Input
Inhibitory Synapse (no Delay)
Inhibitory Synapse with Delay
Inhibitory External Input
Excitatory Synaptic Cluster (no Delays)
Inhibitory Synaptic Cluster (no Delays)
Large, Mixed Synaptic Cluster (no Delays)
Excitatory Synapse with Presynaptic Excitation
Excitatory Synapse with Presynaptic Inhibition
Excitatory Synapse with Mixed Presynaptic Cluster
Inhibitory Synapse with Presynaptic Excitation
Inhibitory Synapse with Presynaptic Inhibition
Inhibitory Synapse with Mixed Presynaptic Cluster
Neuron with: $\theta_i \geq R_{io}$
Neuron with: $\theta_i < R_{io}$
Neuron with Random Output

Figure 3.2: **Symbols for Network Components.** This diagram illustrates the components that can be used in a network based on the SNM. In general, any interneuron connection may have a delay associated with it, but external input connections do not. Synaptic clusters may be composed of either excitatory or inhibitory input connections or both. Synapses and synaptic clusters may connect to either neurons or interneuron connections, forming presynapses. External inputs are assumed not to form clusters, and can only connect directly to a neuron. The different symbols for neurons are used to differentiate between their intrinsic output spike patterns.

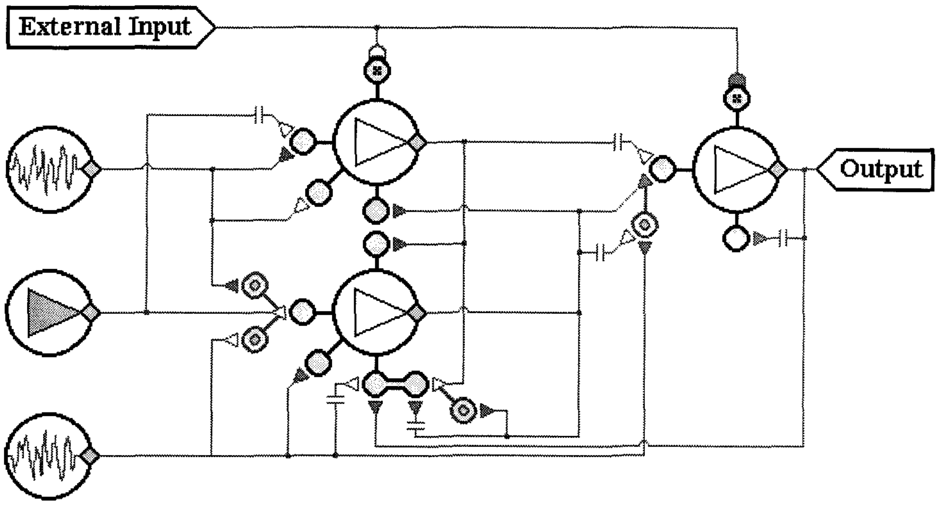


Figure 3.3: **Example Schematic for Neural Network.** This diagram demonstrates the various ways the components of Figure 3.2 can be connected within a network. Some things to notice are: (1) Any interneuron connection can contain delay; (2) Synapses may connect to either neurons or interneuron connections, forming presynapses; (3) Both neurons and interneuron connections may have several synapses or synaptic clusters connected onto them; (4) Neurons may have self connections; (5) Neurons and external inputs may form both excitatory and inhibitory connections; (6) A neuron may connect to another neuron at more than one synapse; (7) Pairs of neurons may have reciprocal connections.

"gain," and $\varepsilon$ is a uniformly distributed random number between 0 and 1. The average time between spikes is $\lambda + \gamma$, with a standard deviation of $\gamma$. With this function determining the output spikes, the probability that the time between spikes is less than or equal to $x$ is given by:

$$P\left(t_m^{K+1} - t_m^K \leq x\right) = \begin{cases} 0 & \text{if } x < \lambda \\ 1 - \mathrm{e}^{-\left(\frac{x-\lambda}{\gamma}\right)} & \text{if } x \geq \lambda \end{cases} \qquad (3.17)$$

Figure 3.3 shows a sample neural network which contains two random output neurons. The network is not designed to perform a particular task, but simply illustrates some of the various ways that the SNM's components can be connected.

## 3.6 Comparison with Other Neuron Models

A model is a tool used to help understand something complicated. A good model reduces the complexity of the system significantly while still preserving its essential features. The degree of reduction needed in a model clearly depends upon the problem to be solved. Among neurobiologists there is some controversy as to the level of detail required in neuron models [101]. Some believe that when a neuron operates within a large network, its details are superfluous, and its input/output characteristics can be represented by a simple model. Others believe that a neuron's morphology and electrical properties play an important role in its computational functions; consequently, models should retain all of the known details. This controversy has led to an abundance in neuron models. In general, most of the models can be classified according to the type of signal used (e.g., digital, analog, or spiking), and the degree to which they model the neuron's geometry (e.g., axon and dendritic branches).

The simplest models are those that ignore the neuron branches and use digital signals. The most significant among these models are those of McCulloch and Pitts [71], Minsky and Papert [75], and Hopfield [45]. The neurons are discrete in nature, both because they have binary states and because time is quantized. At each discrete time step, the excitatory and inhibitory inputs into the neurons are summed and all of the neurons whose total input exceeds a certain voltage threshold, produce an "on" output. (Some models use an asynchronous updating scheme – see Section 1.2.) The rationale behind this type of model is the belief that when there are a large number of connected elements, the network's emergent collective behavior is insensitive to the details of the model. (Hopfield [46] showed that many network properties are preserved when binary neurons are replaced with continuous neurons.) Obviously, the primary advantage of these models is their simplicity, which allows large networks of them to be simulated and analyzed [101].

Probably the most popular neuron models for computation are those which use continuous (analog) output signals. The primary advantage they have over their discrete counterparts is in the differentiability of the transfer function, which makes them suitable for use with backpropagation learning algorithms. The dynamics for these neurons are usually represented by differential equations, which are integrated forward as the network evolves in time (e.g., [34, pp. 72-78], [46], [83], [42, pp. 54-60]). Many of these models are based on an electrical circuit representation of the neuron's membrane consisting of a resistor and capacitor in parallel. The resistor represents the conductivity of transmembrane channels through which the ionic currents flow, while the capacitor separates the charges on each side of the membrane.

One of the shortcomings of both the digital and analog neuron models is their inability to account for refractory periods, which rely on the neurons' past histories. Refractory periods are believed to play a major role in shaping the dynamics of neuronal populations [74, p. 14]. The inclusion of a refractory period is not a problem when using a neuron model with spiking dynamics. These models are usually created by neurobiologists for the purpose of understanding biology, and few researchers have studied spiking neurons for use in computing paradigms. The spiking neuron models range from phenomenological to realistic channel-based models. (See [59] for a survey of the more popular models used by neurobiologists.) Phenomenological models include the integrate-and-fire type, in which the inputs are integrated in time and when the voltage reaches a threshold level, a spike is produced. Of the realistic channel-based models, the best known example is the three channel Hodgkin-Huxley model [43], which describes the initiation and propagation of action potentials in the squid giant axon.[7]

The original Hodgkin-Huxley model consists of four coupled differential equations, one for the membrane voltage, two for the $Na^+$ current, and one for the $K^+$ current. This model allows phenomena, such as the spiking threshold and refractory period, to be understood on physical grounds. Unfortunately, the model is too complicated for a detailed mathematical analysis, and even when being simulated on computer, simplifying assumptions are usually unavoidable [59, pp. 1-2]. For this reason, several

[7]The Hodgkin-Huxley model actually uses continuous dynamics, and the spikes or "action potentials" are a consequence of the different time scales associated with the differential equations. Thus, for some neuron models there is no clear distinction between an analog or spiking output signal.

attempts have been made to reduce the Hodgkin-Huxley model to two coupled differential equations. Of this class of models, the FitzHugh-Nagumo model [23, 79] is one of the best known. It has proven useful for gaining mathematical insight into the electrical behavior of a biological neuron's membrane. However, the parameters used to define this and similar models often have no direct physical interpretation [101].

One of the directions in neuronal modeling that has gained momentum in recent years is to include the effects of the dendritic branches. Rall [85, 88, 86] was the first to explore the physiological significance of dendrites. He ignored the membrane nonlinearities and treated the dendrites as an electrically passive tree. By using the boundary conditions imposed by the tree structure, the voltage spread in time and space could be described by the second order partial differential equation established by Hodgkin and Rushton [44]. This allowed the voltage response at any point in an arbitrarily complex passive tree to be characterized as a function of the currents injected at other points. To overcome the constraint of a passive membrane, Rall [87] developed a complementary compartmental modeling approach (reviewed in [100]). Mathematically, the compartmental approach replaces the continuous cable equation by a matrix of ordinary differential equations. The problem with this approach is that the numerical methods needed to solve this system of equations (which can include thousands of compartments for each neuron) at each time step are very computationally demanding [101]. (A popular neuron simulator, based on this compartmental approach is GENESIS, develop by Bower. See [9] for details.)

By modeling the specific structure and biophysical properties of neurons, researchers have been able to demonstrate some possible computations that can be performed by single neurons. First, a neuron can be directionally selective and used to compute direction of motion [101]. Second, a neuron may respond preferentially to specific input sequences, and can thus be used to recognize certain spatial-temporal patterns [101]. And third, the strategic location of inhibitory and excitatory inputs onto the tree can be use to implement logic operations [101]. Fortunately, all of these functions can be accomplished with the SNM.

While the modeling of the ionic currents and dendritic branches may be necessary by neurobiologists to *understand* the computations being performed by neurons within biological systems, the SNM assumes that such details are not needed to *duplicate* the computations. In this sense, the SNM is relatively simple. It does not attempt to directly model the neuron's morphology, but its effects are included within the synaptic parameters. However, one limitation of the SNM is that it is unable to account for the possible interactions that may occur between inputs within the dendrites.

The SNM has properties similar to the integrate-and-fire models, with the spikes being discontinuities in the underlying analog variables. But one distinguishing characteristics of the SNM is that the neuron dynamics are not based on a set of differential equations. During simulation, this feature allows the neuron state variables to evolve forward in time without worrying about the size of the time step or the stability of the numerical integration technique. Of the neuron models described in literature, the SNM is similar to the more complicated "Spike Response Model" of [25, 27]. In this model, the effect from each of the incoming spikes has a functional form of $te^{-t}$, which is analogous to the input neurotransmitter, $U_{ij}(t)$, in the SNM. But the neuron's refractory term takes a different form, and is additive rather than multiplicative. Also, the production of a spike is probabilistic rather than deterministic.

One of the primary advantages of the SNM over other spiking models is that it is *not* based on any differential equations. Consequently, the stability of the simulated dynamics is not affected by the integration step size. A novel simulation method was presented, called "Simulating in Time Segments," in which all of the time steps within a relatively large segment of time are simulated all at once. It uses an iterative approach; each spike that occurs during the time segment requires a recalculation of the neurotransmitter at the affected synapses. Of course, the additional neurotransmitter can in turn lead to more spikes, but time segments with few spikes can be quickly simulated.

Overall, the SNM fulfills its purpose of providing a relatively simple spiking neuron model which can be used for spike based computations. It is able to produce spike trains comparable to those of real neurons (see Section 4.3.4). But there is no reason to suspect that other spiking neuron models with similar properties can not be used in similar configurations to accomplish the same computations. Thus, depending upon the purpose of the simulation, more complex neuron models may be used, and in some cases even simpler models may suffice.

# Chapter 4 Properties of the SNM

## 4.1 Introduction

Since the SNM is intended to duplicate some of the computational abilities of biological neurons, it is useful to analyze and compare its properties with the observed properties of real neurons. Obviously, the key to using a spiking model is to have a neuron which produces action potentials when presented with a sufficiently large excitatory stimulus. In the SNM, the threshold potential, $\Theta_i$, sets the critical stimulus strength for generating a spike. The neuron only responds passively when the stimulus results in a potential voltage below $\Theta_i$. It is the production of a spike that triggers the release of neurotransmitter at the neuron's output synapses. Thus, only stimuli of sufficient strength (importance) result in information being transferred to other neurons via neurotransmitter.

Section 4.2 discusses some of the neurotransmitter properties, and shows how the total amount of neurotransmitter does not need to be represented as a sum over all of the previous spikes, but can instead be represented by an iterative formula. Furthermore, when the spikes are periodic, the iterative formula for the neurotransmitter can be simplified. Section 4.3 examines the neuron potential, $V_i(t)$, and tests the SNM's ability to replicate some of the well known properties in biological neurons, namely, latency, refractory periods, endogenous oscillatory spiking behavior, and adaptation of firing rate. Section 4.4 analyzes the SNM's frequency transfer function. And finally, Section 4.5 summarizes the properties of the SNM.

## 4.2 Neurotransmitter Concentration, $U_{ij}(t)$

### 4.2.1 Introduction

The release and uptake of neurotransmitter are perhaps the least understood neuronal functions, and the SNM only provides a crude approximation to these complex biochemical reactions. In fact, the same equation is used to model all of the possible types of neurotransmitter. (Of course the parameters can be chosen differently between synapses. This allows for both inhibitory and excitatory neurotransmitters, as well as variations in time constants.) Also, the SNM does not account for the spontaneous release of neurotransmitter without stimulation, nor the random fluctuations in the amount of neurotransmitter released when evoked by an action potential.[1]

In the SNM, when neuron $n_m$ fires, neurotransmitter is released into all of $n_m$'s output connections. Since there may already be neurotransmitter at the connections as a result of previous output spikes, the total neurotransmitter concentration at the synapse connected to $n_i$, $U_{ij}(t)$, is the sum of the effects from all previous spikes; i.e.:

$$U_{ij}(t) = \sum_{t_m^k \le t - d_{ij}} \left( \frac{t - t_m^k - d_{ij}}{\tau_{ij}} \right) e^{-\left(\frac{t - t_m^k - d_{ij}}{\tau_{ij}}\right)} = \sum_k f\left( \frac{t - t_m^k - d_{ij}}{\tau_{ij}} \right) \tag{4.1}$$

$$\text{where:} \quad f(x) \equiv F(x) \cdot u(x) \tag{4.2}$$

$$F(x) \equiv x e^{-x} \tag{4.3}$$

Here, $u(x)$ is the unit step function defined by:

$$u(x) \equiv \begin{cases} 0 & x < 0 \\ 1 & x \ge 0 \end{cases} \tag{4.4}$$

[1] In the SNM, the amount of neurotransmitter released due to an action potential can vary through *learning*, i.e., as the parameters change. However, the release of neurotransmitter is deterministic, not probabilistic.

Also, the time derivative of the neurotransmitter concentration is given by:

$$\begin{aligned}
\dot{U}_{ij}(t) &= \left(\frac{1}{\tau_{ij}}\right) \sum_{t_m^k \leq t-d_{ij}} \left[1 - \frac{t - t_m^k - d_{ij}}{\tau_{ij}}\right] \mathrm{e}^{-\left(\frac{t-t_m^k-d_{ij}}{\tau_{ij}}\right)} \\
&= \left(\frac{1}{\tau_{ij}}\right) \left[\sum_k g\left(\frac{t - t_m^k - d_{ij}}{\tau_{ij}}\right) - \sum_k f\left(\frac{t - t_m^k - d_{ij}}{\tau_{ij}}\right)\right] \\
&= \left(\frac{1}{\tau_{ij}}\right) \left[\mu_{ij}(t) - U_{ij}(t)\right] \qquad (4.5)
\end{aligned}$$

$$\begin{aligned}
\text{where:} \quad \mu_{ij}(t) &\equiv \sum_k g\left(\frac{t - t_m^k - d_{ij}}{\tau_{ij}}\right) \qquad (4.6) \\
g(x) &\equiv G(x) \cdot u(x) \qquad (4.7) \\
G(x) &\equiv \mathrm{e}^{-x} \qquad (4.8)
\end{aligned}$$

In Equations (4.1) and (4.5), $d_{ij}$ and $\tau_{ij}$ represent the time delay and time constant associated with the connection, and $t_m^k$ represents the times of the output spikes from $n_m$, which is the neuron connected to the $j^{\text{th}}$ synapse of $n_i$.

The function $f(x)$ represents the total effect of the neurotransmitter released with each spike. Essentially, it is used to account for the propagation of the action potential down the axon to the synapse, the release of neurotransmitter and its diffusion across the synaptic gap, the response of the postsynaptic receptors,[2] the opening of ion channels within the postsynaptic membrane, and the propagation of the potential change through the dendritic branches to the axon hillock. Notice that while the effects from the previous input spikes decay with time, it is their summation that gives the SNM it unique properties. Unlike many neuron models, the state of a neuron does not just depend upon the current input signals, but the *entire* spike history of the presynaptic neurons.

Appendix A shows plots of $F(x)$ and $G(x)$, and discusses some of their useful properties. The properties of $U(t)$ and $\mu(t)$ are discussed in Appendix B. Section

[2]The tanh $(-)$ function is used to limit the response of the postsynaptic neuron from each synapse. It reflects the probability of neurotransmitter reaching a postsynaptic receptor. See Equation (3.9).

4.2.2 shows how Equation (4.1) can be expressed iteratively, and Section 4.2.3 analyzes the consequence of a periodic spike train on the neurotransmitter concentration at a synapse.

### 4.2.2 Iterative Representation for $U_{ij}(t)$

The problem with expressing $U_{ij}(t)$ as a sum is that the number of previous input spikes may be very large, and it is not practical to keep track of all of them to calculate $U_{ij}(t)$. Obviously, since the contribution of each individual spike approaches zero as time increases, the series can be *approximated* with a truncated one which only accounts for the most recent spikes. But if $U_{ij}(t_o)$ and $\dot{U}_{ij}(t_o)$ are known, where $t_o$ represents any time between the arrival time of the last input spike, $t_m^K + d_{ij}$, and the arrival time of the next spike, $t_m^{K+1} + d_{ij}$, then $U_{ij}(t)$ may be expressed *exactly* as a function of these two known quantities; i.e., from Equations (B.5) and (B.10):

$$\begin{aligned} & U_{ij}(t_m^K + d_{ij} \leq t = t_o + \delta \leq t_m^{K+1} + d_{ij}) \\ &= \sum_{t_m^k \leq t_m^K} \left( \frac{t_o + \delta - t_m^k - d_{ij}}{\tau_{ij}} \right) \mathrm{e}^{-\left(\frac{t_o+\delta-t_m^k-d_{ij}}{\tau_{ij}}\right)} \\ &= G\left(\frac{\delta}{\tau_{ij}}\right) U_{ij}(t_o) + F\left(\frac{\delta}{\tau_{ij}}\right) \mu_{ij}(t_o) \\ &= \mu_{ij}(t_o) \mathrm{e}^{\left(\frac{U_{ij}(t_o)}{\mu_{ij}(t_o)}\right)} \cdot F\left(\frac{\delta}{\tau_{ij}} + \frac{U_{ij}(t_o)}{\mu_{ij}(t_o)}\right) \end{aligned} \tag{4.9}$$

$$\begin{aligned} & \mu_{ij}(t_m^K + d_{ij} \leq t = t_o + \delta \leq t_m^{K+1} + d_{ij}) \\ &= \sum_{t_m^k \leq t_m^K} \mathrm{e}^{-\left(\frac{t_o+\delta-t_m^k-d_{ij}}{\tau_{ij}}\right)} = G\left(\frac{\delta}{\tau_{ij}}\right) \mu_{ij}(t_o) \end{aligned} \tag{4.10}$$

$$\text{where:} \quad \mu_{ij}(t_o) = U_{ij}(t_o) + \tau_{ij}\dot{U}_{ij}(t_o) \tag{4.11}$$

In these expressions, $U_{ij}(t_o)$ and $\mu_{ij}(t_o)$ are constants. However, after another input spike arrives, new values for $U_{ij}(t_o)$ and $\mu_{ij}(t_o)$ need to be found. But, by using the values for $U_{ij}(t_o)$ and $\mu_{ij}(t_o)$ at the arrival time of an incoming pulse, $(t_o = t_m^K + d_{ij})$, the values for $U_{ij}(t_o)$ and $\mu_{ij}(t_o)$ may be expressed in a recursive relationship; i.e., let $\mu_{ij}^K \equiv \mu_{ij}(t_m^K + d_{ij})$ and $U_{ij}^K \equiv U_{ij}(t_m^K + d_{ij})$, then:

$$\begin{aligned}
\mu_{ij}^K &= \sum_{t_m^k \le t_m^K} \mathrm{e}^{-\left(\frac{t_m^K - t_m^k}{\tau_{ij}}\right)} = \mathrm{e}^{-\left(\frac{t_m^K - t_m^{K-1}}{\tau_{ij}}\right)} \sum_{t_m^k \le t_m^{K-1}} \mathrm{e}^{-\left(\frac{t_m^{K-1} - t_m^k}{\tau_{ij}}\right)} + 1 \\
&= \mathrm{e}^{-\left(\frac{t_m^K - t_m^{K-1}}{\tau_{ij}}\right)} \mu_{ij}^{K-1} + 1 \qquad (4.12)
\end{aligned}$$

$$\begin{aligned}
U_{ij}^K &= \sum_{t_m^k \le t_m^K} \left(\frac{t_m^K - t_m^k}{\tau_{ij}}\right) \mathrm{e}^{-\left(\frac{t_m^K - t_m^k}{\tau_{ij}}\right)} = \sum_{t_m^k \le t_m^{K-1}} \left(\frac{t_m^K - t_m^k}{\tau_{ij}}\right) \mathrm{e}^{-\left(\frac{t_m^K - t_m^k}{\tau_{ij}}\right)} \\
&= \mathrm{e}^{-\left(\frac{t_m^K - t_m^{K-1}}{\tau_{ij}}\right)} \cdot \left[ \left(\frac{t_m^K - t_m^{K-1}}{\tau_{ij}}\right) \sum_{t_m^k \le t_m^{K-1}} \mathrm{e}^{-\left(\frac{t_m^{K-1} - t_m^k}{\tau_{ij}}\right)} \right. \\
&\qquad \left. + \sum_{t_m^k \le t_m^{K-1}} \left(\frac{t_m^{K-1} - t_m^k}{\tau_{ij}}\right) \mathrm{e}^{-\left(\frac{t_m^{K-1} - t_m^k}{\tau_{ij}}\right)} \right] \\
&= \mathrm{e}^{-\left(\frac{t_m^K - t_m^{K-1}}{\tau_{ij}}\right)} \cdot \left[ \left(\frac{t_m^K - t_m^{K-1}}{\tau_{ij}}\right) \mu_{ij}^{K-1} + U_{ij}^{K-1} \right] \qquad (4.13)
\end{aligned}$$

In matrix form, Equations (4.12)-(4.13) can be expressed as:

$$\begin{bmatrix} \mu_{ij}^K \\ U_{ij}^K \end{bmatrix} = \mathrm{e}^{-\left(\frac{t_m^K - t_m^{K-1}}{\tau_{ij}}\right)} \cdot \begin{bmatrix} 1 & 0 \\ \left(\frac{t_m^K - t_m^{K-1}}{\tau_{ij}}\right) & 1 \end{bmatrix} \cdot \begin{bmatrix} \mu_{ij}^{K-1} \\ U_{ij}^{K-1} \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad (4.14)$$

Thus, by iteratively calculating the $\mu_{ij}^K$ and $U_{ij}^K$ constants after each spike, an exact expression for $U_{ij}(t)$ can be used without needing to recalculate all of the summation terms in Equation (4.1); i.e., $\mu_{ij}^K$ and $U_{ij}^K$ can be used for $\mu_{ij}(t_o)$ and $U_{ij}(t_o)$ in Equation (4.9) to calculate the neurotransmitter between input spikes, $U_{ij}(t)$. Notice that to calculate the new values for $\mu_{ij}^K$ and $U_{ij}^K$ it is only necessary to use the time between the last two input spikes.

One important consequence of using the representation for $U_{ij}(t)$ given in Equation (4.9) occurs when attempting to simulate a neuron: it is not necessary to specify the entire history of the incoming spikes to set the neurotransmitter's initial condition. Instead, the initial condition may be specified in terms of $U_{ij}(t_o)$ and $\mu_{ij}(t_o)$, where

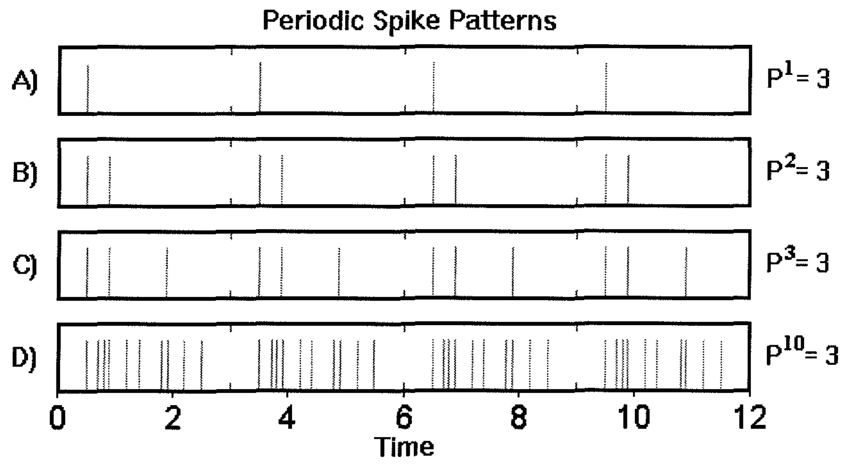


Figure 4.1: **Periodic Spike Patterns.** All four of the periodic patterns have a period of 3, but the number of spikes in each pattern varies. Patterns (A)-(C) have only 1, 2, and 3 spikes in their respective periods, while (D) has 10 spikes in its period.

$t_o$ is now the starting time of the simulation. And when the previous input spikes are assumed to be periodic in nature, explicit solutions for $U_{ij}(t_o)$ and $\mu_{ij}(t_o)$ can be found using the method presented in the next section.

### 4.2.3 Effect of Periodic Input Signals on $U_{ij}(t)$

Often a neuron converges to a periodic spiking pattern, and it is useful to consider the effects of such a pattern on the amount of neurotransmitter released by the neuron. Obviously, as the spiking pattern for $n_m$ becomes periodic, so will the function of $U_{ij}(t)$. A periodic spiking pattern can be described by the number of spikes in each period and the length of the period. With $P$ designating a periodic spike train, the notation is:

$$\begin{aligned} P^N &= \Delta \qquad (4.15) \\ \text{where:} \quad N &\equiv \text{Number of spikes in each period} \\ \Delta &\equiv \text{Time length of each period} \end{aligned}$$

Figure 4.1 illustrates this notation by showing some periodic spiking patterns.

Now, if the pattern has been repeating long enough for the neurotransmitter function to converge to a periodic function, then $U_{ij}(t) = U_{ij}(t+\Delta)$. Iteratively using Equation (4.14) $N$ times gives:

$$\begin{bmatrix} \mu_{ij}^{K+N} \\ U_{ij}^{K+N} \end{bmatrix} = \mathrm{e}^{-\left(\frac{\Delta}{\tau_{ij}}\right)} \cdot \begin{bmatrix} 1 & 0 \\ \left(\frac{\Delta}{\tau_{ij}}\right) & 1 \end{bmatrix} \cdot \begin{bmatrix} \mu_{ij}^{K} \\ U_{ij}^{K} \end{bmatrix} + \begin{bmatrix} \hat{\mu}_{ij}^{K}(N) \\ \hat{U}_{ij}^{K}(N) \end{bmatrix} \tag{4.16}$$

$$\text{where:} \quad \hat{\mu}_{ij}^{K}(N) = \sum_{i=0}^{N-1} \mathrm{e}^{-\left(\frac{t_m^K - t_j^{(K-i)}}{\tau_{ij}}\right)} \tag{4.17}$$

$$\hat{U}_{ij}^{K}(N) = \sum_{i=0}^{N-1} \left(\frac{t_m^K - t_j^{(K-i)}}{\tau_{ij}}\right) \mathrm{e}^{-\left(\frac{t_m^K - t_j^{(K-i)}}{\tau_{ij}}\right)} \tag{4.18}$$

Here, the functions of $\hat{\mu}_{ij}^{K}(N)$ and $\hat{U}_{ij}^{K}(N)$ each assume one of $N$ constant values; i.e., $\hat{\mu}_{ij}^{K}(N)$ and $\hat{U}_{ij}^{K}(N)$ take on different values after each spike, but these values are periodic and repeat every $N$ spikes. Since $\mu_{ij}^{K+N} = \mu_{ij}^{K}$ and $U_{ij}^{K+N} = U_{ij}^{K}$, the explicit solution for $\mu_{ij}^{K}$ and $U_{ij}^{K}$ is:

$$\begin{bmatrix} \mu_{ij}^{K} \\ U_{ij}^{K} \end{bmatrix} = \left(\frac{1}{1-\mathrm{e}^{-\left(\frac{\Delta}{\tau_{ij}}\right)}}\right)^2 \cdot \begin{bmatrix} 1-\mathrm{e}^{-\left(\frac{\Delta}{\tau_{ij}}\right)} & 0 \\ \left(\frac{\Delta}{\tau_{ij}}\right)\mathrm{e}^{-\left(\frac{\Delta}{\tau_{ij}}\right)} & 1-\mathrm{e}^{-\left(\frac{\Delta}{\tau_{ij}}\right)} \end{bmatrix} \cdot \begin{bmatrix} \hat{\mu}_{ij}^{K}(N) \\ \hat{U}_{ij}^{K}(N) \end{bmatrix} \tag{4.19}$$

Notice that:

$$\left(\frac{U_{ij}^K}{\mu_{ij}^K}\right) = \left(\frac{\left(\frac{\Delta}{\tau_{ij}}\right)\mathrm{e}^{-\left(\frac{\Delta}{\tau_{ij}}\right)}}{1-\mathrm{e}^{-\left(\frac{\Delta}{\tau_{ij}}\right)}}\right) + \frac{\hat{U}_{ij}^{K}(N)}{\hat{\mu}_{ij}^{K}(N)} = \psi\left(\frac{\Delta}{\tau_{ij}}\right) + \frac{\hat{U}_{ij}^{K}(N)}{\hat{\mu}_{ij}^{K}(N)} \tag{4.20}$$

where the function of $\psi$ is defined in Equation (C.5). Substituting these expressions for $\mu_{ij}^K$ and $U_{ij}^K$ back into Equations (4.9)-(4.10), let $U_{ij}(t)$ and $\mu_{ij}(t)$ be written in terms of the time intervals between the spikes within the periodic pattern; i.e.:

$$U_{ij}(t) = \hat{\mu}_{ij}^{K}(N)\mathrm{e}^{\left(\frac{\hat{U}_{ij}^{K}(N)}{\hat{\mu}_{ij}^{K}(N)}\right)} \cdot \left(\frac{1}{\kappa\left(\frac{\Delta}{\tau_{ij}}\right)}\right) \cdot F\left(\frac{t - t_m^K - d_{ij}}{\tau_{ij}} + \psi\left(\frac{\Delta}{\tau_{ij}}\right) + \frac{\hat{U}_{ij}^{K}(N)}{\hat{\mu}_{ij}^{K}(N)}\right) \tag{4.21}$$

$$\mu_{ij}(t) = \left( \frac{\hat{\mu}_{ij}^{K}(N)}{1 - e^{-\left(\frac{\Delta}{\tau_{ij}}\right)}} \right) \cdot G\left( \frac{t - t_m^K - d_{ij}}{\tau_{ij}} \right) \tag{4.22}$$

where the scaling function of $\kappa$ used in Equation (4.21) is defined in Equation (C.13). Thus, Equation (4.21) represents a general solution for any periodic spike train. The solution for $\mu_{ij}(t)$ is not needed to integrate the neuron dynamics forward in time, but is useful for specifying a neuron's initial condition when starting a simulation from a periodic spiking condition.

For the special case of only one spike per period, ($P^1 = \Delta$), $\hat{\mu}_{ij}^{K}(1)$ and $\hat{U}_{ij}^{K}(1)$ reduce to 1 and 0 respectively. Thus, $U_{ij}(t)$ and $\mu_{ij}(t)$ become:

$$U_{ij}(t) = \left( \frac{1}{\kappa\left(\frac{\Delta}{\tau_{ij}}\right)} \right) \cdot F\left( \frac{t - t_m^K - d_{ij}}{\tau_{ij}} + \psi\left(\frac{\Delta}{\tau_{ij}}\right) \right) \tag{4.23}$$

$$\mu_{ij}(t) = \left( \frac{1}{1 - e^{-\left(\frac{\Delta}{\tau_{ij}}\right)}} \right) \cdot e^{-\left(\frac{t - t_m^K - d_{ij}}{\tau_{ij}}\right)} \tag{4.24}$$

Figure 4.2 shows $U_{ij}(t)$ for three different values of $\frac{\Delta}{\tau_{ij}}$. Also, applying the properties of $F(x)$, $\psi(x)$, and $\kappa(x)$ found in Appendices A & C to these equations give:

$$\max\{U_{ij}(t)\} = \frac{e^{-1}}{\kappa\left(\frac{\Delta}{\tau_{ij}}\right)} \quad \text{at:} \quad t = t_m^K + d_{ij} + \tau_{ij}\left[1 - \psi\left(\frac{\Delta}{\tau_{ij}}\right)\right] \tag{4.25}$$

$$\min\{U_{ij}(t)\} = \frac{F\left(\psi\left(\frac{\Delta}{\tau_{ij}}\right)\right)}{\kappa\left(\frac{\Delta}{\tau_{ij}}\right)} = \frac{\psi\left(\frac{\Delta}{\tau_{ij}}\right)}{1 - e^{-\frac{\Delta}{\tau_{ij}}}} \quad \text{at:} \quad t = t_m^K + d_{ij} \tag{4.26}$$

$$\langle U_{ij}(t) \rangle = \frac{1}{\Delta} \int_{t_m^K + d_{ij}}^{t_m^K + d_{ij} + \Delta} U_{ij}(t)dt = \frac{\tau_{ij}}{\Delta} \tag{4.27}$$

$$\tag{4.28}$$

$$\max\{\mu_{ij}(t)\} = \frac{1}{1 - e^{-\left(\frac{\Delta}{\tau_{ij}}\right)}} \quad \text{at:} \quad t = t_m^K + d_{ij} \tag{4.29}$$

$$\min\{\mu_{ij}(t)\} = \left(\frac{\tau_{ij}}{\Delta}\right) \cdot \psi\left(\frac{\Delta}{\tau_{ij}}\right) \quad \text{at:} \quad t = t_m^K + d_{ij} + \Delta \tag{4.30}$$

$$\langle \mu_{ij}(t) \rangle = \frac{1}{\Delta} \int_{t_m^K + d_{ij}}^{t_m^K + d_{ij} + \Delta} \mu_{ij}(t)dt = \frac{\tau_{ij}}{\Delta} \tag{4.31}$$

Equation (4.27) shows that the average amount of neurotransmitter varies directly

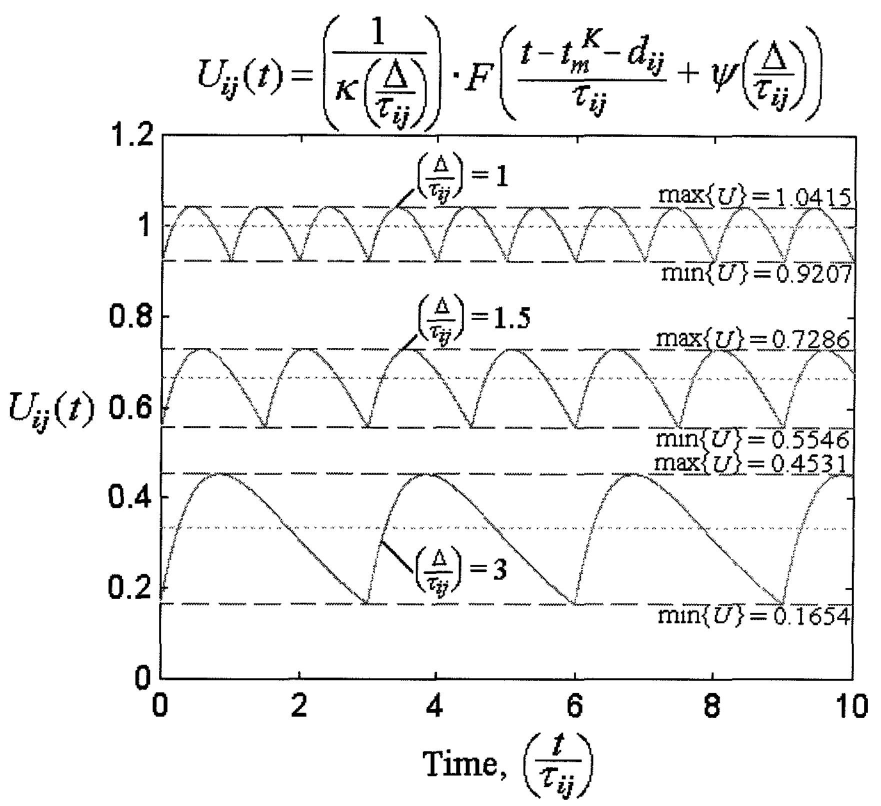


Figure 4.2: $U_{ij}(t)$ **for Periodic Spike Pattern with** $P^1 = \Delta$. This plot shows $U_{ij}(t)$ for three different spike frequencies ($\frac{\Delta}{\tau_{ij}}$ = 1, 1.5, and 3). As the frequency is increased, the average value also increases, and the "bumps" become smaller and more symmetrical. The maximum and minimum values are found using Equations (4.25)-(4.26) and are shown as dashed lines, while the average values are given by Equation (4.27) and are shown as dotted lines.

with the input frequency, $f^{\text{in}} = \frac{1}{\Delta}$, and the integral of the $U_{ij}(t)$ function between spikes equals the total contribution from a single spike (see Equation (3.6)).

## 4.3 Neuron Potential, $V_i(t)$

### 4.3.1 Introduction

Before trying to use the SNM for computational tasks, it is informative to compare its behavior with that of a biological neuron. In real neurons the amplitude and width of an action potential is independent of stimulus intensity; however, many of its other properties are not. In particular, the latency, which is the time delay from the onset of the stimulus to the peak of the action potential, and the refractory period, which is the time required after a spike before another one can be generated, are both functions of stimulus strength. They allow neurons to encode the strength of a stimulus in terms of output spike frequency. (Chapter 8 discusses how information may actually be encoded in spikes.) Also, some neurons have endogenous spiking behaviors, and are capable of producing periodic spike trains in the absence of any external stimulation. Other neurons are able to adapt their output spike rate to become more sensitive to changes within the input signal rather than the input's intensity.

All of these phenomena are properties of the neuron potential. Section 4.3.2 and Section 4.3.3 discuss latency and refractory periods in the SNM. Section 4.3.4 shows how the SNM parameters may be set to produce endogenous spike train outputs. And Section 4.3.5 shows how a neuron can adapt its spike rate to a constant input signal.

### 4.3.2 Latency

Figure 4.3 shows a typical response when a depolarizing (excitatory) current is injected into a real neuron. Notice that a minimum signal strength is required to produce an action potential, and when a stimulus exceeds this threshold, the time delay from the onset of the stimulus to the peak of the action potential, decreases with stimulus strength. The latency, which is the time delay before an action potential is generated, can be estimated for a real neuron by assuming that it responds only passively before reaching threshold and generating an action potential. The neuron's membrane can be modeled as a resistor and capacitor in parallel. When a step current is injected into the neuron, the membrane potential of the neuron, $V_m$, responds as:

$$V_m = V_r + RI^{\text{in}} \cdot \left(1 - \mathrm{e}^{-\left(\frac{t}{\tau}\right)}\right) \tag{4.32}$$

$$\text{where:} \quad \tau = RC \tag{4.33}$$

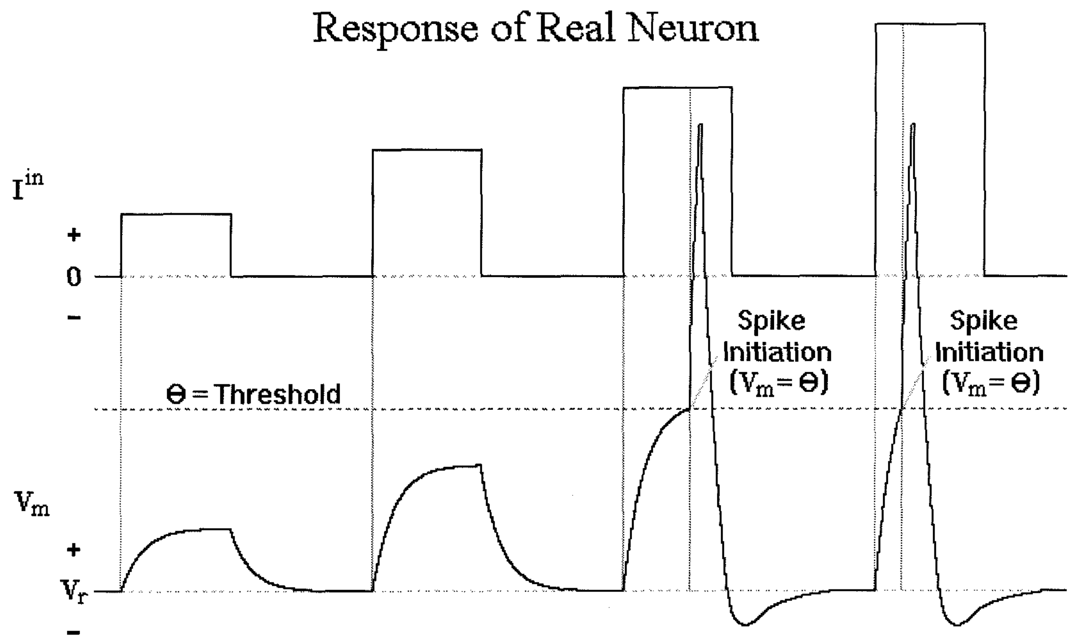


Figure 4.3: **Response of Real Neuron to Depolarizing Stimuli.** The top plot shows the stimulus current, $I$, that was injected into the cell via an electrode, while the bottom plot shows the resulting change in membrane potential. When the membrane potential reaches its threshold, an action potential is generated.

Here $V_r$ is the neuron's resting potential and $I^{\text{in}}$ is the input stimulus. Since the neuron will not fire unless $V_m$ reaches threshold, $\Theta$, it is necessary that:

$$I^{\text{in}} > \frac{\Theta - V_r}{R} \tag{4.34}$$

to generate an action potential. If this condition is satisfied, then the neuron will reach its threshold potential at time:

$$t_\Theta = \tau \ln \left[ \frac{RI^{\text{in}}}{RI^{\text{in}} + V_r - \Theta} \right] \tag{4.35}$$

Of course, the injected current must last longer than $t_\Theta$ for the neuron to reach $\Theta$ and produce a spike. Notice that while this derivation only provides a rough approximation to the complex biochemical response of a neuron to stimuli, the neuron can be thought of as logarithmically encoding the strength of the input signal within the time delay of its response.

Now, since the SNM is a qualitative "high level" model and does not attempt to account for the individual currents attributable to specific ions, it is not possible to directly simulate the injection of an excitatory or inhibitory current into the neuron. However, since functioning neurons do not have currents directly injected into them, but rather receive inputs from other neurons or external stimuli, this is not seen as a limitation of the model. To demonstrate latency in the SNM, a periodic spike train stimulus generated from another neuron can be used.[3] When the stimulus has one spike per period, $(P^1 = \Delta)$, the average amount of input neurotransmitter is inversely proportional to the time between pulses (see Equation (4.27)).

Figure 4.4 shows the results from a sudden onset of input spikes, which are equally spaced. As with biological neurons, when the SNM is sufficiently stimulated to produce an output spike, the time delay between the start of the stimulus and the spike output depends upon the stimulus strength. While this latency phenomenon can be seen qualitatively in the figure, a more detailed analysis of latency in the SNM is presented in Appendix E.

## 4.3.3 Refractory Period

After a real neuron fires, there is a short time period during which it is unable to produce another spike, regardless of the strength of the input stimulus. It usually lasts several milliseconds, and is referred to as the absolute refractory period. After a sufficient length of time has passed, the neuron becomes capable of generating another spike; however, the stimulus must be larger than normal. The period of time during

[3]In the SNM, external inputs do *not* have latency. When the input signal is a spike from another neuron, the voltage changes as $\tanh(te^{-t})$. Thus, its effect is not instantaneous, but rather increases from zero to its maximum level at $t = 1$ before decaying back to zero. This delayed effect is attributed to the diffusion of neurotransmitter across the synaptic gap. But when an input comes from an external stimuli, such as light into the ganglion cells of the retina, there is no diffusion of neurotransmitter. In real neurons, a sudden change in the external input does not have an instantaneous effect due to the required opening of ionic channels. However, the neuron is usually able to respond more quickly to the directly input external stimuli than it is to spikes from other neurons. To simplify the calculations, the SNM approximates this quicker response with an instantaneous one; i.e., the voltage only depends upon $\tanh(I(t))$. To correctly model the response would require calculating the convolution of the external input signal with the neuron's impulse response function. (Notice that $U_{ij}(t)$ can be derived by convoluting a spike train of unit impulse functions with $te^{-t}$.)

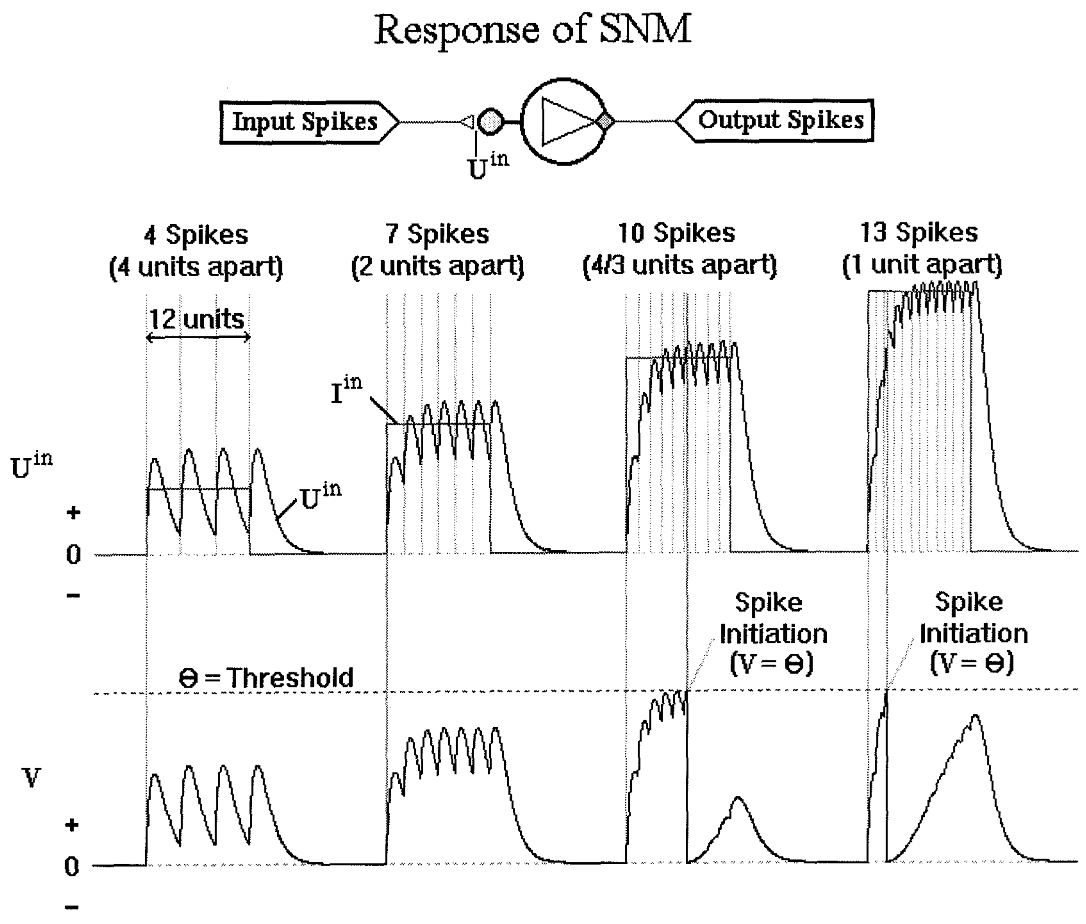


Figure 4.4: **SNM Response to Periodic Input.** The top plot shows the neurotransmitter concentration, $U^{\text{in}}$, at the input synapse, and the spike train which generated it. The spikes were used to simulate the four levels of input current that were injected into the real neuron in Figure 4.3, ($I^{\text{in}} = 0.25, 0.5, 0.75, 1.0$). The bottom plot shows the resulting change in the neuron's potential. When the potential reaches threshold, an output spike is produced. (This plot represents an actual simulation with: $\tau_i = 8.0; R_{i0} = 0.0; R_{ij} = 1.0; T_{ij} = 1.0; \tau_{ij} = 1.0; d_{ij} = 0.0; \Theta_i = 0.665$. The duration of each of the four stimuli lasted for 12 units, with the time between input spikes being: $P^1 = 4, 2, \frac{4}{3}, 1$ units, respectively. The time between each period of stimulation was 16 units with the entire figure representing 118 units.)

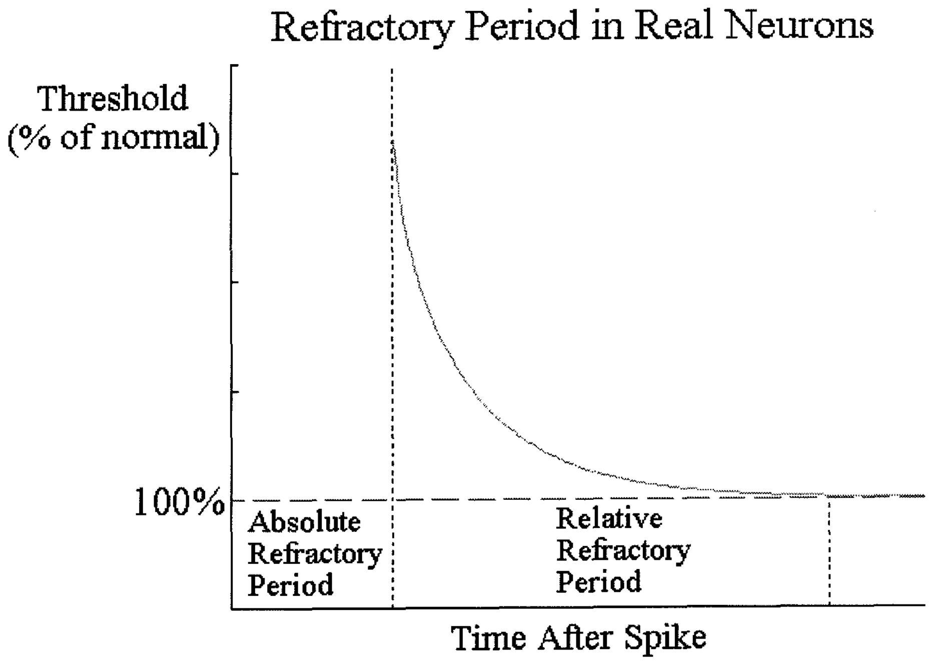


Figure 4.5: **Refractory Period in Real Neurons.** The curve represents the stimulus strength required to generate a second spike as a function of time after the first spike. During the absolute refractory period, which occurs immediately after the firing of the first spike, the neuron can not produce another spike regardless of the strength of the stimulus. As a neuron begins to recover, it enters the relative refractory period, where it regains its spiking capability, but the effective threshold required to generate a new spike is elevated above normal. (Based on Fig. 2-5 in [65].)

which a larger stimulus is required is referred to as the relative refractory period. Figure 4.5 demonstrates these ideas for a real neuron.

Like a real neuron, the SNM has both an absolute and relative refractory period. This is a consequence of the $\tanh^2\left(\frac{t-t_i^K}{\tau_i}\right)$ term[4] in Equation (3.1), which becomes zero immediately after a spike is produced and then increases towards one. In fact, the time between the $K$th and $(K-1)$th output spikes, $\Delta_i^K$, is given by:

$$\Delta_i^K \equiv t_i^K - t_i^{K-1} = \tau_i \cdot \text{Atanh}\left(\sqrt{\frac{\Theta_i}{Z_i(t_i^K)}}\right) \tag{4.36}$$

$$\text{where:} \quad Z_i(t) \equiv R_{i0} + \sum_{j=1}^{N_i} R_{ij} \cdot \tanh\left(\frac{T_{ij}}{R_{ij}} \cdot U_{ij}(t)\right) + \sum_{x=1}^{P_i} R_{ix} \cdot \tanh\left(T_{ix} \cdot I_{ix}(t)\right) \tag{4.37}$$

[4]The $\tanh^2\left(\frac{t-t_i^K}{\tau_i}\right)$ term will be referred to as the "refractory coefficient" throughout this thesis.

Here, $Z_i(t)$ represents the total input signal (including the neuron's resting potential, $R_{i0}$). Notice that Equation (4.36) only has a solution if:

$$\Theta_i < R_{i0} + \sum_{j=1}^{N_i} R_{ij} \cdot \tanh\left(\frac{T_{ij}}{R_{ij}} \cdot U_{ij}(t_i^K)\right) + \sum_{x=1}^{P_i} R_{ix} \cdot \tanh\left(T_{ix} \cdot I_{ix}(t_i^K)\right) \tag{4.38}$$

which is consistent with the fact that a neuron can not fire unless its input signal is above threshold.

The absolute refractory period for the SNM is the minimum possible value of $\Delta_i^K$, which occurs when the denominator of Equation (4.36) reaches its maximum value. This denominator represents what the neuron's potential voltage would be, if it had not previously fired (see Equation (3.1) with $\tanh^2\left(\frac{t-t_i^K}{\tau_i}\right) \approx 1$ as $t \to \infty$). Since the value of tanh (−) is limited to one, the maximum possible potential voltage is:

$$\max\{Z_i(t)\} < R_{i0} + \frac{1}{2}\sum_{j=1}^{N_i} R_{ij}\left(\operatorname{sgn}(T_{ij}) + 1\right) + \frac{1}{2}\sum_{x=1}^{P_i} R_{ix}\left(\operatorname{sgn}(T_{ix}) + 1\right) \equiv R_i^+ \tag{4.39}$$

$$\text{where:} \quad \operatorname{sgn}(x) \equiv \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases} \tag{4.40}$$

Thus, $R_i^+$ is defined as $R_{i0}$ plus the sum of the $R_{ij}$ and $R_{ix}$ terms which correspond to excitatory inputs. (Excitatory inputs have positive values for $T_{ij}$ or $T_{ix}$.) Now, the absolute refractory period is given by:

$$\min\left\{\Delta_i^K\right\} = \tau_i \cdot \operatorname{Atanh}\left(\sqrt{\frac{\Theta_i}{R_i^+}}\right) \tag{4.41}$$

Notice that since the actual potential voltage due to the input synapses must always be less than $R_i^+$, this limit for $\Delta_i^K$ can only be asymptotically approached. The minimum value for $\Delta_i^K$ is plotted in Figure 4.6 as a function of $\frac{R_i^+}{\Theta_i}$.

In addition to enforcing an absolute refractory period, the $\tanh^2\left(\frac{t-t_i^K}{\tau_i}\right)$ term in Equation (3.1) also creates a relative refractory period. In fact, the effective threshold

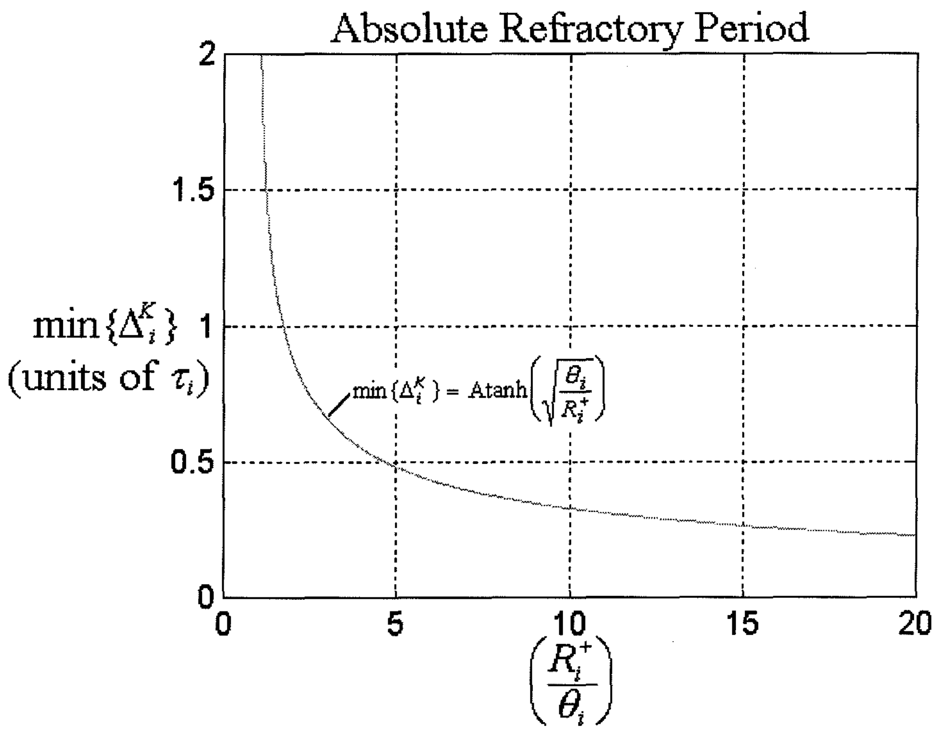


Figure 4.6: **Absolute Refractory Period.** This curve shows the absolute refractory period as a function of the maximum possible excitatory input stimulus divided by the threshold voltage. It represents the minimum possible time between output spikes.

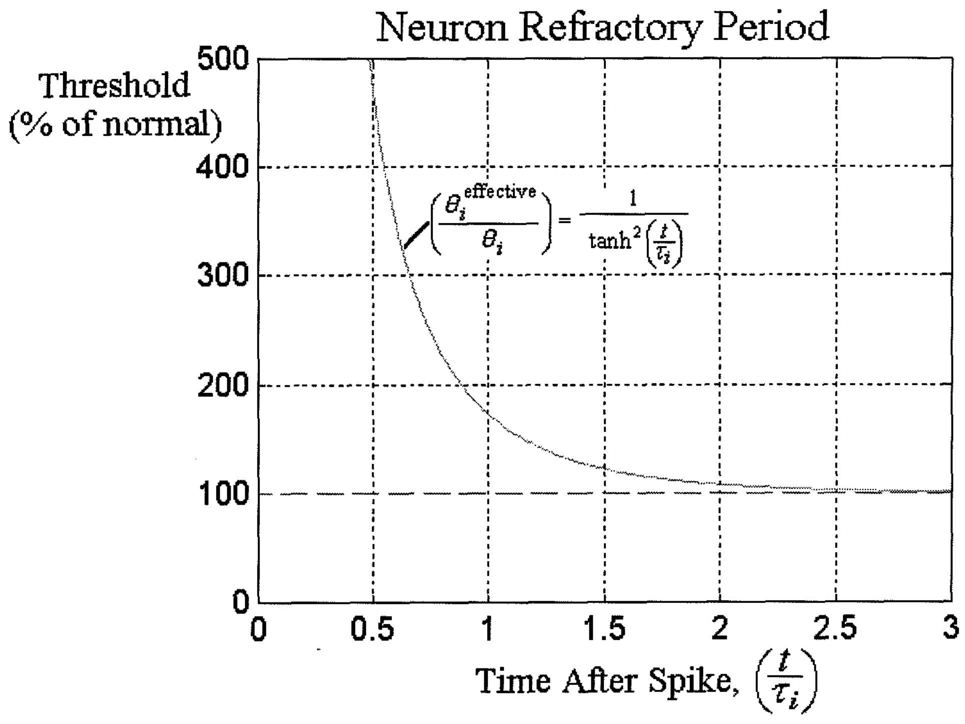


Figure 4.7: **Neuron Refractory Period.** This curve shows the effective spiking threshold as a function of the elapsed time after an initial spike. It represents the stimulus strength required to generate a second spike.

can be thought of as:

$$\Theta_i^{\text{effective}} \equiv \frac{\Theta_i}{\tanh^2\left(\frac{t-t_i^K}{\tau_i}\right)} \tag{4.42}$$

with a neuron only producing a spike when $Z_i(t)$ is greater than $\Theta_i^{\text{effective}}$. The curve for the effective threshold function is plotted in Figure 4.7. It has similar characteristics to the refractory period of real neurons shown in Figure 4.5.

Thus, the SNM has similar refractory properties to real neurons; however, it is important to make a distinction between them. The absolute refractory period in the SNM is a function of the strengths of the input synapses, while in real neurons it is not. The refractory period in a biological neuron is determined by inserting an electrode and injecting current. The absolute refractory period is defined as the minimum time between output spikes for which increases in the input current do not matter. It is a consequence of the time required for the ionic channels to reset themselves before the neuron can produce another pulse. Since the refractory period is measured via an electrode controlling the neuron's potential, it is independent of the neuron's synapses. Injecting a larger current only damages the cell. Since there is no analogous way to inject current into the SNM, the absolute refractory period is defined as the time between spikes when all of the excitatory inputs are at their asymptotic maximum levels. Thus, it represents the minimum possible period between spikes for a neuron when it is used within a network. Unlike real neurons, as the weights for the input stimuli increase, the absolute refractory period decreases. Section 8.4 discusses how the SNM may be modified to include a true absolute refractory period that does not depend upon the input weights. This modification is necessary for the logarithmic encoding scheme discussed; however, it increases the complexity of the neuron dynamics by requiring two distinct operating regions. For nearly all applications, this additional complication is unnecessary, and the standard SNM is used.

### 4.3.4 Spiking Behavior

Real neurons are remarkably diverse in their electrical and biochemical properties. One example of this diversity is exhibited in the patterns of endogenous action potential firing. Many neurons do not fire spontaneously at all, while others beat or burst in a regular manner. Fortunately, the SNM is general enough to reproduce all of these types of spiking behavior, as shown in Figure 4.8.

Neurons that have a steady unchanging resting potential in the absence of external stimulation are called silent neurons.[5] Any neuron with its threshold voltage, $\Theta_i$, greater than its resting potential, $R_{i0}$, is a silent neuron. (See Figure 4.8(A).)

Some neurons fire repetitively at a constant frequency and are called pacing neurons. Their spike pattern is described as beating. While external stimulation can change the firing rate or inhibit it altogether, these neurons do not require external

[5]Silent neurons are usually capable of producing an action potential when presented with an excitatory stimulus; however, some are not, and are referred to as being electrically inexcitable. They carry out their electrical signaling through graded potentials and communicate via gap junctions. The SNM does not consider neurons of this type.

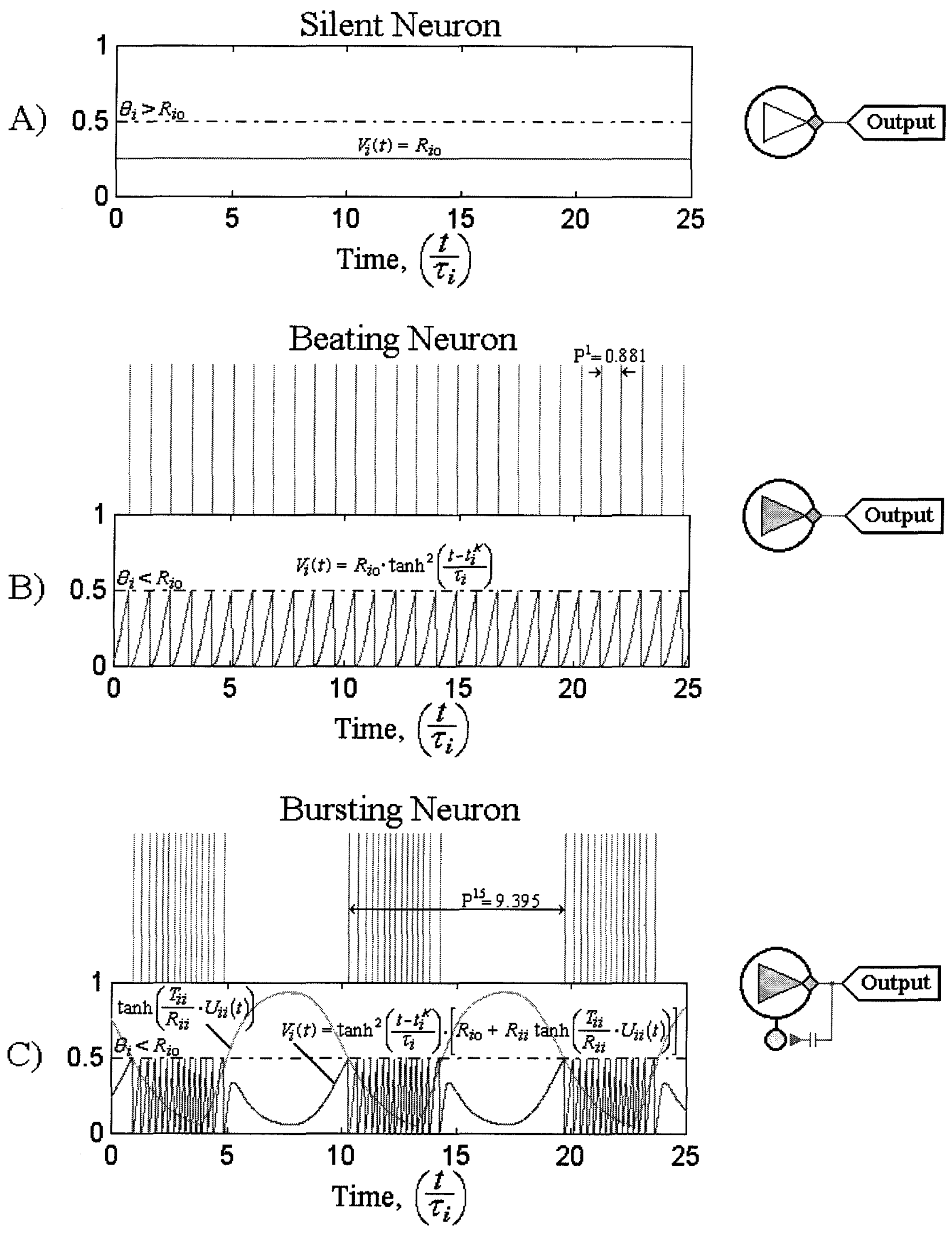


Figure 4.8: **Endogenous Neuron Activity.** (A) shows a silent neuron with $\Theta_i = 0.5$ and $R_{i0} = 0.25$. (B) shows a beating neuron with $\Theta_i = 0.5$, $R_{i0} = 1.0$, and $\tau_i = 1.0$. (C) shows a bursting neuron with $\Theta_i = 0.5$, $R_{i0} = 1.0$, $\tau_i = 0.25$, $R_{ii} = 1.0$, $T_{ii} = -0.5$, $d_{ii} = 3.0$, and $\tau_{ii} = 1.0$.

stimulation to drive their repetitive firing. This type of behavior is exhibited when the threshold voltage, $\Theta_i$, is less than the resting potential, $R_{i0}$ (see Figure 4.8(B)). Once the neuron fires, the refractory period prevents it from immediately reproducing another spike. The time between spikes is given by:

$$t_i^K - t_i^{K-1} = \tau_i \cdot \mathrm{Atanh}\left(\sqrt{\frac{\Theta_i}{R_{i0}}}\right) \tag{4.43}$$

Another type of neuron that fires spontaneously in the absence of external stimulation generates bursts of spikes that are separated by periods of inactivity. These neurons are called bursting neurons and are often used to generate rhythmic behaviors. It is clear from a systems point of view that these neurons have two underlying oscillations, one sets the frequency of spikes during the bursting period, while the other sets the frequency of the bursting periods. (For this discussion, any periodic spike pattern with more than one spike per period is considered bursting. See the beginning of Section 4.2.3 for a description of periodic spike patterns and their notation.) Thus, to produce such behavior the neuron requires two feedback signals: one must be fast to control the individual spikes, while the other must be slow to control the bursting cycles. In real neurons, this is accomplished through the temporal properties of specific ion conductances, with some having time constants in the milliseconds range, while others operate over minutes [52, pp. 261-268]. Similarly, the refractory period sets the time between spikes for the SNM, and the slow feedback signal can easily be realized by utilizing an inhibitory self-connection with delay; i.e., let $T_{ii} < 0$ and $d_{ii} > 0$.[6] (In the SNM, the second subscript refers to the number of the synapse on the receiving neuron, which does not necessarily correspond to the number of the neuron connected to that synapse; i.e., $m = \{N_i\}_j \neq j$. However, to simplify the notation, the double subscript of $ii$ will refer to a self-connection; i.e.,

[6]It is highly unlikely that real neurons use self-connecting synapses with delay, but its effects can be duplicated by stimulating an inhibitory neuron which then connects back to the original neuron. Obviously, this can be modeled using two SNMs, but with the added cost of simulating the dynamics of another neuron. Since the purpose is to construct computing devices and not to model biology, this less costly approach is used.

$m = \{N_i\}_i = i$.) Since the refractory period term, $\tanh^2\left(\frac{t-t_i^K}{\tau_i}\right)$, sets the neuron's potential to zero after a spike is produced, its effect is immediate, but the effect of the self connection with delay is not instantaneous. At the inhibitory synapse, the contribution of neurotransmitter, $U_{ii}(t)$, from each spike takes the form of $f(t) = te^{-t}$ (see Figure A.1), and the delay, $d_{ii}$, allows the neuron to fire several times (burst), before the feedback signal takes effect and inhibits the neuron from firing again. When this occurs, the neuron stops firing, and there are no more contributions to the feedback synapse. As feedback neurotransmitter, $U_{ii}(t)$, diminishes, the neuron is once again able to reach threshold and produce another burst of spikes and repeat the process (see Figure 4.8(C)).

One important property of neurons with an inhibitory self-connection is that they are capable of producing *both* beating and bursting behavior. As long as $0 < \Theta_i < R_{i0}$, the SNM will converge to a periodic spiking output behavior, but the number of spikes in each period, $N$, and the length of each period, $\Delta$, depend upon the parameter values *and* the initial condition. A neuron with an inhibitory synapse may have several stable limit cycles, and the one it converges to depends upon the basin of attraction into which it falls. Figure 4.9 shows two neurons with the same parameters, but one is beating ($P^1 = 0.915$), and the other is bursting ($P^{15} = 9.395$). As a consequence, a neuron can be perturbed by external stimuli from one limit cycle into the basin of attraction of another limit cycle, thus allowing a neuron to switch between different output behaviors, with no change in any of its parameters. Such switching behavior is known to exist within some groups of neurons in biology [62]. As a much simplified example, consider the neurons controlling the muscles responsible for chewing. With a beating spike train output, these neurons can keep the muscles slightly contracted and the mouth closed. When presented with food, a brief command signal might be used to perturb the neurons' behavior [62] and cause them to produce a bursting output, resulting in chewing. The neurons would continue their bursting behavior until another command signal is input, which moves them back into the beating limit cycle's basin of attraction.[7] (See [65, Chapter 16] for a brief description of some known biological neural networks that control behavior.)

The most critical parameter in determining the possible types of output behavior for a neuron is the delay parameter, $d_{ii}$. In general, when $d_{ii}$ is small compared to the other time constants, the neuron is not capable of bursting – only beating. As $d_{ii}$ increases, the number of possible spikes per period also increases. Due to the transcendental nature of the equations governing the spike period in the SNM, it is difficult to quantify the dependence upon $d_{ii}$, but Figures 4.10 and 4.12 demonstrate how the output spike patterns may change with $d_{ii}$ for one set of neuron parameters.

In Figure 4.10 the neuron is started from a "beating initial condition," which means that the neuron is assumed to be already beating before the simulation begins. To determine the beat frequency, $\Delta$, Equations (4.23) and (3.1) are used with $t - t_i^K = \Delta$ and $V_i(t)$ set equal to $\Theta_i$, i.e.:

$$\Theta_i = V_i(\Delta) = \tanh^2\left(\frac{\Delta}{\tau_i}\right) \cdot \left[R_{i0} + R_{ii} \tanh\left(\frac{T_{ii}}{R_{ii}} \cdot U_{ii}(\Delta)\right)\right] \tag{4.44}$$

where:

$$U_{ii}(\Delta) = \left(\frac{1}{\kappa\left(\frac{\Delta}{\tau_{ii}}\right)}\right) \cdot F\left(\frac{n\Delta - d_{ii}}{\tau_{ii}} + \psi\left(\frac{\Delta}{\tau_{ii}}\right)\right) \tag{4.45}$$

$$n = \left\lceil \frac{d_{ii}}{\Delta} \right\rceil \tag{4.46}$$

Notice in Equation (4.23) that $t - t_i^k - d_{ii}$ must be greater than zero, and since $t$ represents the time of the current spike in Equation (4.45), the difference of $t - t_i^k$ must be an integer multiple of $\Delta$. Hence, $n$ represents the number of spikes (including the current spike) that have occurred since the most recent spike contributing to the neurotransmitter was produced; i.e., the effects from the last $n$ spikes have not yet contributed to the current amount of neurotransmitter, $U_{ii}(\Delta)$. (If $d_{ii} = 0$, then $n$

[7] A neural network that produces a rhythmic motor output is often referred to as a central pattern generator (CPG). CPGs are used to control repetitive behaviors like chewing, swimming, or walking.

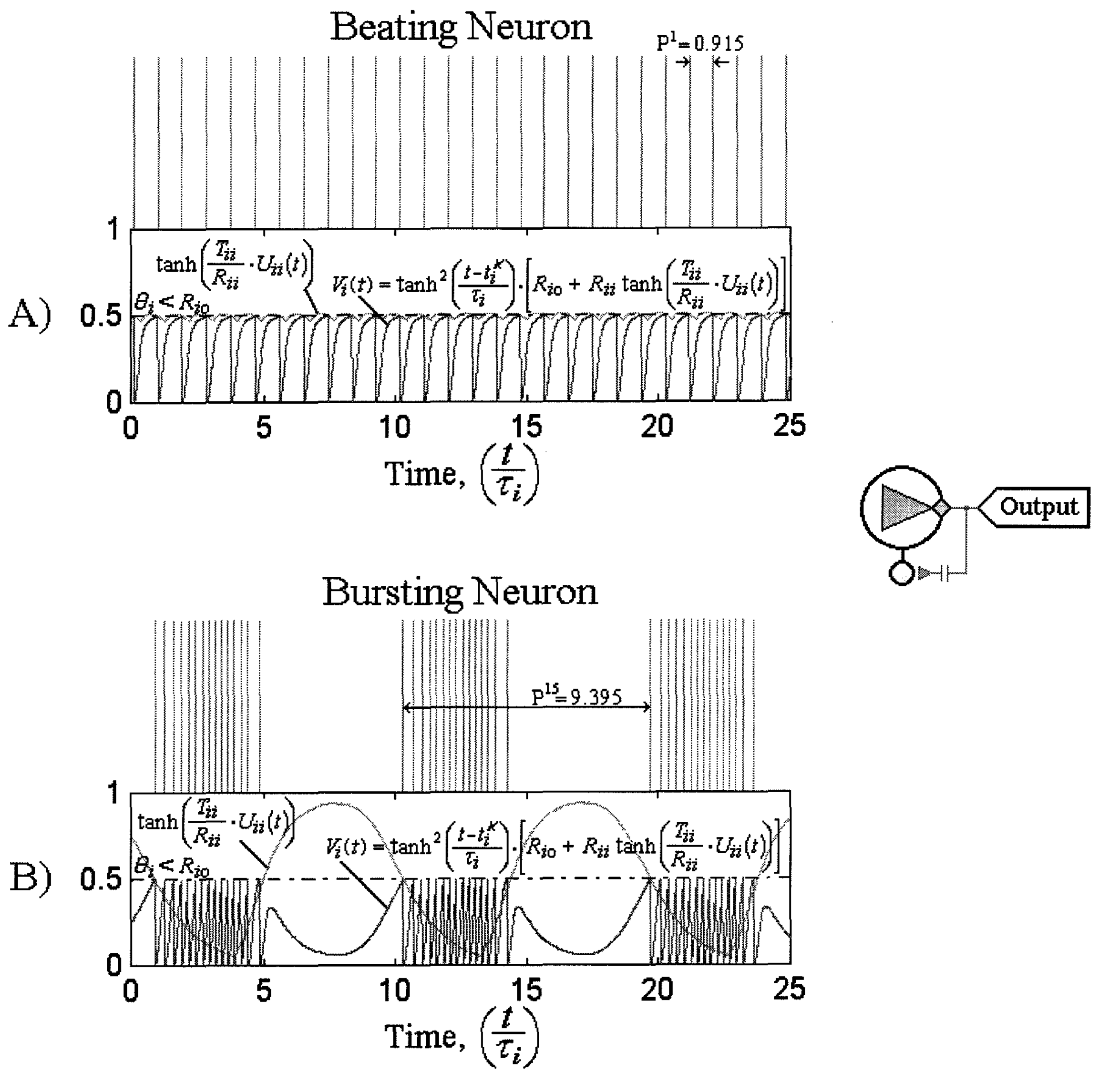
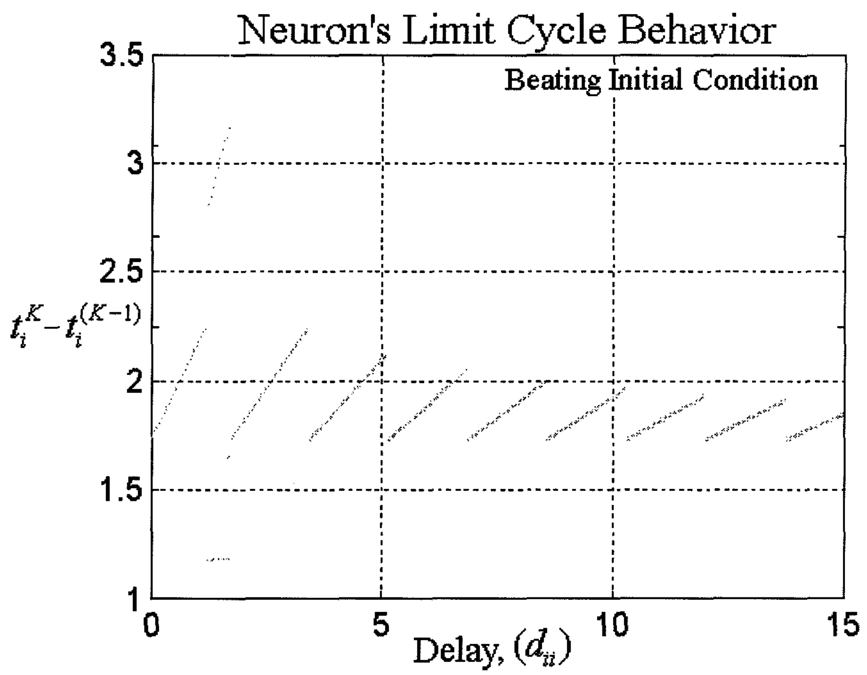


Figure 4.9: **Beating and Bursting Behavior.** This figure shows two neurons with the same parameters, ($\Theta_i = 0.5$, $R_{i0} = 1.0$, $\tau_i = 0.25$, $R_{ii} = 1.0$, $T_{ii} = -0.5$, $d_{ii} = 3.0$, and $\tau_{ii} = 1.0$), but one is beating and the other is bursting. The beating neuron in (A) has $P^1 = 0.915$, while the bursting neuron in (B) has $P^{15} = 9.395$.



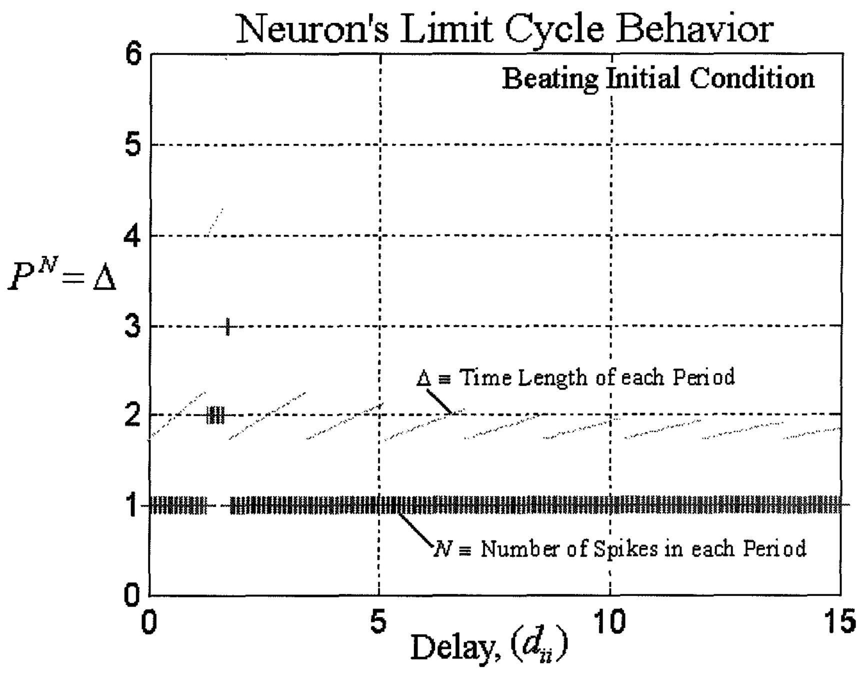


Figure 4.10: **Beating Limit Cycle Behavior.** This figure shows the results for a neuron started from a beating initial condition. (A) shows how the time between spikes changes with the delay, $d_{ii}$. The other parameters were: $\Theta_i = 0.5$, $R_{i0} = 1.0$, $\tau_i = 1.0$, $R_{ii} = 1.0$, $T_{ii} = -1.0$, and $\tau_{ii} = 1.0$. (B) shows the number of spikes in each period, $N$, and the length of each period, $\Delta$. The neuron converged to a beating behavior for all values of $d_{ii}$ between 0 and 15, except when $1.25 \leq d_{ii} \leq 1.60$, there were 2 spikes per period, and when $1.65 \leq d_{ii} \leq 1.70$, there were 3 spikes per period.

must be set to 1.)

If $R_{i0} > \Theta_i$, there exists at least one value for $\Delta$ that solves this set of equations. Figure 4.11 shows $V_i(\Delta)$ and $U_{ii}(\Delta)$ of Equations (4.44)-(4.46) as a function of $\Delta$ for one set of parameters. After numerically approximating the minimal solution for $\Delta$, the neuron was assumed to have produced a spike at $t = 0$, and the neuron dynamics of Equations (3.1)-(3.4) were integrated forward in time, with the amount of neurotransmitter at the inhibitory synapse due to all previous spikes given by Equation (4.23). The dynamics were integrated until $t = 100$, and Figure 4.10 shows the time between spikes for all output spikes that occurred between $t = 25$ and $t = 100$. (A beginning time of 25 was used to allow the neuron to converge to its limit cycle, if different from the initial beating condition.) For nearly all $0 \leq d_{ii} \leq 15$, the neuron continued to beat; however, for $1.25 \leq d_{ii} \leq 1.70$, the neuron converged to limit cycles with 2 or 3 spikes per period, thus indicating that there was no stable beating behavior for this range of $d_{ii}$.

Although not shown in the figures, when there was more than one solution for $\Delta$ in Equations (4.44)-(4.46), the dynamic equations were also integrated forward from these other beating initial conditions. (For most of the values for $d_{ii}$ there was only one solution, but the occurrence of multiple solutions for $\Delta$ increases with $d_{ii}$.) For some solutions, the neuron converged to the beating behavior shown in Figure 4.10, and for other solutions the neuron converged to the bursting behavior shown in Figure 4.12, but for many solutions, the neuron converged to yet another bursting behavior.

In Figure 4.12 the neuron was started from a "zero initial condition," which means that the neuron was assumed to have produced an output spike at $t = 0$, but there was no previous neurotransmitter at the inhibitory feedback synapse, ($U_{ii}(t \leq 0) = 0$). The dynamics were integrated until $t = 150$, and after an initial transient period, the neuron converged to a bursting limit cycle. Figure 4.12 shows the time between spikes for all output spikes that occurred between $t = 75$ and $t = 150$. In general, the number of spikes and length of each period increased with $d_{ii}$, with only beating behavior possible when $0 \leq d_{ii} \leq 1.15$.

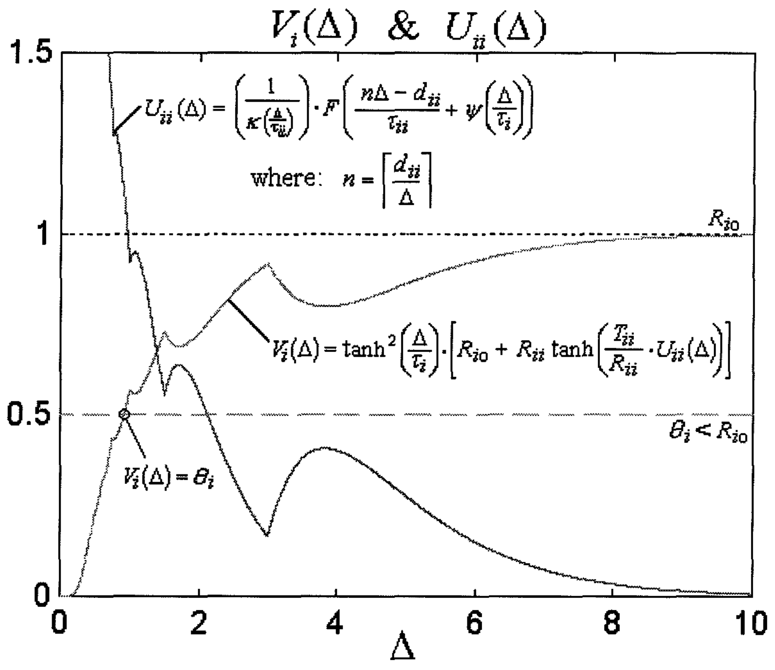
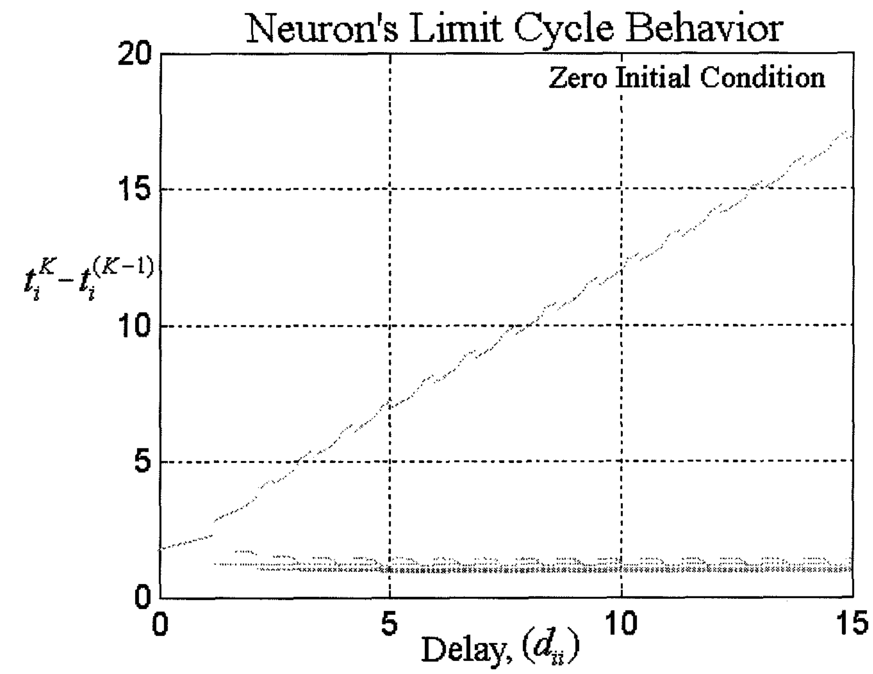


Figure 4.11: **Plot of $V_i(\Delta)$ and $U_{ii}(\Delta)$.** The plot shows $V_i(\Delta)$ and $U_{ii}(\Delta)$ of Equations (4.44)-(4.46) as a function of $\Delta$ for the parameter values of: $R_{i0} = 1.0$, $\tau_i = 0.25$, $R_{ii} = 1.0$, $T_{ii} = -0.5$, $d_{ii} = 3.0$, and $\tau_{ii} = 1.0$. Since $V_i(\Delta)$ goes from zero to $R_{i0}$ as $\Delta$ increases, any $\Theta$ less than $R_{i0}$ must yield at least one solution for $\Delta$. The dips in the curve are where the value of $n$ changes ($3 \leq \Delta \rightarrow n = 1$; $1.5 \leq \Delta < 3 \rightarrow n = 2$; $1 \leq \Delta < 1.5 \rightarrow n = 3$; etc.) A dashed line is drawn for $\Theta = 0.5$, which corresponds to the value used in Figure 4.10. Notice that for some choices of $\Theta$ there can be more than one solution.



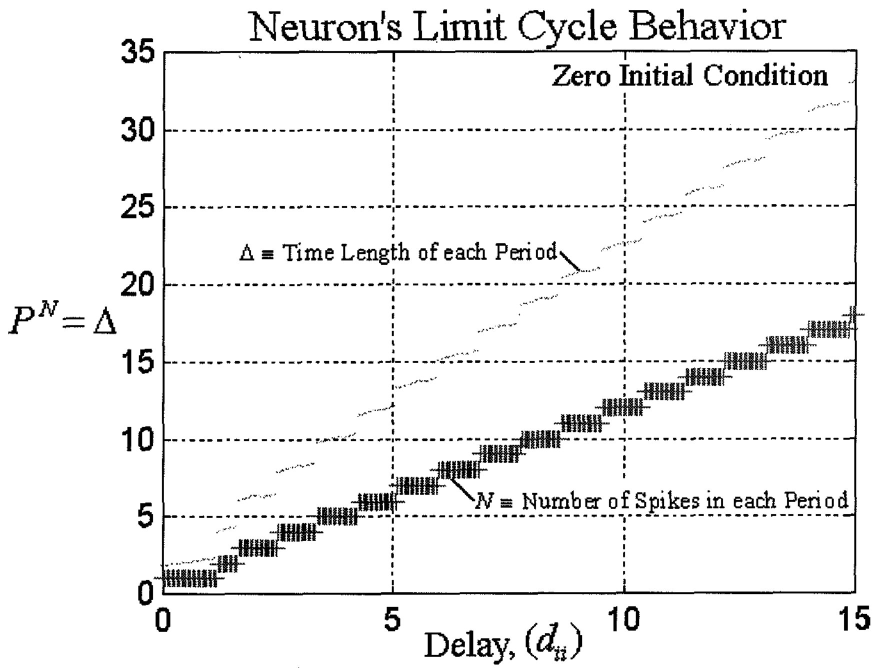


Figure 4.12: **Bursting Limit Cycle Behavior.** This figure shows the results for a neuron started from a zero initial condition. Except for the different initial condition, the neuron was the same as in Figure 4.10. (A) shows how the time between spikes changes with the delay, $d_{ii}$. The multiple data points for each value of $d_{ii}$ reflect the number of spikes per period. (B) shows the number of spikes in each period, $N$, and the length of each period, $\Delta$. Notice that the number of spikes increased at regular intervals as the delay was increased.

### 4.3.5 Adapting Behavior

Another important property that real neurons are known to exhibit is the ability to adapt their spike rate to a constant input signal [53, p. 337]; i.e., when an external input signal first starts, it triggers a rapid burst of spikes from the neuron, which then decays into a slower beating behavior. Adaptation may be caused by a change in the synaptic strengths [56], but the SNM is capable of modeling such behavior with only a feedback connection, and no changes in the parameters are required. Of course, the adaptation is not permanent and the neuron returns to its initial state after a relatively short time. This is in contradistinction to real neurons, where the synaptic modifications attributable to long-term potentiation (LTP) or long-term depression (LTD) can last for days or weeks [106]. (Chapter 9 discusses how the parameters can be permanently modified through learning.)

Figure 4.13 shows how the SNM is able to duplicate this type of adaptation with an inhibitory feedback connection, which is similar to the one used for eliciting bursting behavior in Section 4.3.4. The neuron is silent when the external input signal[8] first begins, and there is no inhibitory neurotransmitter at the feedback synapse. If the external input signal is sufficiently strong, the neuron fires rapidly. But these initial output spikes cause a buildup in the feedback neurotransmitter, which is assumed to have a relatively long time constant. Depending upon the strength of the input signal and the neuron's parameters, the initial burst of spikes may either cause the neuron to temporarily stop producing spikes, or gradually reduce its spike rate. Eventually the output spike rate converges to a value determined by:

$$\Theta_i = V_i(\Delta) = \tanh^2\left(\frac{\Delta}{\tau_i}\right)\cdot\left[R_{i0} + R_{ix}\cdot\tanh\left(T_{ix}\cdot I^{\text{in}}\right) + R_{ii}\tanh\left(\frac{T_{ii}}{R_{ii}}\cdot U_{ii}(\Delta)\right)\right] \tag{4.47}$$

where $U_{ii}(\Delta)$ is calculated using Equation (4.45) with $n = 1$ and $d_{ii} = 0$. Notice that this is the same as Equation (4.44), but with a contribution from an external input

[8]Notice that an *external* input signal is used, and not input spikes from another neuron. While a feedback connection can also be used to adapt the output signal with input spikes, the effects from the adaptation are not as obvious due to the latency effect discussed in Section 4.3.2. External input signals are assumed to have no latency. (See footnote on page 54.)

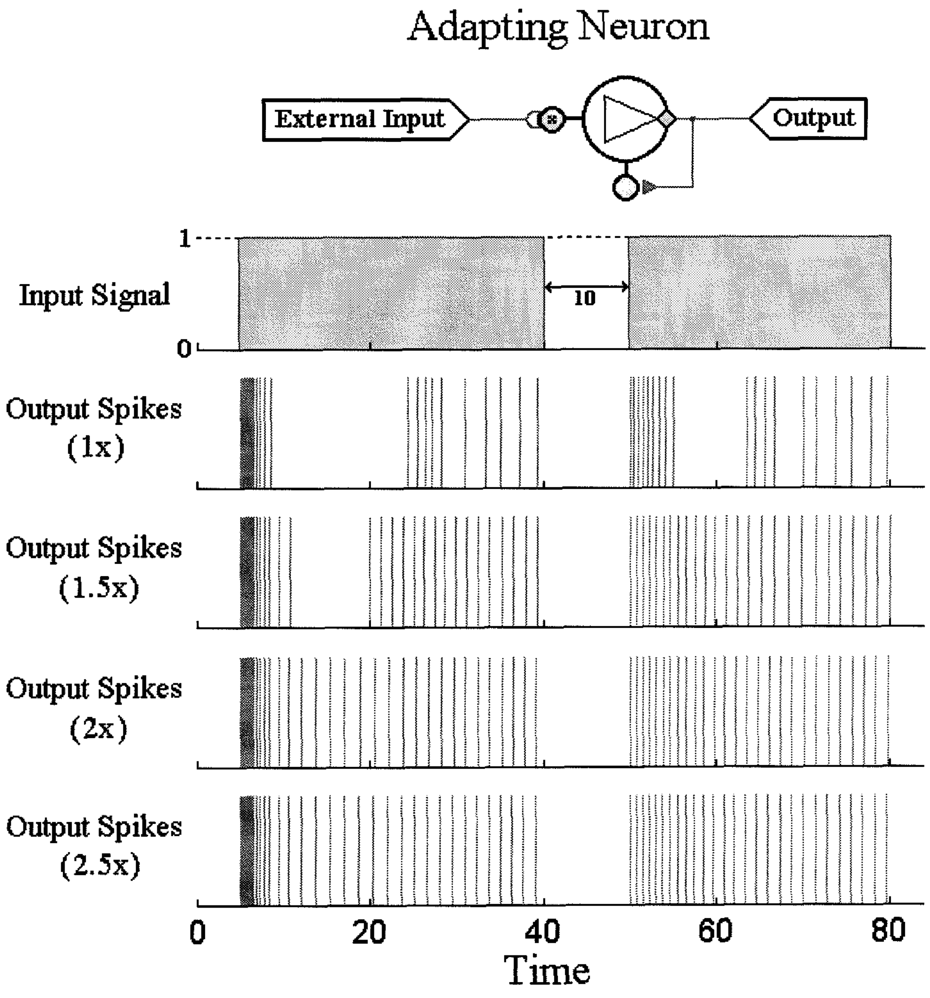


Figure 4.13: **Adapting Neurons.** This figure shows the output spikes for a neuron with an inhibitory feedback signal, which allows it to adapt to changes in input intensity. The parameters are the same for each of the four output spike trains ($\Theta_i = 0.5$, $R_{i0} = 0.0$, $\tau_i = 0.25$, $R_{ii} = 3.5$, $T_{ii} = -3.5$, $\tau_{ii} = 8.0$, $R_{ix} = 4.0$, and $T_{ix} = 1.0$), but the strength of the input signal increases from top to bottom ($I^{\text{in}}$ =1, 1.5, 2, and 2.5). Notice that the initial spike bursts and final output frequencies are very similar for all of the inputs. The input stimulus is discontinued for $40 \leq t \leq 50$, and when the stimulus returns at $t = 50$, the frequency of the initial output burst is much less than previously produced.

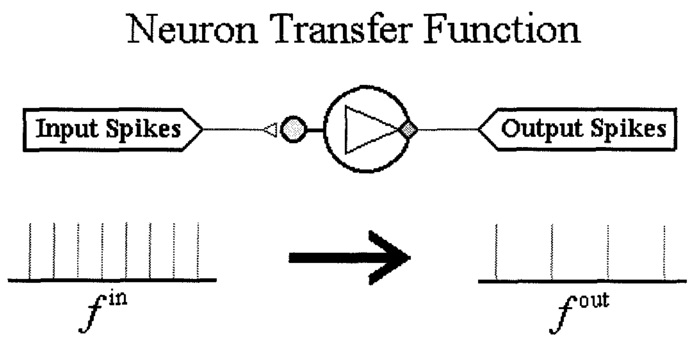


Figure 4.14: **Schematic for the Neuron Transfer Function.** A neuron with a single excitatory input connection, which provides spikes at regular frequency, is used to analyze the frequency transfer function of the SNM.

signal.

When the input signal stops, the output spikes immediately cease; however, the residual neurotransmitter in the feedback synapse slowly decays with time. Thus, if the input stimulus begins again, the initial burst of spikes is attenuated, with the output frequency depending upon the amount of elapsed time since the previous stimulus ended.

## 4.4 Frequency Transfer Function

This section considers the frequency gain of a neuron with a single excitatory input connection, as shown in Figure 4.14. If the input signal is beating ($P^1$), then what is the output frequency of the neuron as a function of the input frequency? If $R_{i0} > \Theta_i$, the neuron already has an intrinsic frequency, which represents its minimum frequency with no input spikes; i.e.:

$$f^{\min} \equiv \frac{1}{\Delta^{\min}} = \begin{cases} 0 & \text{if } \Theta_i \geq R_{i0} \\ \dfrac{1}{\tau_i \text{Atanh}\left(\sqrt{\frac{\Theta_i}{R_{i0}}}\right)} & \text{if } \Theta_i < R_{i0} \end{cases} \tag{4.48}$$

While $U_{ij}(t)$ continues to increase as the input spike frequency rises, the response of the neuron is bounded by the tanh $(-)$ function associated with each input synapse.

Thus, the asymptotic limit on the maximum output frequency of the neuron is given by:

$$f^{\max} \equiv \frac{1}{\Delta^{\max}} = \frac{1}{\tau_i \cdot \operatorname{Atanh}\left(\sqrt{\frac{\Theta_i}{R_{i0}+R_{ij}}}\right)} \tag{4.49}$$

For the neuron to reach threshold and produce an output spike, the minimum amount of neurotransmitter required is:

$$U^{\text{spike}} = \begin{cases} \left(\frac{R_{ij}}{T_{ij}}\right) \cdot \operatorname{Atanh}\left(\frac{\Theta_i - R_{i0}}{R_{ij}}\right) & \text{if } \Theta_i \geq R_{i0} \\ 0 & \text{if } \Theta_i < R_{i0} \end{cases} \tag{4.50}$$

The output frequency is zero when the input spike frequency is insufficient for causing the neurotransmitter level to rise above $U^{\text{spike}}$. But, the case may be that the parameters are such that the maximum neurotransmitter contribution from a single input spike, $\mathrm{e}^{-1}$ is above $U^{\text{spike}}$ and the neuron can produce at least one output spike for each input spike at low frequencies. Of course if there is a very small refractory time constant relative to the neurotransmitter time constant, the neuron may be able to produce two or more output spikes for each input spike.

Due to the variability in behavior with the different neuron parameters, there is no general equation that provides the output frequency as a function of input frequency. For many input frequencies, the output converges to a spike pattern with several spikes per period; e.g., see Figure 4.1. However, it is possible to place bounds on the *average* output spike frequency. (When there are several spikes per period, the average frequency is the number of spikes divided by the length of the period.) Using the maximum and minimum neurotransmitter from the periodic input signal given in Equations (4.25)-(4.26), the output frequency is bounded by:

$$f^{\text{upper}} = \frac{1}{\tau_i \cdot \operatorname{Atanh}\left(\sqrt{\frac{\Theta_i}{R_{i0}+R_{ij}\tanh(u^{\text{upper}})}}\right)} \tag{4.51}$$

$$f^{\text{lower}} = \frac{1}{\tau_i \cdot \operatorname{Atanh}\left(\sqrt{\frac{\Theta_i}{R_{i0}+R_{ij}\tanh\left(u^{\text{lower}}\right)}}\right)} \tag{4.52}$$

$$\text{where:} \quad u^{\text{upper}} \equiv \frac{T_{ij} \cdot \mathrm{e}^{-1}}{R_{ij} \cdot \kappa\left(\frac{1}{\tau_{ij} f^{\text{in}}}\right)} \tag{4.53}$$

$$u^{\text{lower}} \equiv \frac{T_{ij} \cdot F\left(\psi\left(\frac{1}{\tau_{ij} f^{\text{in}}}\right)\right)}{R_{ij} \cdot \kappa\left(\frac{1}{\tau_{ij} f^{\text{in}}}\right)} \tag{4.54}$$

These equations represent what the output spike frequency would be if the neurotransmitter did not vary with each input spike, but instead remained at a constant level of either $\max\{U_{ij}(t)\}$ or $\min\{U_{ij}(t)\}$. Since the actual amount of neurotransmitter varies between these levels, they represent upper and lower bounds on the output spike frequency.

When $\max\{U_{ij}(t)\}$ is above $U^{\text{spike}}$, but $\min\{U_{ij}(t)\}$ is below $U^{\text{spike}}$, the times of the output spikes are strongly coupled with the arrival times of the input spikes. But as $\min\{U_{ij}(t)\}$ rises above $U^{\text{spike}}$, the times of the output spikes are only weakly coupled with the times of the input spikes. This phenomenon can be understood from Figure 4.2 on page 51, which shows the neurotransmitter of an input signal for three different frequencies. If $U^{\text{spike}}$ for the neuron is 0.70, then the neurotransmitter level is always insufficient for producing output spikes when $\frac{\Delta}{\tau_{ij}} = 3$. For $\frac{\Delta}{\tau_{ij}} = 1.5$, the neurotransmitter level is briefly above the required level for a limited time window between input spikes, and the neuron can produce output spikes during those times. But when $\frac{\Delta}{\tau_{ij}} = 1$, the neurotransmitter concentration is always above the required level, and output spikes can be produce at any time.[9]

Using the average amount of neurotransmitter at the synapse given in Equation (4.27), a reasonable estimate for the output frequency can be obtained; i.e.:

$$f^{\text{out}} \approx f^{\text{ave}} \equiv \frac{1}{\tau_i \cdot \operatorname{Atanh}\left(\sqrt{\frac{\Theta_i}{R_{i0} + R_{ij} \tanh\left[\left(\frac{T_{ij}}{R_{ij}}\right) \tau_{ij} f^{\text{in}}\right]}}\right)} \tag{4.55}$$

[9]If $\min\{U_{ij}(t)\}$ is above $U^{\text{spike}}$, then the phase differences between the input and output spikes depend upon the neuron's initial condition at the arrival times of the input spikes. However, the variations in $U_{ij}(t)$ between $\min\{U_{ij}(t)\}$ and $\max\{U_{ij}(t)\}$ with each input spike act like perturbations on the output's oscillatory cycle, and for many input frequencies, the output spikes eventually become phase locked with the input spikes.

Notice that $f^{\text{ave}}$ is not the mean of $f^{\text{upper}}$ and $f^{\text{lower}}$.

Now, the gain of the neuron can be defined as the derivative of the output frequency with respect to the input frequency when the output frequency is at the midpoint of the asymptotic limits given in Equations (4.48)-(4.49). Using Equation (4.55) to represent the relationship between the input and output frequencies, the gain, $\Gamma$ is given by:

$$\Gamma \equiv \left.\frac{df^{\text{out}}}{df^{\text{in}}}\right|_{f^{\text{out}}=\frac{f^{\min}+f^{\max}}{2}} = \left(\frac{\tau_{ij}}{\tau_i}\right)\cdot\left(\frac{T_{ij}}{2\Theta_i}\right)\cdot\left[\frac{\left(\frac{z^{3/2}}{1-z}\right)\cdot\left[1-\left(\frac{\frac{\Theta_i}{z}-R_{i0}}{R_{ij}}\right)^2\right]}{\text{Atanh}^2\left(\sqrt{z}\right)}\right] \tag{4.56}$$

$$\text{where:}\quad z \equiv \begin{cases} \frac{4K}{(1+K)^2} & \text{if } \Theta_i \geq R_{i0} \\ \tanh^2\left[\frac{2\text{Atanh}(\sqrt{K})\cdot\text{Atanh}(\sqrt{k})}{\text{Atanh}(\sqrt{K})+\text{Atanh}(\sqrt{k})}\right] & \text{if } \Theta_i < R_{i0} \end{cases} \tag{4.57}$$

$$K \equiv \frac{\Theta_i}{R_{i0}+R_{ij}} \tag{4.58}$$

$$k \equiv \frac{\Theta_i}{R_{i0}} \tag{4.59}$$

When the output frequency is at its midpoint, $\left(\frac{f^{\min}+f^{\max}}{2}\right)$, the input frequency is:

$$f^{\text{in}} = \left(\frac{R_{ij}}{\tau_{ij}\cdot T_{ij}}\right)\cdot\text{Atanh}\left(\frac{\frac{\Theta_i}{z}-R_{i0}}{R_{ij}}\right) \tag{4.60}$$

Notice in Equation (4.56) that the gain directly depends upon $T_{ij}$, while its dependence upon the other parameters is considerably more complicated. A plot of the frequency transfer function is shown in Figure 4.15 for several different sets of neuron parameters.

## 4.5 Summary of Properties

Overall, the SNM is able to duplicate many of the well known properties of real neurons. One of the key properties that gives the SNM its unique characteristics

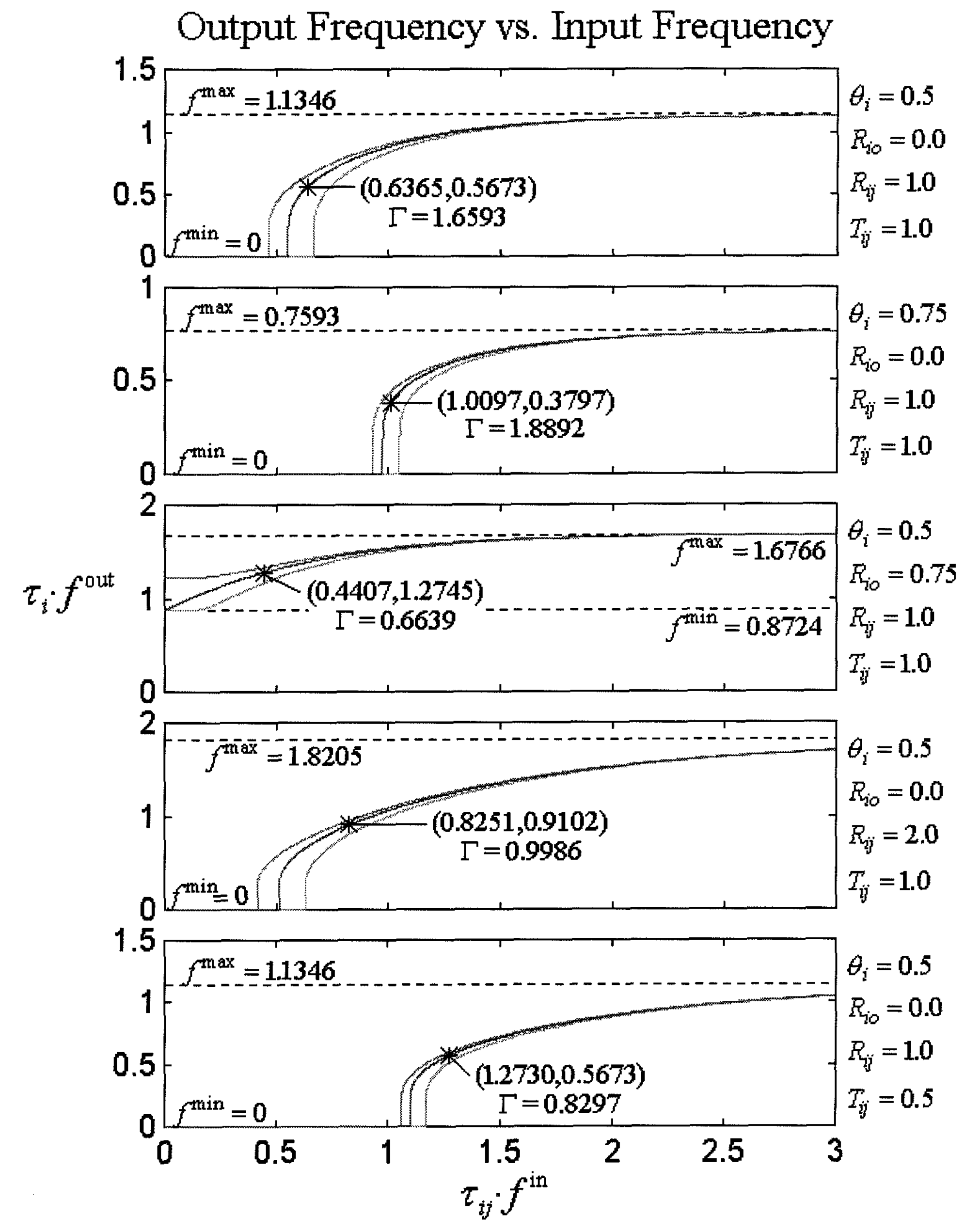


Figure 4.15: **Frequency Gain.** The plot shows the output frequency as a function of the input frequency for 5 different sets of neuron parameters. In each graph, the top curve is $f^{\text{upper}}$, the bottom curve is $f^{\text{lower}}$, and the middle curve is $f^{\text{ave}}$, the estimate for the actual output frequency given in Equation (4.55). The frequencies are scaled by the time constants of $\tau_{ij}$ and $\tau_i$. (Note the different ranges on the y-axes.) An "$*$" denotes where $f^{\text{ave}}$ is at the midpoint of its asymptotic limits. The gain, $\Gamma$, is defined as the slope of the transfer function at this point. For all three frequency curves, $f^{\text{upper}}$, $f^{\text{ave}}$, and $f^{\text{lower}}$, the slope is infinite at the input frequencies which initiate their respective output spikes, provided of course that $\Theta_i > R_{i0}$.

is that the state of a neuron does not just depend upon the current input signals, but depends upon all previous input spikes. Each input spike triggers the release of neurotransmitter at a synapse. Successive input spikes lead to a build up of neurotransmitter at the synapses, causing the neuron's potential to change by more than it could from individual spikes. Although the total neurotransmitter at a synapse depends upon the entire spike history of the presynaptic neurons, the neurotransmitter may be expressed in an iterative formula (Section 4.2.2), so that it is not necessary to recalculate the sum over all previous spikes every time a new spike arrives. And when the input spikes occur with a repeating pattern, the iterative representation for the neurotransmitter can be further simplified (Section 4.2.3).

While neurotransmitter is used by neurons to exchange information, it is the neuron's potential that determines the production of spikes. Like real neurons, the SNM can exhibit latency (Section 4.3.2), refractory periods (Section 4.3.3), endogenous spiking behavior (Section 4.3.4), and adaptation (Section 4.3.5). With latency, strong input stimuli result in the rapid production of output spikes, while weak stimuli require more time and may not even lead to output spikes. After the SNM fires, there is an absolute refractory period during which it is unable to fire. As the SNM starts to recover, there is a relative refractory period during which it is only able to produce output spikes when the input stimulus is stronger than normal. When the neuron parameters are set accordingly, the SNM can produce a beating or bursting output spike train in the absence of external stimulation. With an inhibitory feedback connection, the SNM can adapt its output spike frequency to constant input signals. This allows it to be more sensitive to changes within the input signal rather than the input's strength.

Like other computational devices, the SNM can be analyzed in terms of its transfer function. Since the neuron receives input spikes and produces output spikes, the transfer function can be expressed as the relationship between the neuron's input and output frequencies (Section 4.4). The SNM has both a maximum and minimum output frequency. Unless the neuron has an intrinsic spiking behavior, the minimum output frequency is zero. Excitatory input spikes cause the output spike frequency to increase, but the absolute refractory period limits the maximum possible output spike frequency.

The net effect of having a refractory period along with the strength-latency relationship is that the neuron model can encode information within its output spike frequency [65, p. 39]. As with real neurons, a stronger input stimulus produces a faster spike response and higher output frequency. Notice that either the frequency or the inverse of the time between spikes can be thought of as encoding the strength of the signal. Both encoding schemes produce similar output spike trains; however, a frequency encoding scheme usually implies that the spike rate is averaged over some time period, while an encoding scheme based on the time between spikes uses each pair of spikes to encode information. Thus, a time between spikes encoding scheme can lead to a faster throughput of information. Chapter 8 discusses how spikes can be used to encode information within neural networks.

# Chapter 5 Simulating the SNM

## 5.1 Introduction

The Spiking Neuron Model of Chapter 3 was designed to provide a simple and efficient method for capturing the complex spiking dynamics associated with real neurons. Since there are other more detailed neuron models, which are designed to accurately simulate nearly all of the known information on biological neurons, the SNM is of little value unless it can be easily simulated. This chapter addresses this issue by presenting two possible methods for simulating the SNM. Obviously, there are numerous other approaches that can be used, all of which have various tradeoffs between speed, accuracy, and memory requirements, but the two methods outlined in this chapter are sufficient for demonstrating the usefulness of the SNM.

The first method is the Single Time Step Iterative Method. It is an easy method to implement and understand; however, it has several limitations: it is slow, and it can not be used with adjustable parameters or presynaptic connections. The second method is called Simulating in Time Segments. It overcomes the previous method's limitations, but at the price of requiring more memory. Both methods utilize a fixed time step, with the network's neurons only allowed to produce or receive an output spike at one of these discrete times. The problems caused by using a fixed time step are discussed in Section 5.4.

## 5.2 Single Time Step Iterative Method

The first SNM simulation method uses fixed time steps and calculates the value of the input neurotransmitter, $U_{ij}(t)$, and potential voltage, $V_i(t)$, at each step. When a neuron's potential voltage is at or above threshold, $\Theta_i$, it produces an output spike, which in turn causes neurotransmitter to be input into all of its output synapses. Of course, the time at which the neurotransmitter is released depends upon the delay, $d_{ij}$, associated with each of the interneuron connections.

To implement this method, it is necessary to store the parameters associated with each neuron, connection, and synapse. From the network's initial condition, the amount of neurotransmitter at each connection can be calculated using Equations (4.9) and (4.14). (The initial condition is specified by giving the initial values of the refractory coefficients for each neuron and $U_{ij}$ and $\mu_{ij}$ at each interneuron connection.) Notice that Equation (4.9) is used at each time step, while Equation (4.14) is only used when an input spike arrives; i.e., if $h$ is the size of the time step, and neuron $n_m$ produces an output spike at $t_m^K$, then Equation (4.14) is only used when $(N-1)\cdot h < t_m^K + d_{ij} \leq N \cdot h$, where $N$ is an integer and the current simulation time is given by $t = N \cdot h$. From the neurotransmitter at each synapse, the potential voltages of the neurons can be calculated using Equation (3.1) and compared to the threshold levels to determine the production of output spikes. Since Equation (4.14) is an iterative formula for the *exact* amount of neurotransmitter at a synapse, the values calculated will be precise except for the rounding errors that can occur in the process of determining when a spike is produced or arrives at a synapse.

While this method is easy to implement, it has three main drawbacks: (1) it is slow; (2) it can not be used with presynaptic connections; and (3) it can not be used if $T_{ij}$ is a variable parameter (i.e., if any of the $T_{ij}$ parameters change due to learning). The second and third problems are related, since a presynaptic connection changes the effective value of $T_{ij}$, and Equations (4.9) and (4.14) are only valid when $T_{ij}$ is fixed (see footnote on page 27). The next section discusses an alternative to this method, which simulates time in segments rather than one step at a time.

## 5.3 Simulating in Time Segments

The previous method's performance may be considerably improved by calculating the neuron variables in time segments. While this method can be used with variable parameters, it works best if all of the network's time constants, $\tau_{ij}$ and $\tau_i$, are fixed.

(This will be the case for all of the networks considered in this thesis.) The primary advantage of this method when using fixed time constants is that the computationally expensive neurotransmitter and refractory functions can be calculated once and stored in memory. When a spike occurs, these functions can then be added or multiplied as necessary to update the variables without needing to recompute their values. This method is particularly well suited for simulations in MALTAB, which allows the use of "vectorized" variables; however, it also requires considerably more memory.

Associated with each neuron are vectors for the refractory coefficient, external input contribution to the potential, and the neurotransmitter at each of the input connections. (There may also be presynaptic connections.) If the simulation time is short, then the length of the vectors is determined by the total number of time steps in the simulation. (The case when the simulation time is not short will be discussed next.) For each of the different synaptic time constants, $\tau_{ij}$, the neurotransmitter function, $\left(\frac{t}{\tau_{ij}}\right) \mathrm{e}^{-\left(\frac{t}{\tau_{ij}}\right)}$, is calculated at each time step, over the entire length of the simulation and stored in a vector. Also, the refractory coefficient for each neuron, $\tanh^2\left(\frac{t}{\tau_i}\right)$, is calculated and stored in a vector. Notice that both of these calculations are done before any input/output spikes are actually produced.

Now, the program searches all of the neurons' potential voltages and finds the first one to reach threshold. Its refractory coefficient is updated for the remaining simulation time by *replacing* it with a time shifted version of the vector already calculated. Similarly, the neurotransmitter at its output connections is updated by *adding* a time shifted version of the neurotransmitter function to the vector representing the neurotransmitter already in the synapse.[1] The program then finds the next neuron to reach threshold and repeats the process.

[1]In actuality, to allow for variable parameters, it is necessary to store the times of the input spikes associated with each connection and add the neurotransmitter function only when an input spike is the next event to occur. (Events are either input or output spikes.) This is necessary because when there is a large time delay, the parameters can change between the time of the output spike and the time that the corresponding input spikes are actually received. Thus, the program must first decide whether the next event is an input or output spike. If it is an input spike, the neurotransmitter can be added at the appropriate connections, but if it is an output spike, the neuron's refractory term is reset and the time of this new spike, plus the appropriate delays, is added to the list of incoming spikes. Of course, if there is no delay associated with a connection, it may be added immediately.

One important thing to remember when using this method is that the initial conditions for the neurotransmitter must be specified in terms of the entire simulation time; i.e., it is not sufficient to only specify the initial values for $U_{ij}(t_o)$ and $\mu_{ij}(t_o)$ at each interneuron connection, but instead, the total amount of neurotransmitter at each time step of the simulation, due to all previous input spike activity, must be given. Of course, this may be calculated from $U_{ij}(t_o)$ and $\mu_{ij}(t_o)$ using Equation (4.9), before the actual simulation begins.

If the total length of the simulation is unknown or is too long to have the vectors represent the entire simulation, then the simulation may be divided into time segments, provided that the neurotransmitter and refractory coefficients can be approximated with truncated functions. E.g., suppose that the total simulation time is 100 seconds, and the maximum synaptic time constant is $\tau_{ij} = 0.5$ and the maximum refractory time constant is $\tau_i = 1.0$. Therefore, 6 seconds after a spike, the maximum neurotransmitter contribution can only be 0.00007373, and the minimum refractory coefficient is 0.999975. Thus, it is reasonable to only keep track of the neurotransmitter and refractory terms for the first 6 seconds and approximate them by 0 and 1 for later times.

The length of each simulated time segment may be as long as allowed by the available memory, but the total length of each of the neurons' vectors must be as long as the time segment plus the length of the truncated functions; e.g., in the above example, if time is simulated in segments of 4 seconds, and the truncated functions are 6 seconds, then the length of each of the neurons' vectors must be long enough to hold data for 10 seconds. This is necessary because any input or output spikes that occur at the end of the 4 second time segment currently being simulated can affect neurons up to 6 seconds after they occur. After each time segment is simulated, the neurons' vectors can be divided into two sections, with the first section corresponding to the time just simulated and the next section corresponding to the length of the truncated functions. The first section is then discarded, the second section is shifted to the front, and the remain times are set to their asymptotic initial values. These ideas are illustrated in Figures 5.1-5.3, which represent the actual simulation vectors for a neuron. The simulation used time steps of 0.001 seconds, so that each vector contained 10,000 values.

When comparing this simulation method with the Single Time Step Iterative Method, it is important to realize that they are fundamentally different approaches, even when the time segment is reduced to a single time step. The Time Segments Method uses a precalculated truncated approximation to the neurotransmitter contribution from each input spike, and adds this to the current amount of neurotransmitter present at the junction; conversely, the Single Time Step Iterative Method uses the iterative formulas given in Chapter 4 to calculate the neurotransmitter at each time step. Both methods produce nearly identical results, with only slight variations occurring due to the truncation errors associated the Time Segments Method. In general, it is considerably faster than the Single Time Step Iterative Method, since the non-linear neurotransmitter and refractory functions only need to be calculated once initially and then added at the appropriate times. Also, it is more versatile since it allows for variable parameters and presynaptic connections. Most of the results presented in this thesis were produced using the Time Segments simulation method.

## 5.4 Problems with using Fixed Time Steps

One of the primary advantages of the SNM is that it is not formulated as a differential equation; consequently, the simulation time steps can be arbitrarily large without effecting the accuracy of the calculation. However, the input and output spikes are neither received or produced immediately when a neuron's potential voltage rises above threshold, but instead occur at the discrete simulation time steps.

The size of the time step, $h$, sets the minimum time the potential voltage needs to be above threshold to guarantee that an output spike will be produced. When the neuron's potential voltage is above threshold for less than one time step, the

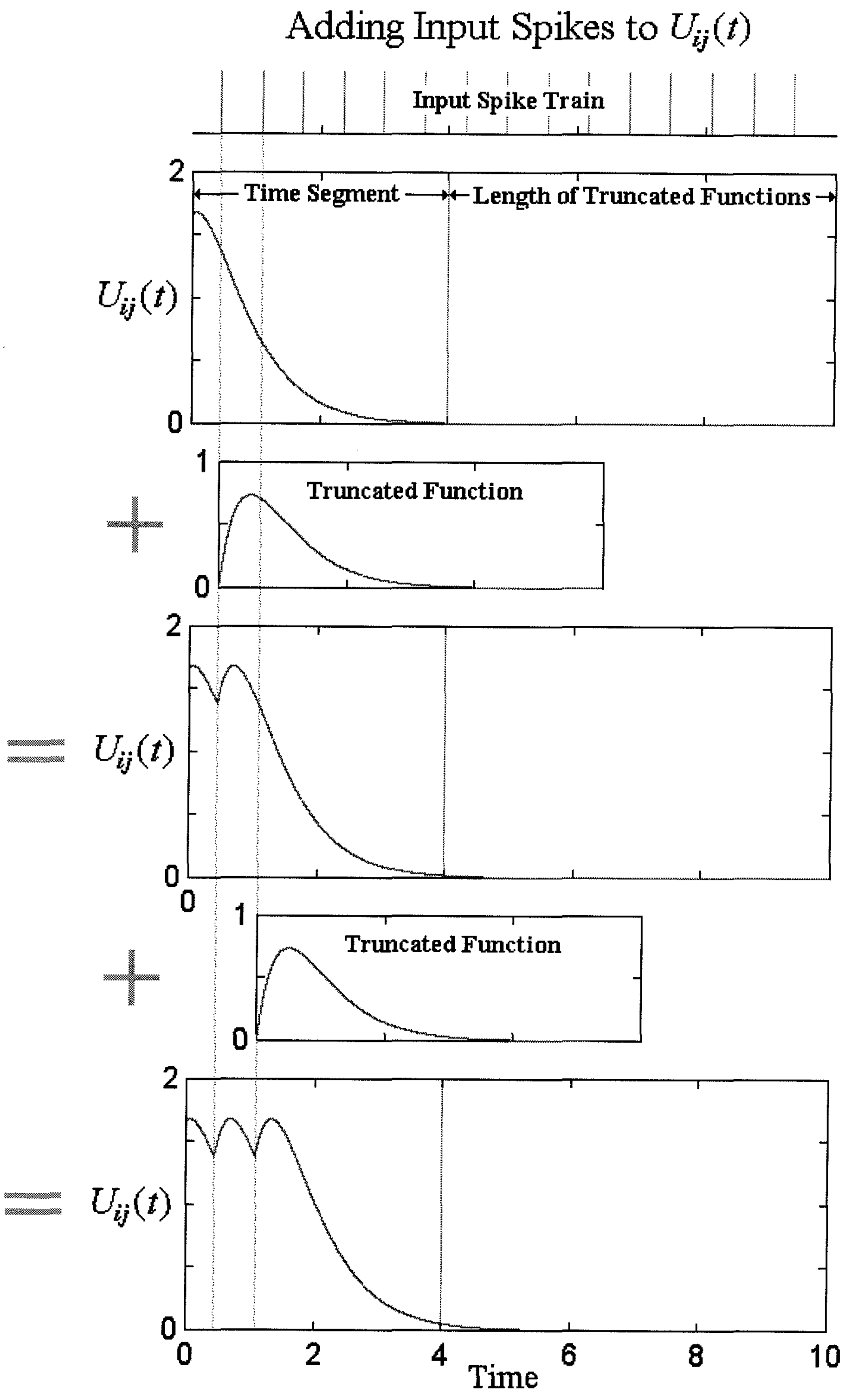
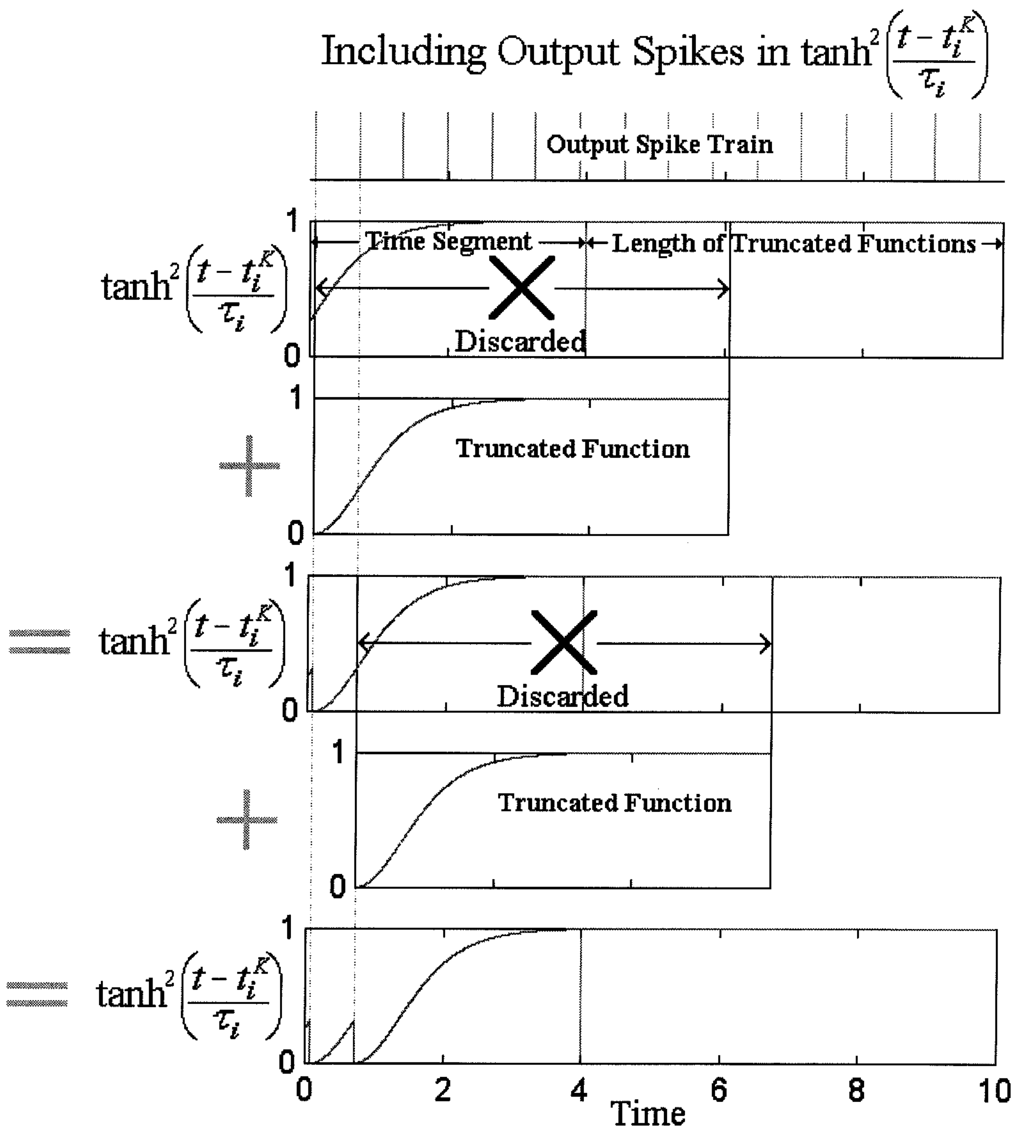


Figure 5.1: **Simulating Neurotransmitter in Time Segments.** This figure shows how time shifted versions of the truncated neurotransmitter function can be added at the appropriate times to simulate the neurotransmitter as a function of time.



Including Output Spikes in $\tanh^2\left(\frac{t-t_i^K}{\tau_i}\right)$
Output Spike Train
Time Segment
Length of Truncated Functions
Discarded
Truncated Function
Discarded
Truncated Function
$\tanh^2\left(\frac{t-t_i^K}{\tau_i}\right)$
+
=
Time
0
2
4
6
8
10

Figure 5.2: **Simulating Refractory Coefficient in Time Segments.** This figure shows how time shifted versions of the truncated refractory coefficient function can replace sections of the refractory coefficient vector stored in memory to simulate its time evolution.

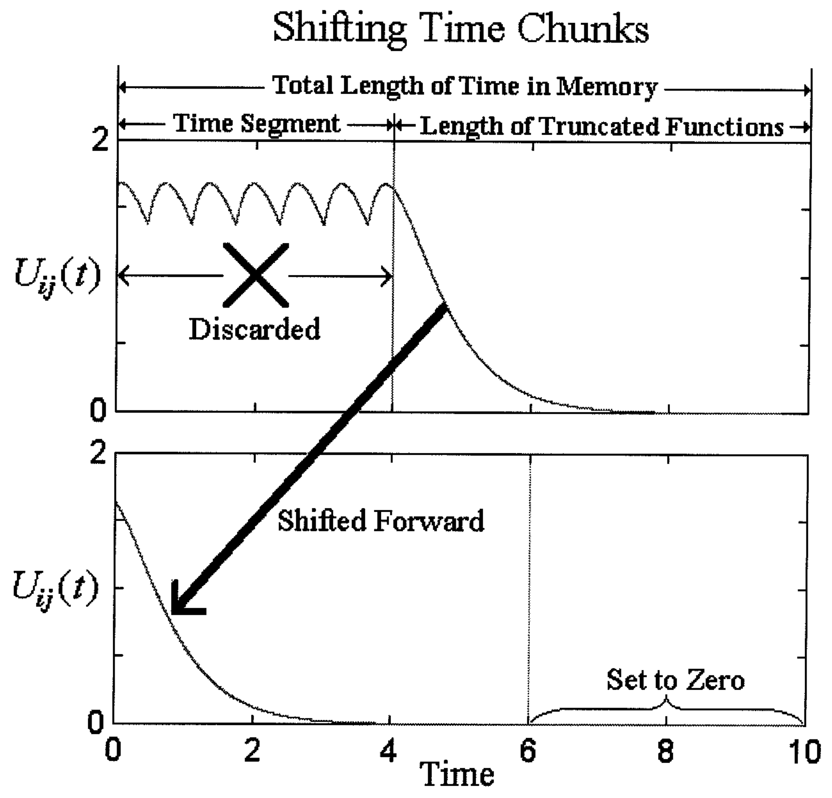


Figure 5.3: **Shifting Time Segments.** This figure shows how the vectors representing the neurotransmitter must be time shifted when the simulation has accounted for all of the events within the current time segment. After all of the vectors are shifted, the simulation resumes at the start of the next time segment. While this figure only shows the shift for a neurotransmitter vector, the time shifts in the neurons' refractory coefficient vectors are identical, except that the undetermined spots at the end of the vectors are set to one instead of zero.

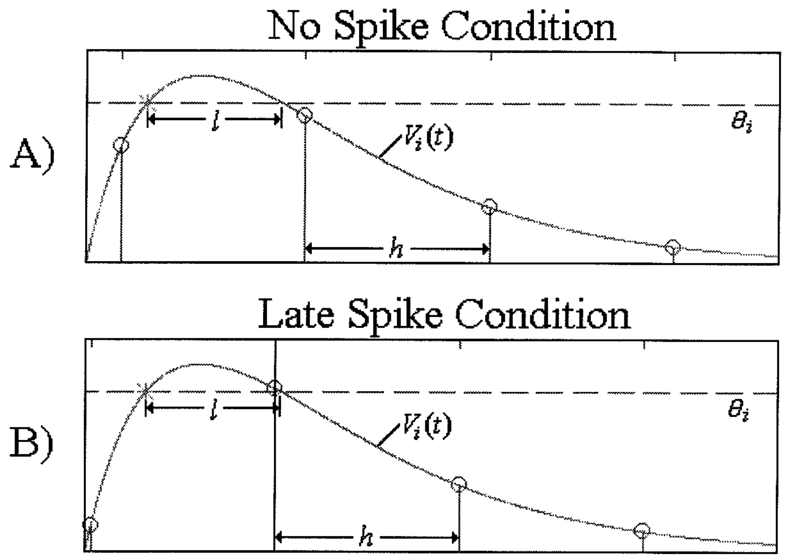


Figure 5.4: **Problems with Simulating the SNM using Fixed Time Steps.** Both (A) and (B) use the same curve for the neuron's potential voltage, $V_i(t)$, and the same time step, $h$, but the curves are sampled at different times. An "$*$" marks the time when $V_i(t)$ first reaches threshold, and a spike should be produced. In (A), $V_i(t)$ is below threshold for all sampled points, so no spike is produced. This can occur when $l < h$. In (B), a spike is produced, but it lags behind the time when $V_i(t)$ first reaches threshold.

probability of an output spike being produced is:

$$P(\text{spike}) = \begin{cases} \frac{l}{h} & l < h \\ 1 & l \geq h \end{cases} \tag{5.1}$$

where $l$ represents the amount of time the neuron's potential, $V_i(t)$, is at or above threshold, $\Theta_i$. And when an output spike is produced, it lags behind the actual time the potential reaches threshold, with the maximum lag given by:

$$\max\{\text{Time Lag}\} = \begin{cases} l & l < h \\ h & l \geq h \end{cases} \tag{5.2}$$

These ideas are depicted in Figure 5.4.

To eliminate the late spike condition, it is possible to search for the onset times of the threshold crossings. Such a search can be triggered by finding the neuron's potential below threshold at one time step and above threshold at the next. When the exact threshold time is found, it can then be used to accurately represent the time of the output spike. Notice that while this search method can be used in conjunction with the Single Time Step Iterative Method, it can not be used with the Time Segments Method. This is because the truncated functions in the Time Segments Method are precalculated at the same fixed time steps and can not be shifted by fractions of a time step and added to the overall function.

It is even a much more difficult task to eliminate the no spike condition. To do so it would be necessary to calculate all of the local maximums in the neurons' potentials and check to see if their values are above threshold. While it is theoretically possible to do these calculations, it is a very computationally expensive task for neurons which have several input connections with different time constants. Consequently, it is usually best to assume that the simulation introduces a probability component into the neuron dynamics, which requires that the neuron's potential be above threshold for at least one time step to *guarantee* the production of an output spike. With reasonably sized time steps, few spikes should be missed. And since biological neurons are somewhat probabilistic and do not always produce spikes when the potential voltage briefly exceeds threshold, this should not severely limit the computing abilities of the simulated SNM.

Notice that using a large time step can increase the error in calculating the time of an output spike; however, this is different than the integrating error associated with using large time steps when the neuron dynamics are formulated as differential equations. In general, when numerically integrating a set of ordinary differential equations, the error at *each* time step depends upon the step size raised to a power, (e.g., for Euler's method, the error is $O(h^2)$). Thus, the total error can grow as the equations are integrated forward, regardless of whether or not a spike is produced. Furthermore, when there are different time scales associated with the equations, the equations may be *stiff* and the numerical integration method used may become unstable. (See [84, Chapter 15] for a discussion of the some of the problems and numerical techniques used to integrate ordinary differential equations.)

## 5.5 Summary of Simulation Methods

While there are numerous methods to simulate the dynamic equations of the SNM, this chapter presented two alternatives. The Single Time Step Iterative Method (Section 5.2), uses the iterative formulas developed in Section 4.2.2 to calculate the exact amounts of neurotransmitter at each synapse. It is relatively easy to implement, but it can not be used with learning or presynaptic connections.

The Simulating in Time Segments method (Section 5.3) approximates the neurotransmitter with truncated functions. When an input spike arrives, the truncated neurotransmitter function is added to a vector which keeps track of the total amount of neurotransmitter at a synapse. While this method requires more memory, it tends to be faster than the Single Time Step Iterative Method, but the total simulation time depends upon the number of input and output spikes associated with each neuron. Also, it is able to be used with learning and presynaptic connections.

With both of these simulation methods, the times of the input and output spikes are subject to rounding errors due to the fixed time steps. If a neuron's potential voltage is not above threshold for at least one simulation time step, the production of the output spike may be missed. But because the SNM is not formulated in terms of differential equations, neither simulation method can suffer from the numerical instabilities associated with integrative techniques. Since it is not necessary to worry about the stability of each neuron, the SNM is ideal for large network simulations.

# Chapter 6 Special Function Neurons

## 6.1 Introduction

This chapter presents some neuron configurations for accomplishing specific functions. These neurons can be used as the building block of larger networks, some of which are presented in Chapter 7. While these neurons usually perform their intended tasks well, they are not meant to be precise computational devices. Like real neurons, they are designed to process information that is noisy or imprecise and they can provide reasonable outputs.

Section 6.2 discusses a "high gain" neuron, which outputs spikes at a nearly constant frequency when the input signal exceeds the neuron's threshold level. The neurons in Section 6.3 have two stable states: silent or oscillating. Input spikes cause them to switch between the states. Since the current state depends upon the input spike history, these neurons are referred to as memory oscillators. They are analogous to digital flip-flops. Section 6.4 analyzes neurons with bounded threshold levels. Normally, a neuron has a minimum threshold level for producing output spikes, but a neuron with a bounded threshold level also has maximum threshold level. When the input signal exceeds the maximum level, the neuron is unable to produce output spikes. And finally, Section 6.5 presents "identity" neurons. Unsynchronized identity neurons produce an output spike for every input spike, and can be used to sum input spike trains onto a single output line. Synchronized identity neurons produce output spikes at regular intervals when spikes are input. Thus, the neuron has an underlying frequency of events (spikes or non-spikes).

Each section in this chapter begins by discussing the desired function the neuron is to accomplish, and then develops constraints on the neuron's parameters for ensuring that the function is performed. In general, the constraint equations presented are sufficient, but not necessary for implementing the desired function, and looser constraints may be imposed, but usually at a cost of complicating the analysis. Each section also contains a sample output from a neuron performing the desired function, and gives the parameter values used. Although it is sometimes difficult to find a set of parameters that satisfy the constraints, since they are continuous equations, the solution provided resides within a connected region of the parameter space, and the parameter values may be modified as desired to alter the neuron's output while remaining within the constraint boundaries.

## 6.2 High Gain Neurons

Previously, Section 4.4, which described the frequency transfer function of the SNM, showed that most neurons require a minimum input spike frequency before any output spikes can be produced. As the input frequency increases, the output frequency reaches a saturation level determined by the neuron's absolute refractory period. A high gain neuron is designed to minimize the transition range. Thus, the output spikes are produced at a nearly constant frequency when the strength of the input stimulus exceeds a minimal strength, and when the input stimulus stops, the neuron's spiking behavior ceases. (The neuron is assumed to have no intrinsic oscillatory behavior; i.e., $\Theta_i > R_{i0}$.) The high gain neuron requires only one input synapse, which is excitatory, although more may be present. It is particularly useful for detecting the presence of an input stimulus.

The neuron's parameters are set so that when the input spike frequency is above a chosen threshold level, the released neurotransmitter causes the synapse to saturate; i.e., $\tanh\left(\frac{T_{ij}}{R_{ij}} \cdot U_{ij}(t)\right) \approx 1$. The change in the neuron's potential due to the saturated synapse is just enough to keep it above threshold, and the neuron's refractory time constant, $\tau_i$, sets the output frequency. Because the synapse must be saturated in order to produce an output spike, the output frequency does not significantly increase when the input stimulus is stronger than the necessary minimum.

Three design parameters determine the behavior for this type of neuron. Figure 6.1 shows the frequency transfer function of a high gain neuron, with the corresponding

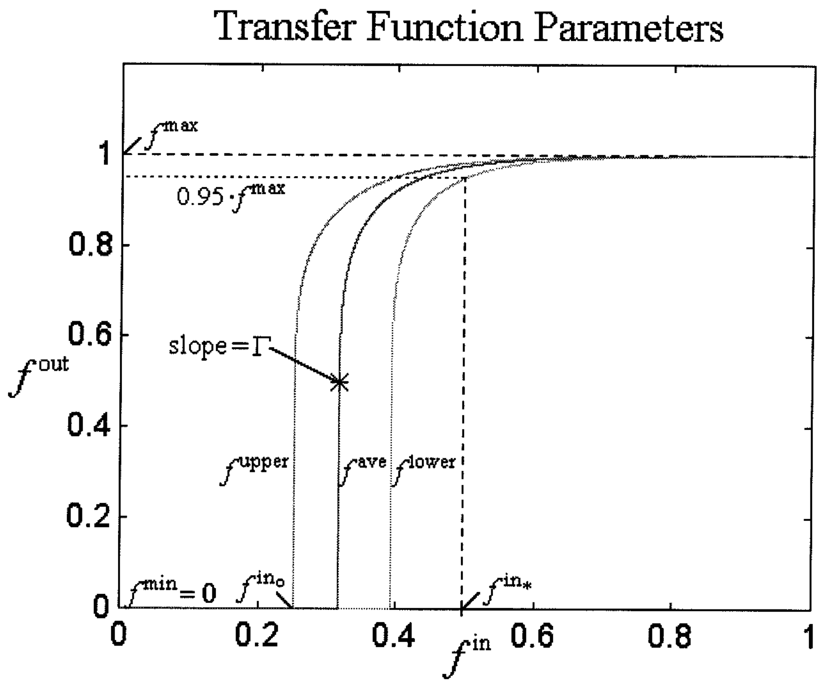


Figure 6.1: **Design Parameters for a High Gain Neuron.** The plot shows the output frequency as a function of the input frequency for a neuron with a high gain transfer function. The left curve is $f^{\text{upper}}$, (see Equation (4.51)), the right curve is $f^{\text{lower}}$, (see Equation (4.52)), and the middle curve is $f^{\text{ave}}$, the estimate for the actual output frequency, (see Equation (4.55)). An "$*$" denotes where $f^{\text{ave}}$ is at the midpoint of its asymptotic limits. The gain, $\Gamma$, is defined as the slope of the transfer function at this point, (see Equation (4.56)). The parameter of $f^{\text{max}}$ sets the maximum possible output frequency of the neuron. The parameter of $f^{\text{in}_o}$ is the input frequency where the neuron can first start producing output spikes. And the parameter of $f^{\text{in}_*}$ is the input frequency where the minimum output frequency, $f^{\text{lower}}$, equals 95% of $f^{\text{max}}$.

parameters labeled. The first parameter is the maximum output frequency of the neuron, $f^{\max}$. The second parameter is the input frequency required to initiate output spikes, $f^{\text{in}_o}$; i.e., the value of $f^{\text{in}}$ where the $f^{\text{upper}}$ frequency curve rises above zero, (see Figure 4.15). And the third parameter is the input frequency where the minimum output frequency, $f^{\text{lower}}$, is at 95% of $f^{\max}$, called $f^{\text{in}_*}$. (The gain of the transfer function, defined in Equation (4.56) could have been used as a design parameter instead of $f^{\text{in}_*}$; however, it is more informative to specify $f^{\text{in}_*}$; e.g., how does the output behavior vary between two neurons with gains of 100 and 200?) Each of these design parameters is associated with a different constraint equation, which the neuron's parameters must satisfy. The constraints are:

1. The maximum output frequency must never exceed $f^{\max}$, regardless of the input frequency. The corresponding constraint equation is:

$$\Theta_i = \tanh^2\left(\frac{1}{\tau_i \cdot f^{\max}}\right) \cdot [R_{i0} + R_{ij}] \tag{6.1}$$

   Notice that the input neurotransmitter can never cause the neuron's potential to increase by more than $R_{ij}$.

2. The input frequency must be greater than $f^{\text{in}_o}$ before the neuron can produce any output spikes. If $R_{i0} + R_{ij}\tanh\left(\frac{T_{ij}\mathrm{e}^{-1}}{R_{ij}}\right) > \Theta_i$, then one lone input spike is sufficient for producing an output spike. In some situations, it is may be desirable to have such a neuron, which can quickly respond to input spikes. (When only one input spike is required to produce an output spike, $f^{\text{in}_o}$ can be set to zero.) But in other situations, it may be best if the neuron does not start oscillating unless several initial spikes are input. Thus, the neuron parameters must also be chosen so that:

$$\Theta_i = R_{i0} + R_{ij}\tanh\left[\frac{T_{ij}}{R_{ij}} \cdot \left(\frac{\mathrm{e}^{-1}}{\kappa\left(\frac{1}{\tau_{ij} f^{\text{in}_o}}\right)}\right)\right] \tag{6.2}$$

   This equation uses the maximum possible value for $U_{ij}(t)$ in an infinite spike train with $P^1 = \frac{1}{f^{\text{in}_o}}$, (see Equation (4.25)).

3. When the input frequency is at $f^{\text{in}*}$, the output frequency must be greater than 95% of $f^{\max}$. To assure that this condition is satisfied, the neuron parameters must be chosen so that the output frequency is $0.95 \cdot f^{\max}$ when the neurotransmitter level is at its minimum value; i.e., the output spike frequency is set as if the neurotransmitter did not vary with each input spike but instead remained at the constant level of $\min\{U_{ij}(t)\}$ given in Equation (4.26)). Thus, the constraint equation associated with $f^{\text{in}*}$ is:

$$\Theta_i = \tanh^2\left(\frac{1}{0.95\tau_i \cdot f^{\max}}\right) \cdot \left\{ R_{i0} + R_{ij} \tanh\left[\frac{T_{ij}}{R_{ij}} \cdot \left(\frac{\psi\left(\frac{1}{\tau_{ij} f^{\text{in}*}}\right)}{1 - e^{-\left(\frac{1}{\tau_{ij} f^{\text{in}*}}\right)}}\right)\right]\right\} \tag{6.3}$$

Since the amount of neurotransmitter at the synapse varies with each input spike, using the minimum amount of neurotransmitter in this equation guarantees that the neuron's output frequency will be greater than $0.95 \cdot f^{\max}$ when the input frequency is $f^{\text{in}*}$.

These three equations constrain the choices for the high gain neuron's parameters. Solving for $R_{i0}$ in Equation (6.2), substituting into Equations (6.1) and (6.3), solving for $\left(\frac{\Theta_i}{R_{ij}}\right)$ in these new equations, and setting them equal gives:

$$\frac{1 - \tanh\left[\frac{T_{ij}}{R_{ij}} \cdot \left(\frac{e^{-1}}{\kappa\left(\frac{1}{\tau_{ij} f^{\text{in}_o}}\right)}\right)\right]}{\tanh\left[\frac{T_{ij}}{R_{ij}} \cdot \left(\frac{\psi\left(\frac{1}{\tau_{ij} f^{\text{in}*}}\right)}{1 - e^{-\left(\frac{1}{\tau_{ij} f^{\text{in}*}}\right)}}\right)\right] - \tanh\left[\frac{T_{ij}}{R_{ij}} \cdot \left(\frac{e^{-1}}{\kappa\left(\frac{1}{\tau_{ij} f^{\text{in}_o}}\right)}\right)\right]}$$

$$= \left[\frac{\tanh^2\left(\frac{1}{0.95\tau_i f^{\max}}\right)}{\tanh^2\left(\frac{1}{\tau_i f^{\max}}\right)}\right] \cdot \left[\frac{1 - \tanh^2\left(\frac{1}{\tau_i f^{\max}}\right)}{1 - \tanh^2\left(\frac{1}{0.95\tau_i f^{\max}}\right)}\right] \tag{6.4}$$

The RHS of this equation is always positive, and as $\tau_i$ approaches infinity, it goes to $\left(\frac{1}{0.95}\right)^2$. As $\tau_i$ approaches zero, the RHS goes to infinity. Thus, this equation can always be satisfied for any set of parameter values provided that:

$$\frac{\psi\left(\frac{1}{\tau_{ij} f^{\text{in}*}}\right)}{1-\mathrm{e}^{-\left(\frac{1}{\tau_{ij} f^{\text{in}*}}\right)}} > \left(\frac{\mathrm{e}^{-1}}{\kappa\left(\frac{1}{\tau_{ij} f^{\text{in}_o}}\right)}\right) \tag{6.5}$$

This equation simply implies that the minimum neurotransmitter concentration at the synapse whose input frequency is $f^{\text{in}*}$ must be greater than the maximum neurotransmitter concentration at the synapse whose input frequency is $f^{\text{in}_o}$. As long as $f^{\text{in}*} > f^{\text{in}_o}$, there is a range of synaptic time constants, $\tau_{ij}$, for which this is true.[1] But for values of $f^{\text{in}_o}$ close to $f^{\text{in}*}$, large values of $\tau_{ij}$ are required.

If several spikes arrive nearly simultaneously, $N$ input spikes are required to produce an output spike, where $N$ is given by:

$$N = \left\lceil \left(\frac{R_{ij}\cdot\mathrm{e}}{T_{ij}}\right)\cdot\text{Atanh}\left(\frac{\Theta_i - R_{i0}}{R_{ij}}\right) \right\rceil \tag{6.6}$$

But if the input spikes are beating with $f^{\text{in}} > f^{\text{in}_o}$, then the number of required spikes is determined by the neuron's latency (see Equation (E.14)). And if $f^{\text{in}} \leq f^{\text{in}_o}$, the neurotransmitter concentration is never sufficient for producing output spikes.

As an example, consider designing a high gain neuron with the output frequency being between 0.95 and 1.0, for all input spikes with an input frequency greater than 0.5, ($f^{\text{max}} = 1.0$, $f^{\text{in}*} = 0.5$). Also, assume that the neuron is not to produce any output spikes unless the average input frequency is greater than 0.25, ($f^{\text{in}_o} = 0.25$). To satisfy the constraint equations, the neuron's parameters were chosen as: $\tau_i = 0.3$, $R_{i0} = 0.4663$, $R_{ij} = 0.03623$, $T_{ij} = 0.1209$, $d_{ij} = 0.0$, $\tau_{ij} = 1.5730$, and $\Theta_i = 0.5$. When these values are substituted in Equation (4.56) the neuron's gain, $\Gamma$, is calculated to be $\approx 300$. The transfer function for this neuron was previously shown in Figure 6.1, and Figure 6.2 shows some sample output spikes. Also, to designate a high gain neuron, a "$\Gamma$" is place inside the neuron symbol.

[1]Notice that for a *fixed* synaptic time constant, $\tau_{ij}$, the value of $f^{\text{in}}$ where the $f^{\text{lower}}$ frequency curve rises above zero is not an independent quantity, but is determined by $f^{\text{in}_o}$, (see Figure 6.2). Also notice that $f^{\text{in}*}$ must be greater than this value.

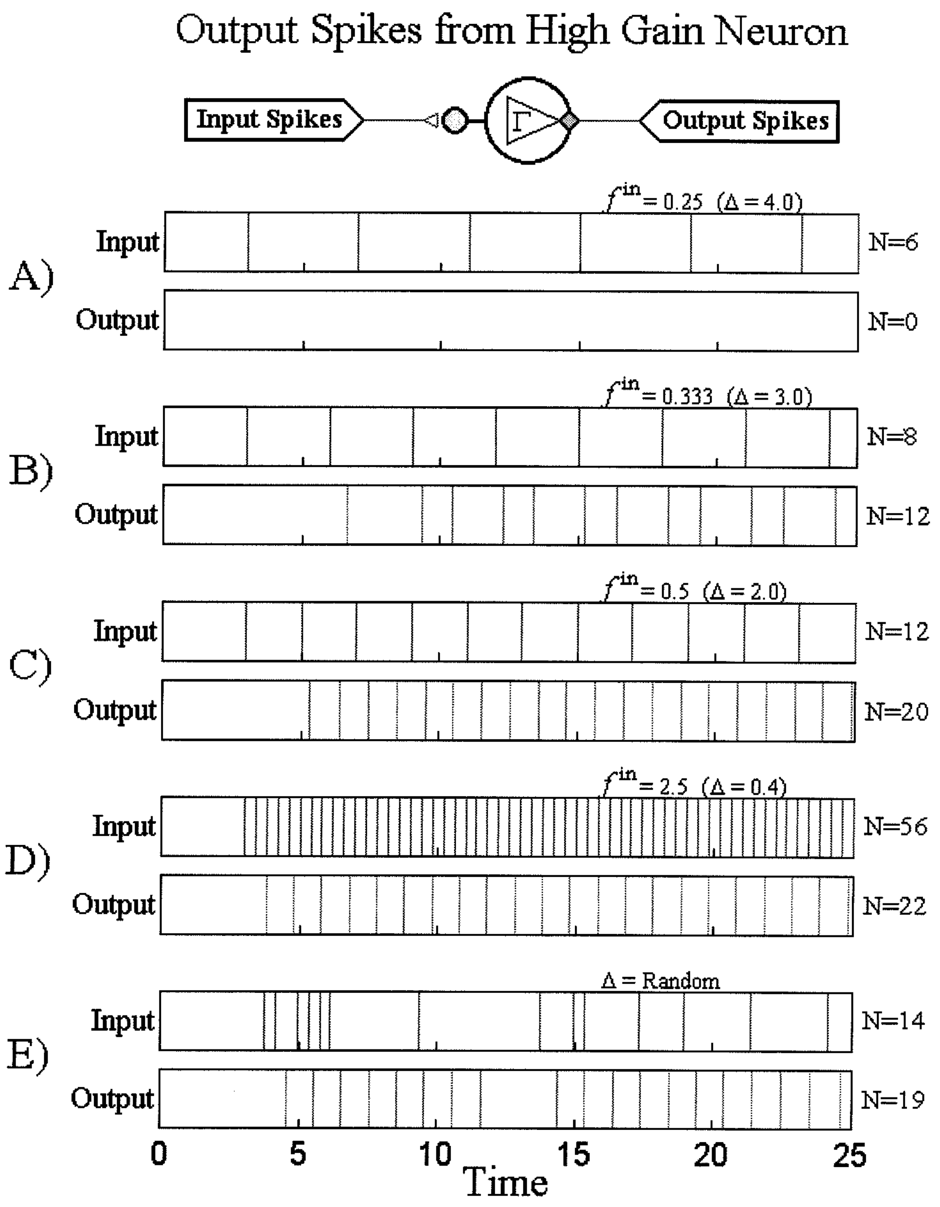


Figure 6.2: **Sample Output for High Gain Neuron.** The plot shows the input and output spikes at 5 different frequencies for the example high gain neuron described in the text. The numbers on the right side, $N$, represent the number of spikes. In (A), the input frequency is 0.25, which is the maximum input frequency for which the neuron is unable to produce any spikes. (B) shows the output spikes when $f^{\text{in}} = 0.333$. This input frequency is sufficient for producing spikes, but below 0.3919, the frequency required to produce output spikes on the $f^{\text{lower}}$ curve of Figure 6.1. (C) shows the output spikes when $f^{\text{in}} = f^{\text{in}*} = 0.5$. (D) shows the output spikes when $f^{\text{in}} = 2.5$. Notice that the output frequency appears to be almost identical to that of (C). (E) shows an output spike train for random input spikes. Due to the high gain, the neuron either does not fire, or produces output spikes at a frequency close to 1.

## 6.3 Memory Oscillators

### 6.3.1 Introduction

Previously, Section 4.3.4 described beating and bursting neurons, which have an endogenous oscillatory spiking behavior. This section describes neurons that are capable of self-sustaining oscillations. They are not endogenous oscillators, but do so when stimulated by an initial input signal, and continue to oscillate until an inhibitory signal is received. Thus, these neurons can be thought of as memory neurons, and are analogous to digital flip-flop circuits, which store binary values.

Two different types of memory oscillators are presented. They vary in the configuration of the input signals used to control their behavior, and represent only a sampling of the numerous ways that the SNM can be connected to perform "memory-like functions." Section 6.3.2 discusses the Dual Control Memory Neuron, which uses two input signals; one is excitatory and starts oscillations, while the other is inhibitory and stops oscillations. Section 6.3.3 discusses the Single Control Memory Neuron, which uses only one input signal, and every time the neuron receives a spike, it switches from oscillating to silent, or vice versa.

### 6.3.2 Dual Control Memory Neuron

Figure 6.3 shows a diagram of a Dual Control Memory Neuron, which uses two input signals. The first input connects at an excitatory synaptic cluster, while the second input connects at an inhibitory synapse. The neuron also has a feedback connection, which may contain delay, going into the excitatory cluster. The neuron parameters are chosen so that when the neuron is silent, (not producing spikes), and an excitatory input spike arrives at the "ON" input, the neuron produces an output spike. The excitatory feedback connection causes the neuron to oscillate after producing an initial output spike. It remains in a stable oscillatory state until a spike is received at the "OFF" input, whereupon the oscillation ceases.

To understand how the parameters must be chosen, it is necessary to examine the

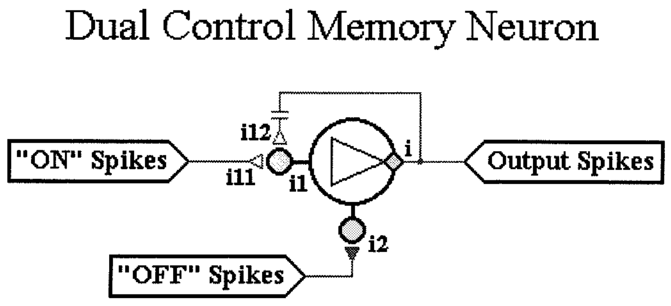


Figure 6.3: **Dual Control Memory Neuron.** This neuron uses two input signals: one is excitatory and initiates oscillations, while the other is inhibitory and stops oscillations. An excitatory feedback connection allows the oscillations to be self-sustaining once an initial output spike is produced. Both excitatory connections go into the same synaptic cluster, while the inhibitory input has a separate synapse. The labels by the connections refer to the subscripts used on their corresponding variables: "$i$" refers to the neuron number; "$i1$" refers to the excitatory synaptic cluster; "$i11$" refers to the excitatory input connection; "$i12$" refers to the excitatory feedback connection; and "$i2$" refers to the inhibitory synapse.

equation for the neuron's potential voltage, which is given by:

$$\begin{aligned} V_i(t) &= \tanh^2\left(\frac{t-t_i^K}{\tau_i}\right)\cdot\left\{R_{i0} + R_{i1}\cdot\tanh\left[\frac{1}{R_{i1}}\cdot(T_{i11}\cdot U_{i11} + T_{i12}\cdot U_{i12})\right]\right. \\ &\qquad \left. + R_{i2}\cdot\tanh\left(\frac{T_{i2}}{R_{i2}}\cdot U_{i2}\right)\right\} \end{aligned} \tag{6.7}$$

The variable subscripts correspond to the labels shown in Figure 6.3. For this memory neuron to work as desired, three constraints must be satisfied:

1. When there is no inhibitory signal, a single excitatory input spike must be sufficient for producing an output spike. Therefore, it is necessary for the maximum neurotransmitter released due to an excitatory input spike to be large enough to cause the neuron to reach threshold; i.e.:

$$\Theta_i \leq R_{i0} + R_{i1}\cdot\tanh\left(\frac{T_{i11}}{R_{i1}}\cdot \mathrm{e}^{-1}\right) \tag{6.8}$$

Here, $\mathrm{e}^{-1}$ is used for the maximum value of $F(t)$, (see Equation (A.4)). Notice that the refractory coefficient, $\tanh^2\left(\frac{t-t_i^K}{\tau_i}\right)$, is assumed to be one. If it is significantly less than one, then the neuron must have fired recently, which indicates that it is already oscillating.

2. A single output spike must be sufficient for initiating the neuron's oscillatory behavior. Obviously, if a feedback spike can cause the neuron to produce another output spike, then the production of an initial output spike leads to a self-sustaining oscillatory behavior. This constraint is satisfied if the maximum neurotransmitter released due to a spike at the feedback connection is sufficient for triggering another output spike; i.e.:

$$\Theta_i \leq \tanh^2\left(\frac{d_{i12}+\tau_{i12}}{\tau_i}\right)\cdot\left\{R_{i0}+R_{i1}\cdot\tanh\left(\frac{T_{i12}}{R_{i1}}\cdot\mathrm{e}^{-1}\right)\right\} \qquad (6.9)$$

   Notice that the maximum neurotransmitter concentration occurs when $t = t_i^K + d_{i12}+\tau_{i12}$, (see Equation (3.5)), and since the neuron must have fired to produce the feedback spike, this time is substituted into the refractory coefficient.

3. A single inhibitory input spike must be sufficient for preventing the production of output spikes; i.e., it must cause the oscillations to cease. Also, when an excitatory and inhibitory input signal arrive at nearly the same time, the inhibitory signal should "win" and the neuron should not produce output spikes. The maximum neurotransmitter due to an inhibitory spike is $|T_{i2}\mathrm{e}^{-1}|$, so if:

$$\Theta_i > R_{i0}+R_{i1}+R_{i2}\cdot\tanh\left[\frac{T_{i2}}{R_{i2}}\cdot\mathrm{e}^{-1}\right] \qquad (6.10)$$

   then after an inhibitory spike arrives, there must be a period of time during which the neuron is unable to produce an output spike, regardless of the amount of neurotransmitter at the excitatory input connections. (Notice that the maximum contribution to the neuron's potential from the excitatory synaptic cluster is bounded by $R_{i1}$. Also note that for an inhibitory connection $T_{i2} < 0$.)

   However, Equation (6.10) is not sufficient to guarantee that the oscillatory behavior ceases. The case may be such that the effect from the inhibitory neurotransmitter rapidly decays, while the residual amount of excitatory neurotransmitter from the previous feedback spikes remains sufficient for producing another output spike, leading back to oscillation. To assure that this can not happen, the effect from the inhibitory neurotransmitter must be of a sufficiently long duration for the excitatory neurotransmitter to decay below a self-sustaining level. Therefore, to determine the maximum time that the neurotransmitter concentration at the feedback connection is sufficiently high enough for producing another output spike, assume that the neuron is in a stable oscillatory pattern with $P^1 = \Delta$.[2] To calculate the exact time, it is necessary to include the refractory coefficient; however, since this only increases the effective value for $\Theta_i$, the refractory coefficient can be ignored, yielding an upper bound on the time being sought. Using the expression for the neurotransmitter given in Equation (4.23), the maximum time, $\bar{t}^{\max}$, is found from:

$$\Theta_i = R_{i0} + R_{i1}{\cdot}\tanh\left[\frac{T_{i12}}{R_{i1}{\cdot}\kappa\left(\frac{\Delta}{\tau_{i12}}\right)}{\cdot}F\left(\frac{\bar{t}^{\max} - d_{i12}}{\tau_{i12}} + \psi\left(\frac{\Delta}{\tau_{i12}}\right)\right)\right] \tag{6.11}$$

Solving for $\bar{t}^{\max}$ yields:

$$\begin{aligned} \bar{t}^{\max} &= \tau_{i12}{\cdot}\left(x - \psi\left(\frac{\Delta}{\tau_{i12}}\right)\right) + d_{i12} && (6.12) \\ \text{where:}\quad [a, x] &= F^{-1}\left[\left(\frac{R_{i1}}{T_{i12}}\right){\cdot}\kappa\left(\frac{\Delta}{\tau_{i12}}\right){\cdot}\text{Atanh}\left(\frac{\Theta_i - R_{i0}}{R_{i1}}\right)\right] && (6.13) \end{aligned}$$

(An algorithm for calculating $F^{-1}(-)$ is given in Appendix D. See footnote on page 328 concerning the notation used in Equation (6.13).) Thus, if the effects from an inhibitory spike are able to prevent a neuron from firing for at least $\bar{t}^{\max}$, then the neuron must stop oscillating. This is the case if:

$$\Theta_i > R_{i0} + R_{i1} + R_{i2}{\cdot}\tanh\left[\frac{T_{i2}}{R_{i2}}{\cdot}F\left(\psi\left(\frac{\bar{t}^{\max}}{\tau_{i2}}\right)\right)\right] \tag{6.14}$$

Notice that from the definition of $\psi(-)$ in Figure C.1, the neurotransmitter

[2]If a neuron is oscillating due to a feedback connection, then the time between output spikes can be found by solving for $\Delta$ in Equations (4.44)-(4.46). (See also Figure 4.11.)

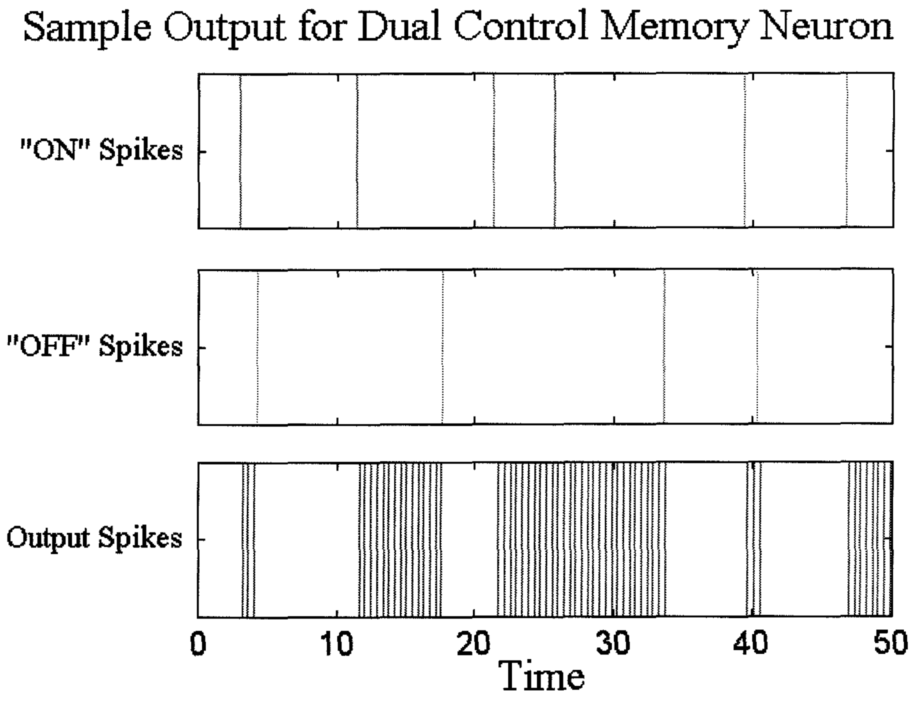


Figure 6.4: **Sample Output for Dual Control Memory Neuron.** This plot shows a set of random spikes for the "ON" and "OFF" inputs, and the resulting output spikes from the neuron. When the neuron is in an oscillatory state, $\Delta = 0.423$. The neuron's parameters are: $\Theta_i = 0.5$, $R_{i0} = 0.0$, $\tau_i = 0.2$, $R_{i1} = 0.53$, $T_{i11} = 2.75$, $\tau_{i11} = 0.375$, $T_{i12} = 2.75$, $d_{i12} = 0.20$, $\tau_{i12} = 0.375$, $R_{i2} = 0.10$, $T_{i2} = -0.10$, and $\tau_{i2} = 1.0$. Also, $\bar{t}^{\max}$ in Equation (6.13) is 1.18.

released from an inhibitory input spike is sufficiently high for at least $\bar{t}^{\max}$. Because the inhibitory spike may arrive at any time within the oscillatory cycle, it remains undetermined as to when this inhibitory period starts relative to the last output spike. But, if no output spikes are allowed to occur during any period of time greater than or equal to $\bar{t}^{\max}$, then the neuron's oscillation ceases.

When the neuron's parameters are chosen to satisfy the constraints of Equations (6.8), (6.9), and (6.14), the neuron operates as desired: "ON" spikes start oscillations, while "OFF" spikes stop oscillations. Figure 6.4 shows a sample output for a neuron with parameters satisfying the constraints. Notice that when the neuron is already oscillating, another "ON" spike does not significantly affect the output spike train. The parameters in this example were chosen so that the neurotransmitter released at the feedback connection forces the excitatory synaptic cluster to be nearly saturated, i.e., $\tanh(-) \approx 1$. Thus, additional neurotransmitter from an input spike will not significantly perturb its oscillatory cycle. (This same memory cell could have been designed with separate synapses for the excitatory input and feedback connections. But then each synapse would effect the neuron's potential independently and additional excitatory input spikes would cause temporary increases in the neuron's output frequency.) Although not demonstrated in Figure 6.4, an excitatory input spike that arrives a short time after an inhibitory input spike will not initiate an output spike. After each inhibitory spike, there is a "dead time" of approximately $\bar{t}^{\max}$ during which the neuron is insensitive to excitatory inputs. The actual length of time depends upon the relative phases between the feedback spike, inhibitory spike, and excitatory spike.

### 6.3.3 Single Control Memory Neuron

Figure 6.5 shows a diagram of a Single Control Memory Neuron, which has only one input signal. The input signal connects to the neuron at both excitatory and inhibitory synapses.[3] Like the previous memory neuron, it uses a feedback connection to sustain oscillations. The feedback signal forms both an excitatory synapse into the neuron and an inhibitory presynapse onto the excitatory input signal connection. The parameters are chosen so that when the neuron is silent and an input spike arrives, the excitatory synapse overpowers the inhibitory synapse causing the neuron to start producing output spikes. The excitatory feedback connection causes the neuron to continue oscillating after producing an initial output spike. It remains in a stable oscillatory state until another input spike is received. The inhibitory presynapse from the feedback signal prevents the input spike from exciting the neuron; however, the inhibitory connection is unhampered, and causes the oscillation to cease. Thus, this neuron switches between the oscillatory and silent states with each input spike.

To understand how the parameters must be chosen, it is useful to examine the

[3]Real neurons are not believed to form both excitatory and inhibitory connections (Dale's law – see footnote on page 19.) Hence, this method for creating a memory oscillatory is biologically implausible – at least with a single neuron.

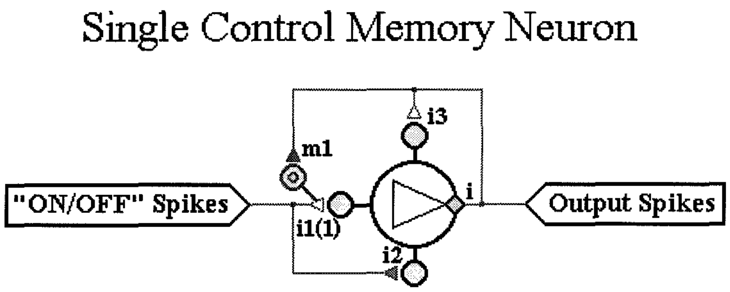


Figure 6.5: **Single Control Memory Neuron.** This neuron has one input signal, which forms both excitatory and inhibitory connections. An excitatory feedback connection allows the oscillations to be self-sustaining once an initial output spike is produced. The excitatory input connection is stronger than the inhibitory input connection, and when the neuron is not oscillating, an input spike has a net excitatory effect on the neuron, leading to oscillation. When the neuron is oscillating the excitatory input connection is inhibited by the presynaptic connection from the neuron's feedback signal, and the net effect from an input spike is inhibitory, causing the neuron to stop oscillating. The labels by the connections refer to the subscripts used on their corresponding variables: "$i$" refers to the neuron number; "$i1(1)$" refers to the excitatory input connection which has a presynapse labeled "$m1$" attached to it; "$i2$" refers to the inhibitory input connection; and "$i3$" refers to the excitatory feedback connection.

equation for the neuron's potential voltage, which is given by:

$$\begin{aligned} V_i(t) \;=\; & \tanh^2\left(\frac{t-t_i^K}{\tau_i}\right)\cdot\left\{R_{i0} + R_{i1}\tanh\left(\frac{T_{i1}}{R_{i1}}\cdot U_{i1(1)}\right)\right. \\ & \left. + R_{i2}\tanh\left(\frac{T_{i2}}{R_{i2}}\cdot U_{i2}\right) + R_{i3}\tanh\left(\frac{T_{i3}}{R_{i3}}\cdot U_{i3}\right)\right\} \end{aligned} \tag{6.15}$$

The variable subscripts correspond to the labels shown in Figure 6.5. For this memory neuron to work as desired, three constraints need to be satisfied:

1. When the neuron is silent, an input spike must cause the neuron to produce an output spike. If the input spike arrives at $t = 0$, then an initial output spike is produced if there is a value of $t$ such that:

$$\Theta_i \leq R_{i0} + R_{i1}\cdot\tanh\left[\frac{T_{i1}}{R_{i1}}\cdot F\left(\frac{t}{\tau_{i1}}\right)\right] + R_{i2}\cdot\tanh\left(\frac{T_{i2}}{R_{i2}}\cdot F\left(\frac{t}{\tau_{i2}}\right)\right) \tag{6.16}$$

If the neuron parameters are chosen so that $\tau_{i1} = \tau_{i2}$, and $\frac{T_{i1}}{R_{i1}} = \left|\frac{T_{i2}}{R_{i2}}\right|$,[4] then this equation simplifies to:

$$\Theta_i \leq R_{i0} + (R_{i1} - R_{i2}) \cdot \tanh\left(\frac{T_{i1}}{R_{i1}} \cdot F\left(\frac{t}{\tau_{i1}}\right)\right) \tag{6.17}$$

2. A single output spike must be sufficient for initiating the neuron's oscillatory behavior. Obviously, if a feedback spike can cause the neuron to produce another output spike, then the production of an initial output spike leads to a self-sustaining oscillatory behavior. This constraint is satisfied if the maximum neurotransmitter released due to a spike at the feedback connection is sufficient for triggering another output spike; i.e.:

$$\Theta_i \leq \tanh^2\left(\frac{\tau_{i3}}{\tau_i}\right) \cdot \left[R_{i0} + R_{i3} \cdot \tanh\left(\frac{T_{i3}}{R_{i3}} \cdot \mathrm{e}^{-1}\right)\right] \tag{6.18}$$

   Notice that the maximum neurotransmitter occurs when $t = t_i^K + \tau_{i3}$, (see Equation (3.5)), and since the neuron must have fired to produce the feedback spike, this time is substituted into the refractory coefficient. (The feedback connection may also contain delay, as in Dual Control Memory Neuron.)

3. When the neuron is oscillating, an input spike must cause the neuron to stop oscillating. This constraint is satisfied if the inhibitory input connection is sufficiently strong to prevent the production of output spikes. If the period between output spikes is $\Delta$, then the total contribution to the neuron's potential due to an input spike is given by:

$$R_{i1} \cdot \tanh\left[\left(\frac{T_{i1}}{R_{i1}}\right) \cdot F\left(\frac{t}{\tau_{i1}}\right) \cdot \mathrm{e}^{-R_{m1} \cdot \tanh\left[\left(\frac{T_{m1}}{R_{m1}}\right) \frac{F\left(\psi\left(\frac{\Delta}{\tau_{m1}}\right)\right)}{\kappa\left(\frac{\Delta}{\tau_{m1}}\right)}\right]}\right]$$

[4]These restrictions on the neuron's parameters are not necessary for the memory neuron to operate as desired, but they simplify the analysis. In general, $\tau_{i2}$ should be less than or equal to $\tau_{i1}$ to ensure that the inhibitory effects do not outlast the excitatory effects and stop oscillation after only a few initial output spikes have been produced.

$$+R_{i2}\cdot\tanh\left[\left(\frac{T_{i2}}{R_{i2}}\right)\cdot F\left(\frac{t}{\tau_{i2}}\right)\right] \tag{6.19}$$

This represents the worst case scenario with the inhibitory presynaptic neurotransmitter on the excitatory synapse being at a minimum, (see Equation (4.26)).

Now, the contribution to the neuron's potential must inhibit the neuron from firing. But as with the Dual Control Memory Neuron, it is possible that the inhibitory effects from the input spike decay more rapidly than the residual neurotransmitter at the feedback synapse, causing the neuron to start oscillating again. Thus, the neuron must be inhibited long enough for the excitatory neurotransmitter to decay below a self-sustaining level. The required time, $\bar{t}^{\mathrm{max}}$, given in Equations (6.12)-(6.13) still applies to this memory neuron, with the appropriate changes in the parameter subscripts. Thus, if the effects from an input spike are able to prevent a neuron from firing for at least $\bar{t}^{\mathrm{max}}$, then the neuron must stop oscillating. This is the case if:

$$\begin{aligned}\Theta_i \quad > \quad & R_{i0} + R_{i3}\cdot\tanh\left(\frac{T_{i3}\mathrm{e}^{-1}}{R_{i3}\cdot\kappa\left(\frac{\Delta}{\tau_{i3}}\right)}\right) + R_{i2}\cdot\tanh\left[\left(\frac{T_{i2}}{R_{i2}}\right)\cdot F\left(\psi\left(\frac{\bar{t}^{\mathrm{max}}}{\tau_{i2}}\right)\right)\right] \\ & + R_{i1}\cdot\tanh\left[\left(\frac{T_{i1}}{R_{i1}}\right)\cdot F\left(\psi\left(\frac{\bar{t}^{\mathrm{max}}}{\tau_{i1}}\right)\right)\cdot\mathrm{e}^{-R_{m1}\cdot\tanh\left[\left(\frac{T_{m1}}{R_{m1}}\right)\frac{F\left(\psi\left(\frac{\Delta}{\tau_{m1}}\right)\right)}{\kappa\left(\frac{\Delta}{\tau_{m1}}\right)}\right]}\right]\end{aligned} \tag{6.20}$$

When this constraint equation is satisfied, an input spike cause the neurotransmitter at the inhibitory synapse to remain sufficiently high for at least $\bar{t}^{\mathrm{max}}$, and the output oscillation must cease.

When the neuron's parameters are chosen to satisfy Equations (6.16), (6.18), and (6.20), the neuron operates as desired, switching between a silent and oscillatory state with each input spike. Figure 6.6 shows a sample output for a neuron with parameters satisfying the constraints. Notice that the derivations of the constraint equations were based on the assumption that the neuron was already in a steady-state condition; i.e., if the neuron is silent, then there is no neurotransmitter at any of the

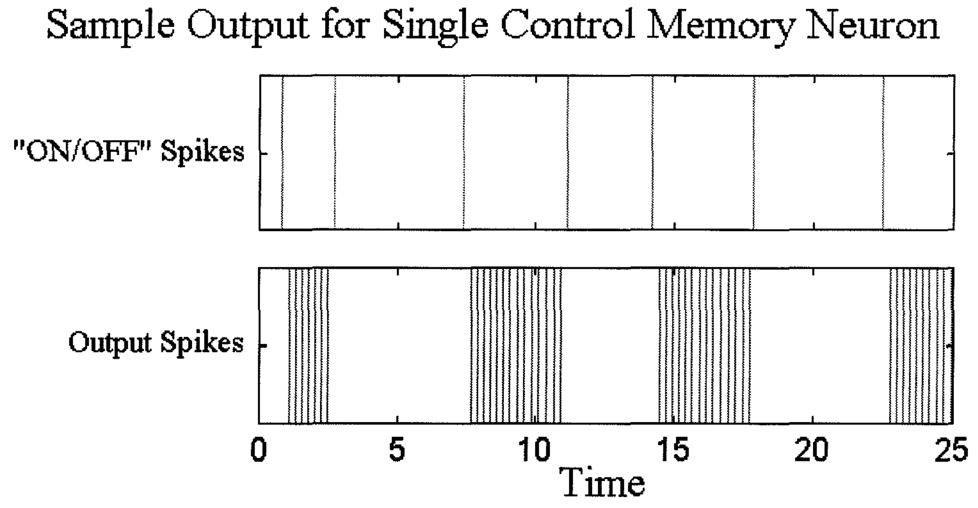


Figure 6.6: **Sample Output for Single Control Memory Neuron.** This plot shows a set of random input spikes, and the resulting output spikes from the neuron. When the neuron is in an oscillatory state, $\Delta = 0.265$. The neuron's parameters are: $\Theta_i = 0.5$, $R_{i0} = 0.0$, $\tau_i = 0.3$, $R_{i1} = 8.5$, $T_{i1} = 8.5$, $\tau_{i1} = 0.4$, $R_{i2} = 7.0$, $T_{i2} = -7.0$, $\tau_{i2} = 0.4$, $R_{i3} = 1.0$, $T_{i3} = 2.0$, $\tau_{i3} = 0.435$, $R_{m1} = 6.0$, $T_{m1} = -6.0$, and $\tau_{m1} = 0.25$. Also, $\bar{t}^{\max}$ is 1.53.

synapses, and similarly, if the neuron is oscillating, then there is only neurotransmitter at the feedback synapse and presynapse, which is given by the periodic solution for $P^1$ found in Section 4.2.3. But if the neuron has recently received an input spike and switched its state, a second input spike arriving shortly after the first may not be able to cause the neuron to switch states again. Essentially, there is a dead time of approximately $\bar{t}^{\max}$, during which the neuron is insensitive to the next input spike.

## 6.4 Bounded Threshold Levels

In the SNM as well as in real neurons, a spiking output signal is produced when the input signal exceeds the neuron's threshold level; however, there are some situations where it is desirable to have the neuron only respond when the input signal is within a limited range. Such neurons can be used to route input signals along network paths with different gains. A bounded threshold neuron fails to produce any spikes for large or small input signals outside of the specified range, and are said to have a *bounded threshold level.* In the SNM, bounded thresholds can be effectively realized by using

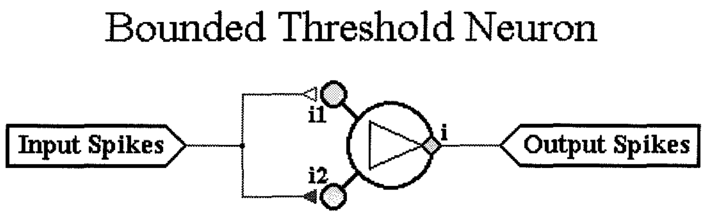


Figure 6.7: **Neuron with a Bounded Threshold Level.** This neuron has one input signal, which forms two synaptic connections: one is excitatory and the other is inhibitory. The labels by the connections refer to the subscripts used on their corresponding variables: "$i$" refers to the neuron number; "$i1$" refers to the excitatory synapse; and "$i2$" refers to the inhibitory synapse.

both excitatory and inhibitory synapses from the same source.

To understand how a bounded threshold level can be created, consider a silent neuron, $(R_{i0} < \Theta_i)$, with a single input signal, which connects to both an excitatory and inhibitory synapse. Although the signal may be from an external stimulus, with no loss in generality, assume the input is from another neuron. Furthermore, assume that both synapses have the same time constants and time delays, $(\tau_{i1} = \tau_{i2}$ and $d_{i1} = d_{i2})$; therefore, $U_{i1} = U_{i2} \equiv U_{ij}(t)$. Figure 6.7 shows a schematic of a bounded threshold neuron.

The potential voltage of the bounded threshold neuron is given by:

$$\begin{aligned} V_i(t) &= \tanh^2\left(\frac{t-t_i^K}{\tau_i}\right)\cdot\left\{R_{i0} + R_{i1}\tanh\left(\frac{T_{i1}}{R_{i1}}\cdot U_{ij}(t)\right)\right. \\ &\qquad\qquad \left. + R_{i2}\tanh\left(\frac{T_{i2}}{R_{i2}}\cdot U_{ij}(t)\right)\right\} \qquad (6.21) \end{aligned}$$

The first synapse is excitatory, $(T_{i1} > 0)$, while the second synapse is inhibitory, $(T_{i2} < 0)$. To further facilitate the analysis, the number of parameters can be reduced, with the firing condition being:

$$\frac{\Theta_i}{R_{i1}} = \tanh^2\left(\frac{t-t_i^K}{\tau_i}\right)\cdot\left[\frac{R_{i0}}{R_{i1}} + \tanh\left(K\cdot h(t)\right) - r\tanh\left(h(t)\right)\right] \qquad (6.22)$$

where: $$K \equiv r\cdot\left|\frac{T_{i1}}{T_{i2}}\right| \qquad (6.23)$$

$$r \equiv \frac{R_{i2}}{R_{i1}} \tag{6.24}$$

$$h(t) \equiv \frac{|T_{i2}|}{R_{i2}} \cdot U_{ij}(t) \tag{6.25}$$

Although the refractory term of $\tanh^2\left(\frac{t-t_i^K}{\tau_i}\right)$ limits the effective input signal, the neuron is incapable of firing unless:

$$B(h) \equiv \tanh\left(K \cdot h(t)\right) - r\tanh\left(h(t)\right) > \frac{\Theta_i}{R_{i1}} - \frac{R_{i0}}{R_{i1}} \equiv \bar{\Theta} \tag{6.26}$$

Now, a neuron can have a bounded threshold level if $B(h)$ attains a maximum; i.e., increasing $h(t)$ actually decreases the value of $B(h)$. For such a maximum to exist, there must be a value for $h(t)$, such that:

$$\frac{dB}{dh} = K\text{sech}^2\left(K \cdot h(t)\right) - r\text{sech}^2\left(h(t)\right) = 0 \tag{6.27}$$

$$\frac{d^2B}{dh^2} = -2K^2\text{sech}^2\left(K \cdot h(t)\right) \cdot \tanh\left(K \cdot h(t)\right) + 2r\text{sech}^2\left(h(t)\right) \cdot \tanh\left(h(t)\right) < 0 \tag{6.28}$$

When Equation (6.27) is substituted into Equation (6.28), it can only be less than zero for $K > 1$. Consequently, $K \cdot h(t) > h(t) > 0$, and since $\text{sech}^2(-)$ is a decreasing function (for positive arguments), there can only be a solution for $h(t)$ in Equation (6.27) if $r < K$. Finally, a bounded threshold implies that there is an upper limit on the values for $h(t)$ which satisfy Equation (6.26). Thus, as $h(t) \to \infty$, $B(h) < \bar{\Theta}$ or $1 - r < \bar{\Theta}$. To summarize, the $r$ and $K$ parameters must be chosen such that:

$$0 < 1 - \bar{\Theta} < r < K \qquad \text{and:} \qquad 1 < K \tag{6.29}$$

With any values of $r$, $K$, and $\bar{\Theta}$ that satisfy Equation (6.29), there is a limited range of $h(t)$ values which can produce a spike. Henceforth, to simplify the analysis, assume $r = 1$, ($R_{i1} = R_{i2}$), which allows the constraints to be satisfied for any $K > 1$, and $0 < \bar{\Theta} < 1$. Figure 6.8 shows $B(h)$ as a function of $h(t)$ for several different values of $K \geq 1$, and Figure 6.9 shows the lines for $h(t)$ where $B(h) = \bar{\Theta}$.

Theoretically, when choosing the neuron parameters to respond to a specified input

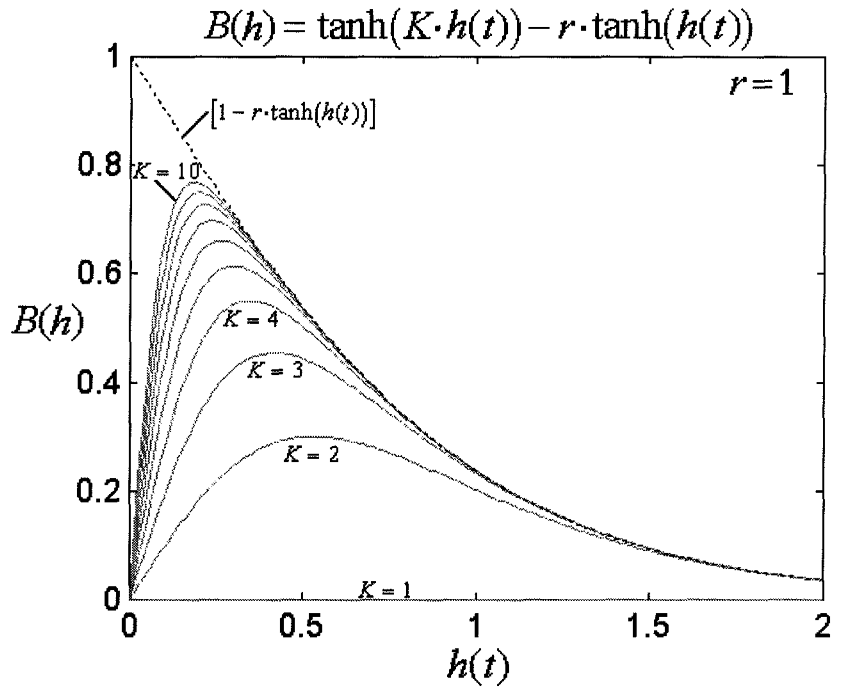


Figure 6.8: **Plot of** $B(h)$**.** The plot shows $B(h)$ vs. $h(t)$ for $K = 1, 2, \ldots, 10$. The value for $r$ is set to 1. Notice that for larger $K$ values, the curve becomes higher and more narrow.

range, any value for $K > 1$ may be used; however, larger $K$ values work best if the ratio of the maximum responsive input neurotransmitter to the minimum responsive input neurotransmitter, is close to 1; i.e.:

$$Q \equiv \frac{U^{\max}}{U^{\min}} \approx 1 \tag{6.30}$$

When $Q$ is close to one, there is a narrow response range.

The corresponding values for $h(t)$ are given by:

$$h^{\min} = \frac{|T_{i2}|}{R_{i2}} \cdot U^{\min} \tag{6.31}$$

$$h^{\max} = \frac{|T_{i2}|}{R_{i2}} \cdot U^{\max} = Q \cdot h^{\min} \tag{6.32}$$

With larger $K$ values, the gradient of the curve in Figure 6.8 is steeper at the boundaries, and there is less ambiguity about when the input signal is within the desired range.

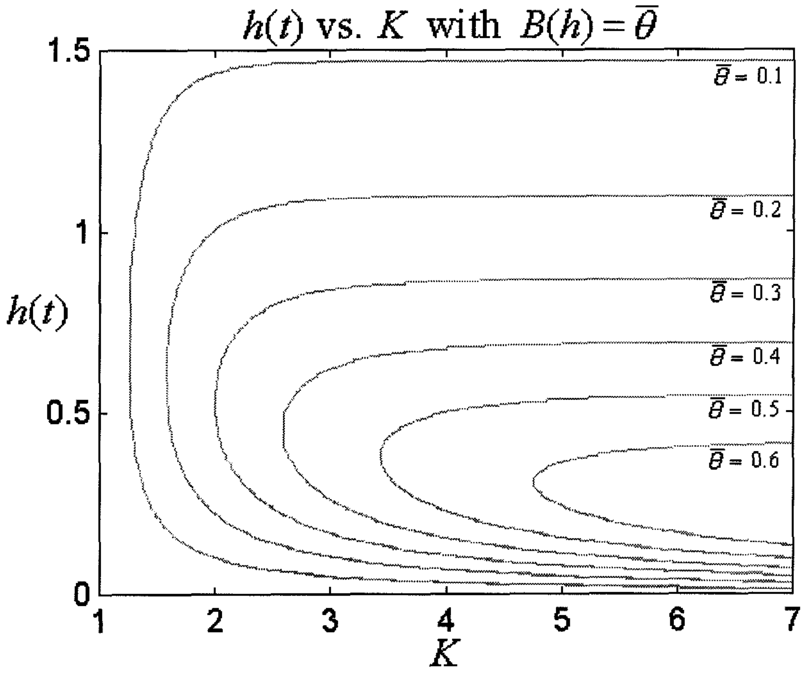


Figure 6.9: **Lines of $h(t)$ where $B(h) = \bar{\Theta}$.** The plot shows the values for $h(t)$, with $r = 1$, where $\tanh(K \cdot h(t)) - \tanh(h(t)) = \bar{\Theta}$. All points to the right of the curves, satisfy the inequality in Equation (6.26). Notice that for any given values of $K > 1$ and $0 < \bar{\Theta} < 1$, there is both an upper and lower limit on the values for $h(t)$ which can produce a spike. As $K$ increases the upper limit approaches $\text{Atanh}\left(\frac{1-\bar{\Theta}}{r}\right)$ and the lower limit decreases like $\left(\frac{\text{Atanh}(\bar{\Theta})}{K}\right)$.

As a design example, consider choosing the parameters so that a neuron will only respond when the level of input neurotransmitter, $U_{ij}(t)$ is between 0.9 an 1.1. For these values, $Q = 1.1/0.9 \approx 1.2222$. Since this $Q$ value is relatively close to 1, $K$ was chosen to be 10. Solving for $h^{\min}$ in the transcendental equation of:

$$\tanh\left(K \cdot h^{\min}\right) - \tanh\left(h^{\min}\right) = \bar{\Theta} = \tanh\left(KQ \cdot h^{\min}\right) - \tanh\left(Q \cdot h^{\min}\right) \qquad (6.33)$$

gives $h^{\min} = 0.1661$ and $\bar{\Theta} = 0.7658$. Letting $R_{i1} = R_{i2} = 1$, and using Equation (6.31) yields $T_{i2} = -0.1846$. ($T_{i2}$ is negative since the second synapse is inhibitory.) With $K = 10$, Equation (6.23) gives $T_{i1} = 1.8456$. Now $\Theta_i$ and $R_{i0}$ can be arbitrarily chosen to satisfy the definition of $\bar{\Theta}$ in Equation (6.26). Figure 6.10 shows how a neuron with these parameters responds to different input levels. (For the beating input stimulus shown, the maximum and minimum neurotransmitter levels converge to the values given in Equations (4.25)-(4.26).)

Although the bounded threshold neuron is designed to respond to inputs within a specific range, one of its possible uses is in signaling the start and end of an input stimulus. Its response range can be made sufficiently small so that nearly all input signals are either above or below the threshold region; however, for those signals above threshold, the neurotransmitter level must pass through the responsive region when the stimulus begins and ends. During these times, the neuron outputs a spike indicating the start or end of a stimulus (see Figure 6.10 (C)).

Finally, notice that if a time delay is used on one of the input synapses, then the bounded threshold neuron shown in Figure 6.7 can be used to detect changes in the input signal. This is because a change in the input signal will alter the neuron's potential through the synapse without delay before the other synapse can counteract its effect. When used in this configuration, the neuron is said to have a *gradient threshold.* If the time delay is associated with the inhibitory synapse, then the neuron is sensitive to sudden increases in the input stimulus. And if the time delay is associated with the excitatory synapse, then the neuron is sensitive to sudden decreases in the input stimulus.

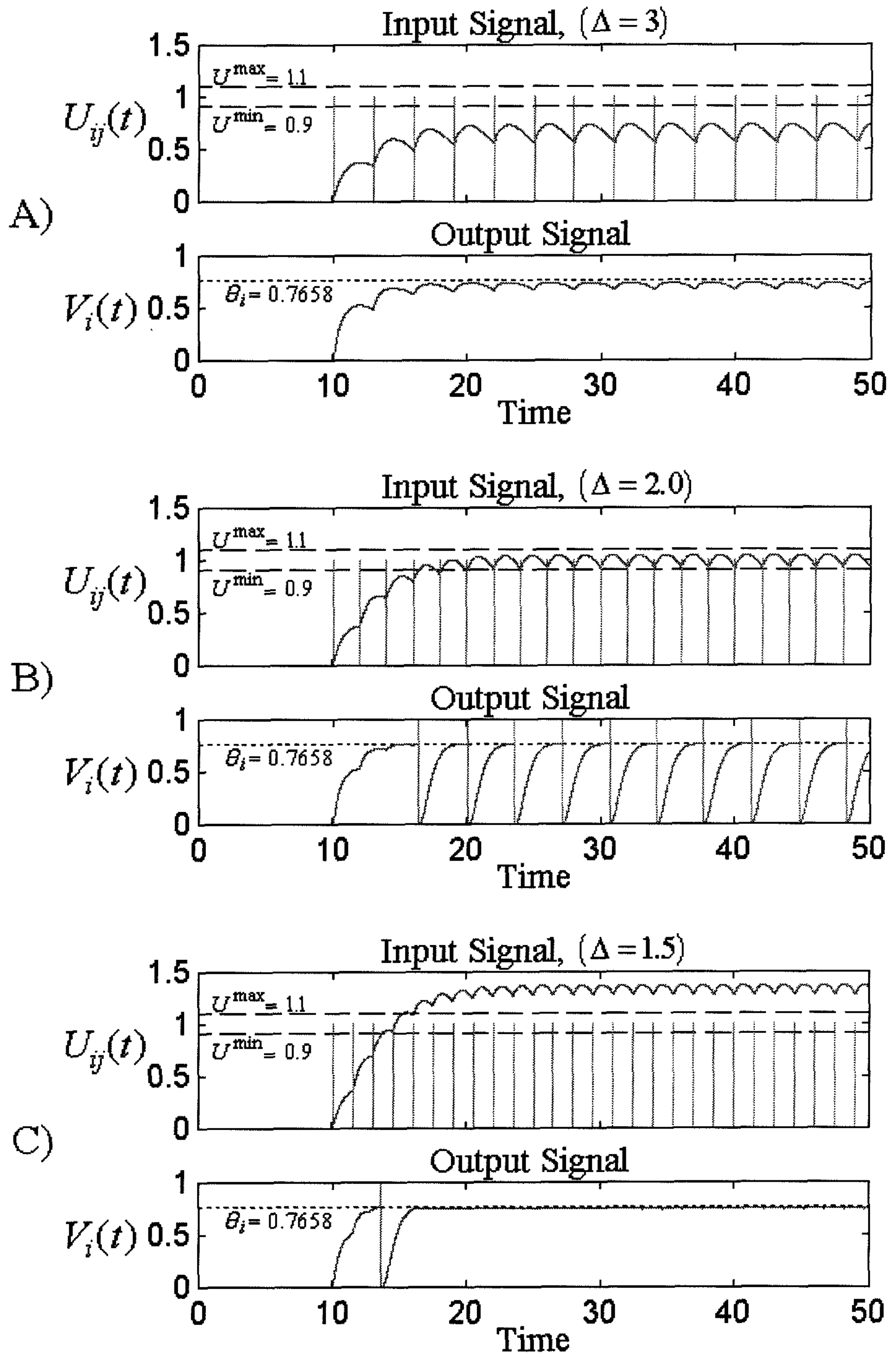


Figure 6.10: **Neuron Response with Bounded Threshold.** The plot shows the neuron's response to the onset of a periodic spike input. In (A), $U_{ij}(t)$ never becomes large enough to produce an output spike. In (B), $U_{ij}(t)$ converges within the bounded threshold region. The neuron produces output spikes at a frequency determined by the refractory period. In (C), $U_{ij}(t)$ increases above the bounded threshold region, but as it passes through, an output spike is generated. ($\tau_i = 1.0$, $R_{i0} = 0.0$, $R_{i1} = 1.0$, $R_{i2} = 1.0$, $T_{i1} = 1.8456$, $T_{i2} = -0.1846$, $d_{ij} = 10.0$, $\tau_{ij} = 2.0$, and $\Theta_i = 0.7658$.)

## 6.5 Identity Neurons

### 6.5.1 Introduction

For some computing problems, it is useful to have a neuron with an identity response, in which the output spikes are correlated with the input spikes. Section 6.5.2 describes an unsynchronized identity neuron, which outputs a spike for each incoming spike. This neuron is useful for summing uncorrelated input signals. The time lag between the input and output spikes depends upon the recent spike history; i.e., spikes occurring within a spike train produce output spikes with less delay than do isolated spikes. (An isolated spike is defined as an input spike with no other recent input spikes preceding it, which are still influencing the neuron's current state.) Essentially this neuron acts as a non-linear delay element, with strong stimuli passing through quickly, while weak stimuli are delayed. (A strong stimulus is represented by a closely spaced spike train, while a weak stimulus is represented by a single isolated spike.)

Section 6.5.3 describes a synchronized identity neuron. For this neuron, the output spikes occur at regular intervals. It can be thought of as a beating neuron (see Section 4.3.4), which has no output beats unless an input spike occurs. The neuron *almost* outputs spikes at a regular frequency, but its potential remains slightly below threshold; however, when a spike is input, it sufficiently increases the neuron's potential to produce an actual spike at the time of the next "pseudospike." This type of neuron is necessary for computations that require synchronized input signals, such as the pattern recognition or logic function evaluation tasks described in Chapter 7.

Section 6.5.4 describes a synchronized inverse neuron. This neuron operates the same as the synchronized identity neuron, except that the signal is inverted; i.e., the neuron only outputs spikes when no spikes are being input. If an input signal is connected to both a synchronized identity neuron and a synchronized inverse neuron, then they produce output spikes opposite times; i.e., during the pseudospike events in which the synchronized identity neuron produces an output spike, the synchronized inverse neuron does not, and vice versa.

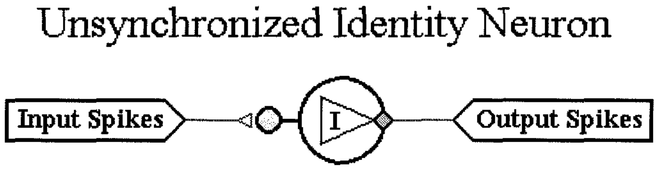


Figure 6.11: **Unsynchronized Identity Neuron.** The neuron shown here has only one input signal, although more may be used. For each input spike, the neuron produces an output spike. To indicate a neuron that has an identity response, "I" is placed inside the neuron symbol.

## 6.5.2 Unsynchronized Identity Neurons

An unsynchronized identity neuron attempts to output a spike for each incoming spike. When designing such a neuron, it is important to remember that the neuron's refractory period limits its maximum output frequency; thus, for closely correlated input spikes, the neuron is unable to produce an output spike for each one. To understand its behavior, consider a neuron with only one input synapse, and assume that the minimum allowed time between input spikes is $\Delta_j$. Figure 6.11 shows an unsynchronized identity neuron. The equation for its potential reduces to:

$$V_i(t) = \tanh^2\left(\frac{t - t_i^K}{\tau_i}\right) \cdot \left[R_{i0} + R_{ij} \tanh\left(\frac{T_{ij}}{R_{ij}} \cdot U_{ij}(t)\right)\right] \qquad (6.34)$$

Since the identity neuron should only produce spikes when there are input spikes, the neuron must be a silent neuron with $R_{i0} < \Theta_i$. A one-to-one correlation between input and output spikes occurs if four constraints are satisfied. The first two constraints characterize the neuron's behavior when spikes first start arriving, while the last two constraints apply to an input spike train that suddenly stops.

1. An isolated input spike must result in an output spike, and to prevent the neuron from lagging behind the input, the output spike should occur before another input spike can arrive; i.e., there must exist some value of $t$, $(0 < t < \Delta_j)$, such that:

$$R_{i0} + R_{ij} \tanh\left(\frac{T_{ij}}{R_{ij}} \cdot F\left(\frac{t}{\tau_{ij}}\right)\right) > \Theta_i \qquad (6.35)$$

Here, $t = 0$ corresponds to the arrival time of the input spike. Notice that since the input spike is isolated, the neuron must not have fired recently; thus, $\tanh^2\left(\frac{t-t_i^K}{\tau_i}\right) \approx 1$. Also, $F\left(\frac{t}{\tau_{ij}}\right)$ is the response of $U_{ij}(t)$ to a single input spike, and its maximum value is $\mathrm{e}^{-1}$.

2. The neuron's refractory period must not prevent the neuron from responding when a second input spike follows the first by only $\Delta_j$; i.e., there must exist some value of $t$, $(\Delta_j \leq t < 2\Delta_j)$, such that:

$$\tanh^2\left(\frac{t-t_i^1}{\tau_i}\right)\cdot\left\{R_{i0} + R_{ij}\tanh\left[\frac{T_{ij}}{R_{ij}}\cdot\left[F\left(\frac{t}{\tau_{ij}}\right) + F\left(\frac{t-\Delta_j}{\tau_{ij}}\right)\right]\right]\right\} \geq \Theta_i \tag{6.36}$$

$$\text{where:} \qquad t_i^1 \equiv \tau_{ij}\cdot F^{-1}\left(\frac{R_{ij}}{T_{ij}}\cdot\mathrm{Atanh}\left(\frac{\Theta_i - R_{i0}}{R_{ij}}\right)\right) \tag{6.37}$$

The neurotransmitter in Equation (6.36) is given by the sum of the responses for the two input spikes, and $t_i^1$ is the solution of Equation (6.35) and represents the time of the first output spike. (An algorithm for calculating $F^{-1}(-)$ is given in Appendix D.) Equation (6.36) represents the worst case scenario in the sense that it uses the minimum expression for $U_{ij}(t)$ following two input spikes separated by $\Delta_j$. (If there were other previous input spikes $U_{ij}(t)$ would be greater.) Similarly, using $t_i^1$ minimizes the refractory recovery time.

3. When the input consists of a spike train with $P^1 = \Delta_j$, there must be a oscillatory solution so that the neuron is able to produce one output spike for each input spike; i.e., there must exist some value of $t$, $(0 < t < \Delta_j)$, such that:

$$\tanh^2\left(\frac{\Delta_j}{\tau_i}\right)\cdot\left\{R_{i0} + R_{ij}\tanh\left[\frac{T_{ij}}{R_{ij}\cdot\kappa\left(\frac{\Delta}{\tau_{ij}}\right)}\cdot F\left(\frac{t}{\tau_{ij}} + \psi\left(\frac{\Delta}{\tau_{ij}}\right)\right)\right]\right\} \geq \Theta_i \tag{6.38}$$

Here, $U_{ij}(t)$ is the result of a periodic input spike train (see Equation (4.23)), and $t = 0$ corresponds to the arrival time of the last input spike, ($K^{\text{th}}$ input spike). Since the solution is assumed to be oscillatory, with one output spike per input spike, the refractory term is set as if the neuron last produced an output spike at the same relative phase to the previous input spike, i.e., $t - t_i^{K-1} = \Delta_j$.

4. The neuron can only produce one output spike for each input spike. The worst case scenario for this constraint again occurs with a periodic spike train input, which maximizes $U_{ij}(t)$. Thus, after a neuron produces an output spike, which corresponds to the last input spike, no more spikes should be produced if the input spikes cease; i.e., for all $t$, $(t_i^K < t)$:

$$\tanh^2\left(\frac{t - t_i^K}{\tau_i}\right)\cdot\left\{R_{i0} + R_{ij}\tanh\left[\frac{T_{ij}}{R_{ij}\cdot\kappa\left(\frac{\Delta}{\tau_{ij}}\right)}\cdot F\left(\frac{t}{\tau_{ij}} + \psi\left(\frac{\Delta}{\tau_{ij}}\right)\right)\right]\right\} < \Theta_i \tag{6.39}$$

where:

$$t_i^K \equiv \tau_{ij}\cdot F^{-1}\left[\left(\frac{R_{ij}\cdot\kappa\left(\frac{\Delta}{\tau_{ij}}\right)}{T_{ij}}\right)\cdot \operatorname{Atanh}\left(\frac{\Theta_i}{R_{ij}\tanh^2\left(\frac{\Delta_j}{\tau_i}\right)} - \frac{R_{i0}}{R_{ij}}\right)\right] - \tau_{ij}\cdot\psi\left(\frac{\Delta}{\tau_{ij}}\right) \tag{6.40}$$

Notice that $t_i^K$ is the solution to Equation (6.38), and represents the time of the last output spike generated after the last input spike. Also, $t_i^K$ represents the minimum possible lag time between an input and output spike.

Figures 6.12-6.13 illustrate these four constraints for one set of parameters, and Figure 6.14 shows the potential voltages for a neuron satisfying them. Figure 6.15 shows the output spikes for an example input spike train. Notice that while this neuron is referred to as an identity neuron, it really functions as a time filter,[5] with the strong stimuli having short delay times. While even the weakest stimuli elicit a spike response in this identity neuron, it could have easily been designed to filter out

[5]Usually filters are thought of as operating in the frequency domain, where they take an input waveshape and modify the frequency spectrum to produce an output waveshape; e.g., a high-pass filter attenuates the low frequency components of a signal, while its high frequency components pass through. Here, a "time filter" operates on the input signal's magnitude (encoded in frequency), with weak input signals being delayed and strong signals passing through with little or no delay. If the "identity" neuron is designed so that two or more closely spaced input spike are required to produced an output spike, then weak signals are also attenuated, and the neuron acts as a high-pass filter. Notice also that if the time between input spike is less than $\Delta_j$, then the neuron can produce more output spikes than input spikes. In this case, high frequency signals are amplified.

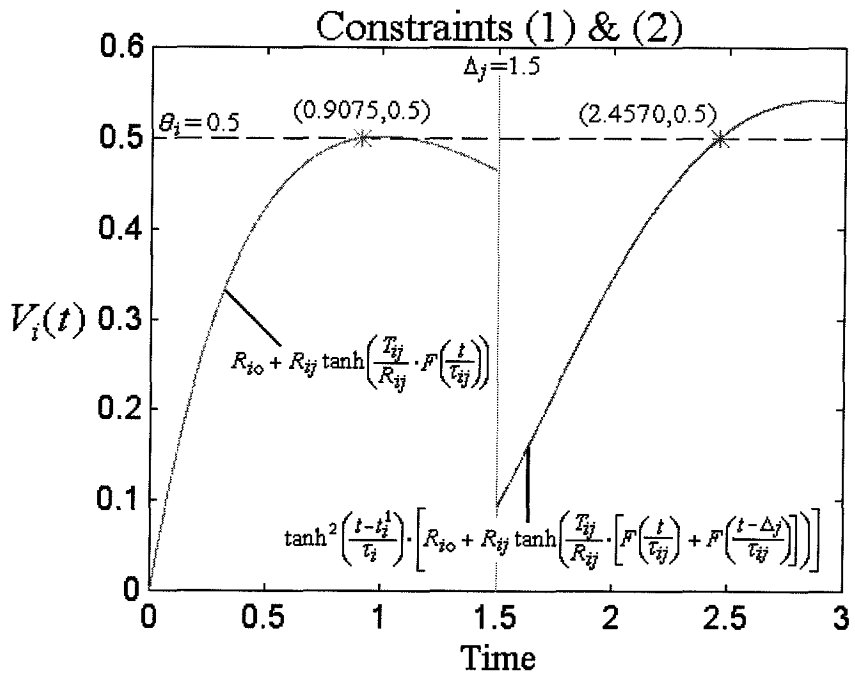


Figure 6.12: **Plot of Identity Constraints (1) & (2).** The plot shows Equation (6.35) for $0 < t < \Delta_j$, and Equation (6.36) for $\Delta_j \leq t < 2\Delta_j$. For the neuron being modeled, the parameters are: $\Theta_i = 0.5$; $R_{i0} = 0.0$; $R_{ij} = 1.0$; $T_{ij} = 1.5$; $\tau_{ij} = 1.0$; and $\tau_i = 1.25$. Also, $\Delta_j = 1.5$. Notice that both constraints are satisfied.

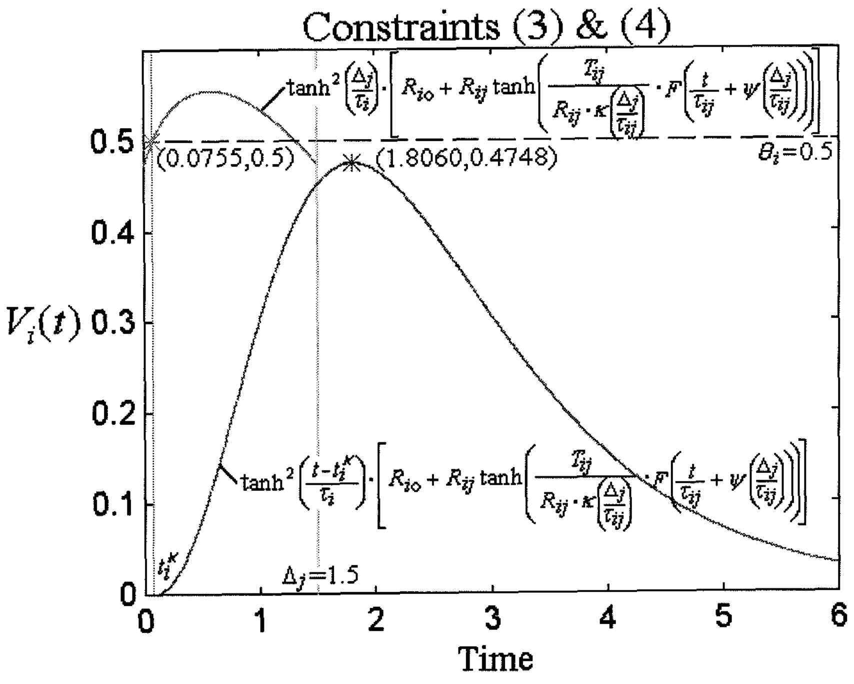


Figure 6.13: **Plot of Identity Constraints (3) & (4).** The plot shows Equation (6.38) for $0 < t < \Delta_j$, and Equation (6.39) for $t_i^K < t$. ($t_i^K$ is determined by the solution for $t$ in the third constraint – see Equation (6.40).) For the neuron being modeled, the parameters are: $\Theta_i = 0.5$; $R_{i0} = 0.0$; $R_{ij} = 1.0$; $T_{ij} = 1.5$; $\tau_{ij} = 1.0$; and $\tau_i = 1.25$. Also, $\Delta_j = 1.5$. Notice that both constraints are satisfied.

signals below a certain magnitude, by modifying the first and second constraints.

When an identity neuron has more than one input synapse, its output closely resembles the sum of all input spike trains; however, when there is more than one input providing a strong stimulus or there are closely correlated spikes between several inputs, the neuron is unable to recover from its own refractory period to produce an output spike for each input spike. In general, when the constraint equations are satisfied, the identity neuron will not produce more output spikes than input spikes, but may produce fewer spikes when there is "overlap" between them.

### 6.5.3 Synchronized Identity Neurons

The synchronized identity neuron is very similar to its unsynchronized counterpart. They both produce an output spike for an input spike, but with the synchronized identity neuron, the output spikes occur at regular intervals. Synchronized neurons use an additional input signal from a beating neuron, called the pace neuron. It provides a strong input signal that causes the identity neuron to *almost* reach threshold and output spikes at a regular frequency. At these times when the neuron is almost able to spike, it is described as having a "pseudospike" output. By using small values for the weights associated with the other inputs, they appear as perturbations of the strong stimulus from the pace neuron. The parameters of the identity neuron are chosen so that an input spike is able to perturb the neuron's potential enough to cause it to reach threshold and fire. Of course, the output spike occurs near the time of a pseudospike, causing the identity neuron's output to be synchronized with the pace neuron. The output spike train is analogous to a clocked bit pattern, with each spike representing a "1" and each non-spike representing a "0". Figure 6.16 shows a diagram of a synchronized identity neuron. If the same pace neuron is used to drive several identity neurons receiving different stimuli, then the output spikes associated with each signal are in phase.

When designing a synchronized identity neuron, it is useful to first consider the input from the pace neuron. (All of the terms associated with the pace neuron have

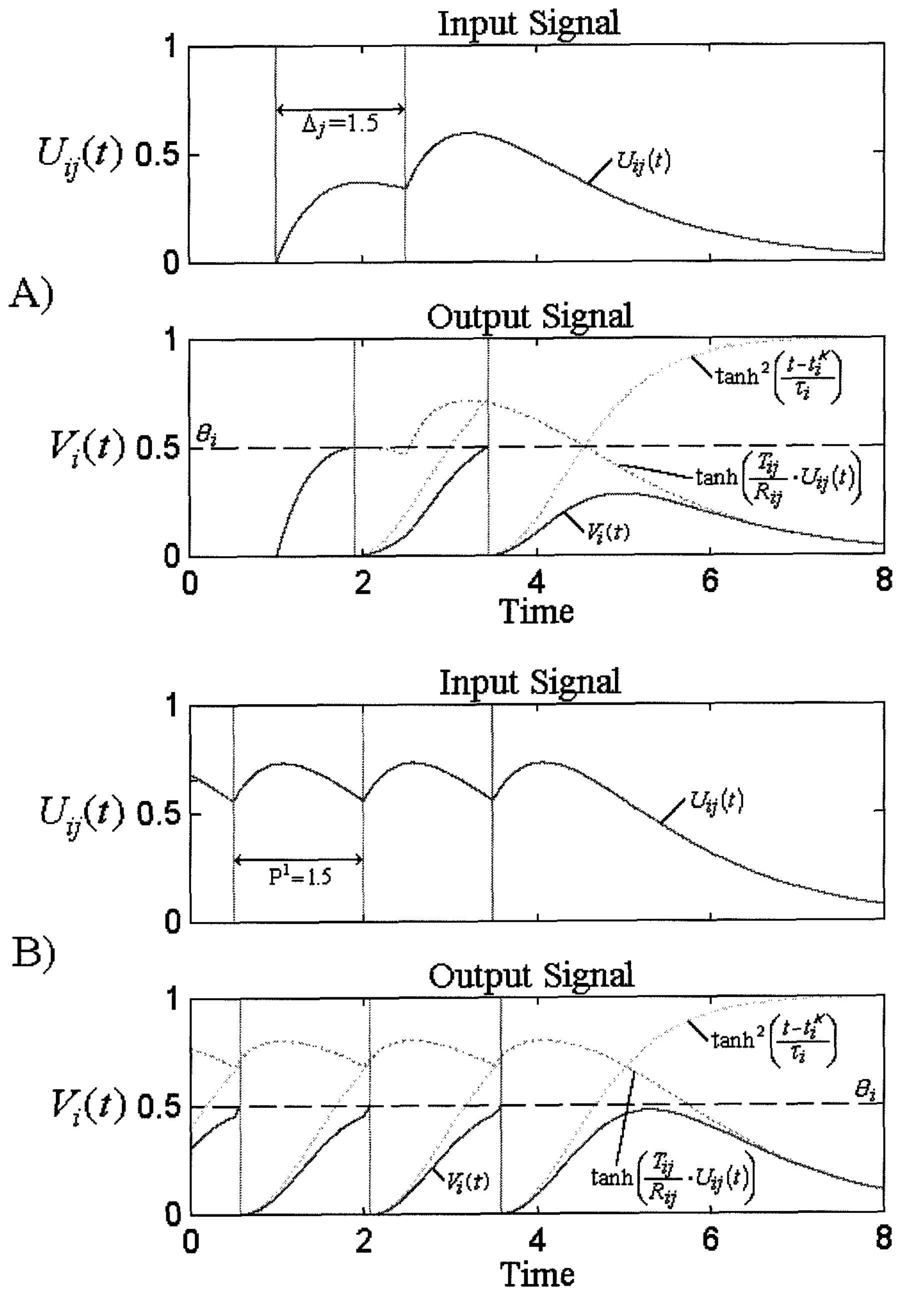


Figure 6.14: **Plot of Identity Neuron's Potentials.** The plot shows $U_{ij}(t)$ and $V_i(t)$ for a neuron satisfying the constraints (same parameters as in Figures 6.12-6.13). In (A), the first two constraints imply that the neuron should produce two output spikes for two isolated input spikes separated by $\Delta_j$. The input spikes occur at $t = 1$ and $t = 2.5$, while the output spikes occur at $t = 1.9075$ and $t = 3.4570$. In (B), the third and fourth constraints imply that the neuron should produce only one output spike for each input spike within a spike train of $P^1 = \Delta_j$, and when the spike train stops, no more spikes should be produced. During the spike train, the output spikes lag the input spikes by only 0.0755.

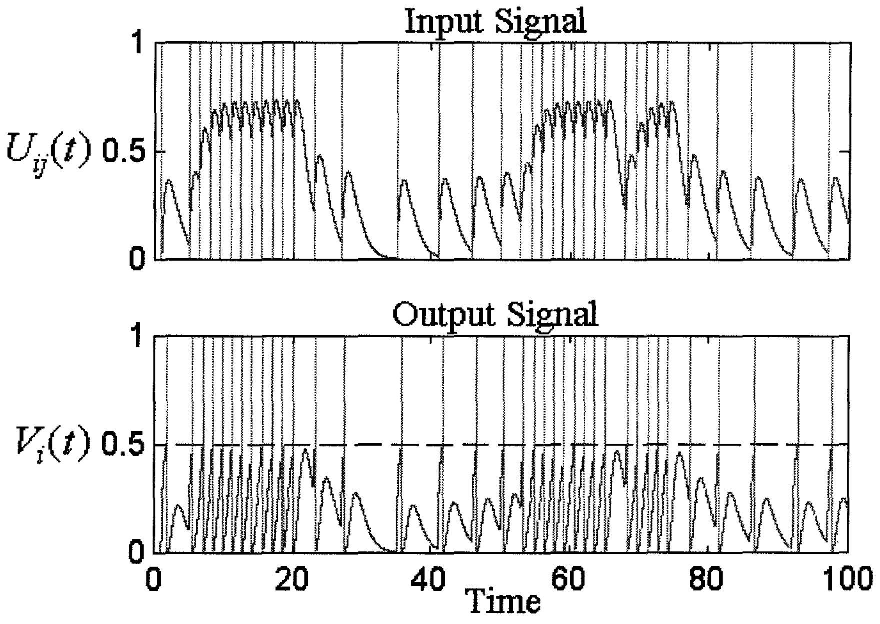


Figure 6.15: **Unsynchronized Identity Neuron's Response to Input Spikes.** The plot shows the input and output spikes as well as $U_{ij}(t)$ and $V_i(t)$ for an identity neuron satisfying the constraints (same parameters as in Figures 6.12-6.13). In the sample input spike pattern, all of the spikes are separated by at least $\Delta_j = 1.5$. Notice that each input spike results in an output spike, with the delay between them dependent upon the previous input activity. The maximum delay occured after the first input spike and was 0.9075, while the minimum delay occured after the 12th input spike and was 0.085.

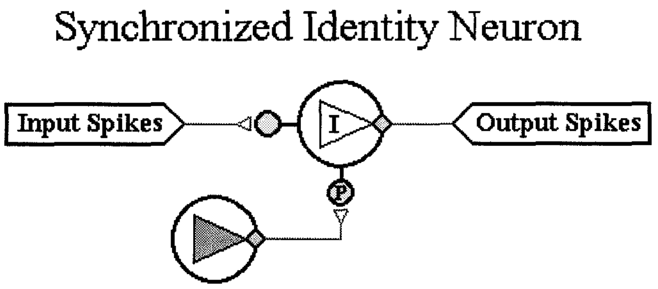


Figure 6.16: **Synchronized Identity Neuron.** The neuron has an input connection from a pace neuron, which beats at a regular frequency. The identity neuron shown here has only one other input signal, although more may be used. For each spike from the input signal, the neuron produces an output spike at the time of the next spike from the pace neuron. To indicate a synchronized identity neuron, "I" is placed inside the neuron symbol, and "P" is placed on the excitatory synapse from the pace neuron.

a "$p$" subscript. Also, the delay term for the pace neuron is inconsequential, and is considered to be zero, i.e., $d_{ip} = 0$.) The output spike period of the pace neuron, $\Delta_p$, sets the maximum oscillation frequency of the identity neuron. If $\Delta_p$ is less than the minimum allowed time between the spikes of the input signal, $\Delta_j$, then there can never be more than one input spike between pseudospikes, and the parameters can be chosen so that there will be a one-to-one correlation between the input and output spikes. But if $\Delta_p$ is greater than $\Delta_j$, then there may be more than one input spike between pseudospikes, and there will usually be fewer output spikes than input spikes. Essentially $\Delta_p$ acts as a time window for averaging the input signals, and when there are one or more input spikes within the window, an output spike is produced. While there is some loss of information with larger time windows, it is usually necessary to use values of $\Delta_p$ greater than $\Delta_j$, when several synchronized identity neurons are being used to phase lock input stimuli.[6] Since the pace neuron is beating, its output neurotransmitter is given by Equation (4.23), and the SNM reduces to:

$$V_i(t) = \tanh^2\left(\frac{t - t_i^K}{\tau_i}\right) \cdot \left\{ V_{ip}(t) + \sum_{j=1}^{N_i} R_{ij} \cdot \tanh\left(\frac{T_{ij}}{R_{ij}} \cdot U_{ij}(t)\right) \right\} \tag{6.41}$$

$$\text{where:} \quad V_{ip}(t) = R_{i0} + R_{ip} \tanh\left[\left(\frac{T_{ip}}{R_{ip} \cdot \kappa\left(\frac{\Delta_p}{\tau_{ip}}\right)}\right) \cdot f\left(\frac{t - t_p^K}{\tau_{ip}} + \psi\left(\frac{\Delta_p}{\tau_{ip}}\right)\right)\right] \tag{6.42}$$

Here, $V_{ip}(t)$ represents the resting potential, $R_{i0}$, plus the potential voltage contribution from the pace neuron, and $N_i$ represents the number of input synapses (not counting the input synapse from the pacing neuron).

To prevent the identity neuron from firing without an input signal, the parameters must be chosen so that the neuron is unable to reach threshold with just the pace signal input. Using the maximum value for the neurotransmitter from the pace neuron

[6]As an example, consider two input spike trains with spikes occurring at: $t_1 = \{0.5, 6.5, 12.5, 18.5\}$ & $t_2 = \{4.5, 9.5, 14.5, 19.5\}$. If the pace neuron had $\Delta_p = 1$ and created pseudospikes at: $t_p = \{0, 1, 2, \ldots\}$, then the first spike train would produce output spikes at: $t_{i1} = \{1, 7, 13, 19\}$, and the second spike train would produce output spikes at: $t_{i2} = \{5, 10, 15, 20\}$. But if the pace neuron had $\Delta_p = 5$ and created pseudospikes at: $t_p = \{0, 5, 10, \ldots\}$, then the output spikes would be synchronized and occur at: $t_{i1} = t_{i2} = \{5, 10, 15, 20\}$. Thus, larger windows allow for better synchronization.

given in Equation (4.25), this constraint becomes:

$$\max\{V_{ip}(t)\} = R_{i0} + R_{ip}\tanh\left(\frac{T_{ip}\mathrm{e}^{-1}}{R_{ip}\cdot\kappa\left(\frac{\Delta_p}{\tau_{ip}}\right)}\right) < \Theta_i \tag{6.43}$$

While this constraint needs to be satisfied, the LHS should be close to $\Theta_i$, to ensure that the neuron is in a state of *almost* firing (generating pseudospikes). Also, the time constant associated with the input signal from the pace neuron, $\tau_{ip}$, should be very small. Larger values for $\tau_{ip}$ cause the identity neuron to be near threshold over a longer time period, which allows the actual output spikes to lead the pseudospikes.

Associated with each pseudospike is a time window when it is sensitive to input spikes. Spikes occurring during this period cause the neuron to generate an output spike at the time of the pseudospike. To ensure that no input spikes are lost and that one input spike can not generate more than one output spike, the width of the window must equal the time between pseudospikes, $\Delta_p$, (see Figure 6.17). Thus, for an isolated input spike, there must exist an interval of length $\Delta_p$ such that its neurotransmitter contribution causes the neuron's potential to rise above threshold. By using the definition of $\psi(\Delta)$ implied by Figure C.1, this constraint simplifies to:

$$R_{ij}\tanh\left[\frac{T_{ij}}{R_{ij}}\cdot f\left(\psi\left(\frac{\Delta_p}{\tau_{ij}}\right)\right)\right] = \Theta_i - R_{i0} - R_{ip}\tanh\left(\frac{T_{ip}\mathrm{e}^{-1}}{R_{ip}\cdot\kappa\left(\frac{\Delta_p}{\tau_{ip}}\right)}\right) \tag{6.44}$$

If the spike is isolated, the neuron must not have fired recently; thus $\tanh^2\left(\frac{t-t_i^K}{\tau_i}\right) \approx 1$.

Now when this constraint is satisfied, there is both a minimum and maximum value for $\delta$ in Figure 6.17, which cause the corresponding pseudospike to become an output spike. The minimum value is given by:

$$\delta^{\min} \equiv \tau_{ij}\cdot\psi\left(\frac{\Delta_p}{\tau_{ij}}\right) \tag{6.45}$$

Of course, the maximum value is given by $\delta^{\max} = \Delta_p + \delta^{\min}$.

Figure 6.17 shows a pseudospike train, and the time window associated with one of the pseudospikes. Any isolated input spike arriving between $\delta^{\max}$ and $\delta^{\min}$ will

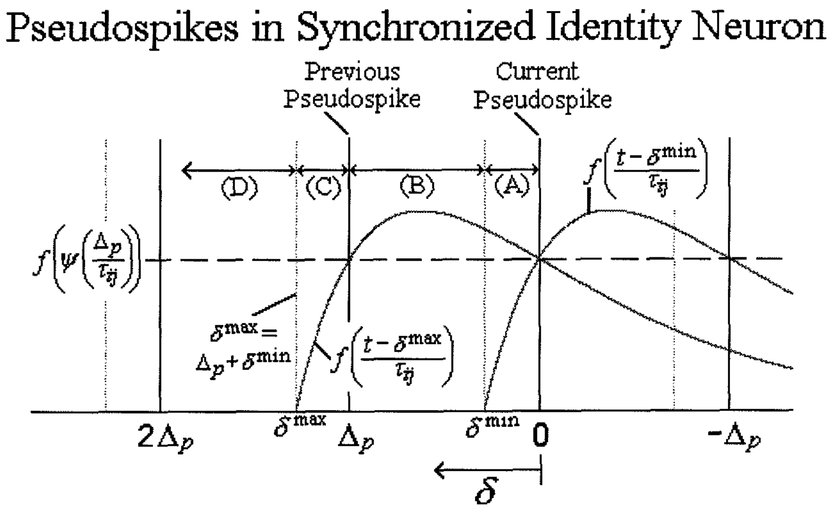

Pseudospikes in Synchronized Identity Neuron
Previous Pseudospike
Current Pseudospike
(D)
(C)
(B)
(A)
$f\left(\psi\left(\frac{\Delta_p}{\tau_{ij}}\right)\right)$
$f\left(\frac{t-\delta^{\min}}{\tau_{ij}}\right)$
$f\left(\frac{t-\delta^{\max}}{\tau_{ij}}\right)$
$\delta^{\max} = \Delta_p + \delta^{\min}$
$2\Delta_p$
$\delta^{\max}$
$\Delta_p$
$\delta^{\min}$
0
$-\Delta_p$
$\delta$

Figure 6.17: **Pseudospikes in Synchronized Identity Neuron.** The plot shows a train of pseudospikes separated by $\Delta_p$. Associated with each pseudospike, is a time window, $(\delta^{\min} \leq \delta < \delta^{\max})$, during which an isolated input spike will cause the neuron to be at or above threshold at the time of the pseudospike. Also shown is the neurotransmitter function of $f(-)$ for input spikes arriving at $\delta^{\max}$ and $\delta^{\min}$. At the time of the current pseudospike, both of these functions equal $f\left(\psi\left(\frac{\Delta_p}{\tau_{ij}}\right)\right)$, which is the minimum neurotransmitter contribution needed to reach threshold. Any input spike arriving between $\delta^{\max}$ and $\delta^{\min}$ will contribute more than enough neurotransmitter. All input spikes can be classified according to the region in which they arrive. The regions depicted above are used to describe the constraints for designing a synchronized identity neuron.

cause the pseudospike to produce an output spike. But what if the input spike is not isolated, and instead is one of many? When $\Delta_p > \Delta_j$, there *appears* to be two possible ways to design the identity neuron. The first method would have the identity neuron producing output spikes when an input spike fell within the pseudospike's time window, and if the number of output spikes lags behind the number of input spikes, then the neuron would continue producing output spikes until it "caught up" – even if no other spikes were being input. The second method would have the identity neuron producing output spikes only when an input spike fell within the corresponding time window, and if several input spikes occur within one window, then only one output spike is produced. The problem with the first method is that short duration, high frequency input stimuli can cause the neuron to fire constantly, with little or no correlation between the occurrence of the stimulation and the time of the output spikes. While the second method is ideal for synchronizing the timing between input stimuli, it loses information concerning the intensity of the input stimuli. In reality, both methods are physically unrealizable using the SNM.

For the first design option, where each input spike results in an output spike, the identity neuron requires an infinite memory. To see this, consider a very long input train of spikes separated by $\Delta_p$, which is also the spacing between the pseudospikes. Assume that an "extra" input spike occurs at the beginning of the spike train, between two of the regularly spaced spikes. Now, the identity neuron will produce an output spike at each of its pseudospikes during the spike train, but because of the extra input spike, it will be one output spike behind. As the spike train progresses, the effect of the extra spike diminishes and the neurotransmitter function converges to the solution for a periodic input, (see Equation (4.23)). Consequently, when the spike train stops, the response of the identity neuron becomes independent of the extra spike, and the necessary output spike is not produced. Thus, it is not possible to design a synchronized neuron which can produce an output spike for each input spike if the time between input spikes can be less than the time between output spikes.

For the second design option, where output spikes are only produced when an input spike occurs within the pseudospike's corresponding time window, the identity neuron needs to ignore the accumulation of neurotransmitter from multiple input spikes. To see this, consider the neurotransmitter released by an isolated input spike. Its functional form, $f(-)$, must have a sufficient time constant to allow the identity neuron to respond and produce an output spike, independent of when the input arrives within the pseudospike's time window, $(\delta^{\min} \leq \delta < \delta^{\max})$. When several input spikes arrive within the same time window, the total amount of neurotransmitter is equal to the sum of the individual contributions, but all of this neurotransmitter decays with same time constant as $f(-)$. By the time of the next pseudospike, the neurotransmitter can still be high enough to produce an output spike even if no input spikes arrive during that pseudospike's time window. Thus, it is not possible to design a neuron to respond to each individual input spike, and not produce multiple output spikes when an arbitrary number of input spikes occur within the same time window.

To accommodate these design limitations, the synchronized identity neuron is designed to obey seven constraints, which determine the neuron's behavior depending upon when an input spike arrives within a pseudospike's time window. The regions used to define the desired behavior are depicted in Figure 6.17. The constraints are:

1. A pseudospike can not produce an actual output spike by itself; an input spike is necessary. (This is the same as Equation (6.43).)

2. An isolated input spike must produce an output spike, and the window associated with each pseudospike must be exactly equal to the time between pseudospikes. The window is defined as the values for $\delta$, such that: $\delta^{\min} \leq \delta < \delta^{\max} = \Delta_p + \delta^{\min}$, where $\delta^{\min}$ is given by Equation (6.45). (This is the same as Equation (6.44).)

3. An isolated group of $n$ simultaneously arriving input spikes can not produce more than $n$ output spikes.[7] The total neurotransmitter contribution is given by

[7]Constraint (3) prevents the output from lagging too far behind the input. It does not prevent the total number of output spikes from exceeding the total number of input spikes. In fact, when a group of input spikes arrives with the time between each spike being close to $\Delta_p$, the identity neuron can sometimes produce an extra output spike at the end of the group, depending upon the phase difference between the input spikes and the pseudospikes.

$n$ times the contribution of each individual spike, i.e., $U_{ij}(t) = nf\left(\frac{t-t_o}{\tau_{ij}}\right)$. Now, the neuron can not produce more output spikes than input spikes if the total amount of neurotransmitter concentration does not remain over the required minimum level, $f\left(\psi\left(\frac{\Delta_p}{\tau_{ij}}\right)\right)$, for more than $(n+1)$ pseudospikes or $n\Delta_p$ time units. Therefore, this constraint becomes:

$$nf\left(\psi\left(\frac{n\Delta_p}{\tau_{ij}}\right)\right) < f\left(\psi\left(\frac{\Delta_p}{\tau_{ij}}\right)\right) \tag{6.46}$$

When this equation is satisfied for all integer values of $n \geq 2$, the neuron can not produce extra output spikes. This transcendental equation is satisfied for $n = 2$ when $\left(\frac{\Delta_p}{\tau_{ij}}\right) > 1.45127$, and for higher $n$ values, the ratio decreases. Thus, this constraint is satisfied if:

$$\tau_{ij} < 0.68905\Delta_p \tag{6.47}$$

4. When an input spike arrives in Region (A), $(0 \leq \delta < \delta^{\min})$, the neuron *may* produce an output spike at the time of the current pseudospike, depending upon the previous input activity. By constraint (6), if no output spike is produce, then the next pseudospike must produce one. (Since the response of the neuron is indeterminate for an input spike within this region, there is no equation associated with this constraint.)

5. When an input spike arrives in Region (B), $(\delta^{\min} \leq \delta \leq \Delta_p)$, the neuron *must* produce an output spike regardless of the neuron's previous input/output activity. When the input spike is isolated, this is automatically satisfied by constraint (2). If the input spike is not isolated, then there might be an output spike at the time of the previous pseudospike. Thus, the neuron's refractory

term must be taken into account,[8] and the worst case scenario is given by:

$$\begin{aligned} V_i(t) &= \tanh^2\left(\frac{\Delta_p}{\tau_i}\right)\cdot\Bigg\{\max\{V_{ip}(t)\} \\ &+ R_{ij}\cdot\tanh\left[\mathrm{e}^{-\frac{\Delta_p}{\tau_{ij}}}U^{\min}+\frac{T_{ij}}{R_{ij}}\cdot f\left(\frac{\delta^{\min}}{\tau_{ij}}\right)\right]\Bigg\} \geq \Theta_i \end{aligned} \tag{6.48}$$

$$\text{where:} \qquad U^{\min} = \mathrm{Atanh}\left(\frac{\Theta_i - \max\{V_{ip}(t)\}}{R_{ij}}\right) \tag{6.49}$$

Here, $U^{\min}$ represents the minimum amount of neurotransmitter needed to produce an output spike, and $\mathrm{e}^{-\frac{\Delta_p}{\tau_{ij}}}U^{\min}$ is a lower limit on the minimum possible amount of neurotransmitter remaining at the time of the current pseudospike. Also, when $\delta = \delta^{\min}$, the $f(-)$ is at its smallest value for all possible $\delta$ within Region (B).

6. When an input spike arrives in Region (C), $(\Delta_p \leq \delta < \delta^{\max})$, the neuron must produce an output spike if there was no output spike at the previous pseudospike. (In this region, the "previous pseudospike" actually occurs after the input spike.) If the input spike is isolated, then this constraint is satisfied by constraint (2). But if the input spike is not isolated, then there might have been an output spike at the pseudospike which occurred at $\delta = 2\Delta_p$, (2 pseudospikes before the current spike), and the worst case scenario is given by:

$$\begin{aligned} V_i(t) &= \tanh^2\left(\frac{2\Delta_p}{\tau_i}\right)\cdot\Bigg\{\max\{V_{ip}(t)\} \\ &+ R_{ij}\cdot\tanh\left(\mathrm{e}^{-\frac{2\Delta_p}{\tau_{ij}}}U^{\min}+\frac{T_{ij}}{R_{ij}}\cdot f\left(\frac{\delta^{\max}}{\tau_{ij}}\right)\right)\Bigg\} \geq \Theta_i \end{aligned} \tag{6.50}$$

Notice $\mathrm{e}^{-\frac{2\Delta_p}{\tau_{ij}}}U^{\min}$ is a lower limit on the minimum possible amount of neu-

[8]Since an output spike occurs as soon as the neuron reaches threshold, in general it will not exactly coincide with the maximum of the neurotransmitter from the pace neuron, which defines the time of the pseudospike. When the parameters are set to satisfy all of the constraint equations, the output spikes slightly lead the pseudospikes. Since this lead time actually *increases* the value of the refractory term, $\tanh^2\left(\frac{\Delta_p}{\tau_i}\right)$ represents the worst case.

rotransmitter remaining if an output spike occurs 2 pseudospikes before the current spike. And when $\delta = \delta^{\max}$, the function $f(-)$ is at its smallest value for all possible $\delta$ within Region (C).

7. When an isolated input spike arrives in Region (D), ($\delta^{\max} \leq \delta$), the neuron can not produce an output spike at the current pseudospike if there are no input spikes in other regions. By constraint (2), the neuron would have already produced an output spike, and since the input spike did not arrive during the current pseudospike's time window, it should not produce an output spike. (No additional equation is required.)

Now, when a synchronized identity neuron is designed to satisfy all of these constraints, it will always output a spike when it receives a group of input spikes. If the spikes are isolated or widely separated in time, then there will be a one-to-one correspondence between the input and output spikes; however, if the input spikes are closely spaced, then the neuron may still be producing output spikes several pseudospikes later, but the total number of output spikes tends to be less than or equal to the number of input spikes. (An extra spike can occur when the spacing between the input spikes is close to $\Delta_p$.) When a synchronized identity neuron has more than one input synapse, its output closely resembles what it would be if all of the input spikes arrived at the same synapse. But the separate synapses increase the likelihood of the neuron producing more output spikes than input spikes.

Figures 6.18-6.20 show the input and output spikes as well as $U_{ij}(t)$ and $V_i(t)$ for three synchronized identity neurons using different values for $\Delta_p$. All three examples use the same input spike train, in which the spikes are separated by at least 0.25, ($\Delta_j = 0.25$). The differences in the input neurotransmitter concentration, $U_{ij}(t)$, are due to the different values of $\tau_{ij}$.[9] The total length of each test run was 10 units, with the first pseudospike occurring at $t = 0$. Notice that the neurons only produce output

[9]In addition to the differences in $\Delta_p$, the parameters of $\tau_{ij}$, $\tau_i$, and $T_{ip}$, were chosen as follows: $\tau_{ij} = 0.5\Delta_p$, $\tau_i = 0.2\Delta_p$, and $T_{ip} = \mathrm{e}R_{ip} \cdot \kappa \left(\frac{\Delta_p}{\tau_{ip}}\right) \cdot \mathrm{Atanh}\left(\frac{0.95\Theta_i - R_{i0}}{R_{ip}}\right)$. Notice that this choice for $T_{ip}$ sets the maximum potential reached by each pseudospike at $0.95\Theta_i$. Also, while not varying between neurons, the parameters of $R_{i0}$, $R_{ij}$, and $T_{ij}$ were chosen as follows: $R_{i0} = 0.0$, $R_{ij} = \frac{\Theta_i - \max\{V_{ip}(t)\}}{0.99}$,

spikes near the times of their pseudospikes; the output spikes are always leading the pseudospikes, but by never more than 0.001. The response of the neuron in Figure 6.18, with $\Delta_p = 0.2 < \Delta_j$, closely matches the input spike pattern. There are 21 input and output spikes, with a one-to-one correlation between them. The neuron in Figure 6.19, with $\Delta_p = 0.4 > \Delta_j$, does not maintain the same one-to-one correlation and produces only 18 output spikes, but the output spikes reflect the "character" of the input pattern, with gaps in the input spike train appearing in the output pattern. The response of the neuron in Figure 6.20, with $\Delta_p = 0.6 > 2\Delta_j$, appears to be independent of the input spiking pattern, with each of the 16 pseudospikes becoming an output spike. The problem with this identity neuron is that the long time between pseudospikes prevents the output spikes from reflecting the variations within the input pattern; however, the output spikes do correctly indicate the presence of input spikes. Thus, while the output spike pattern from a synchronized identity neuron with a small value for $\Delta_p$, closely reflects the input spikes, a larger value for $\Delta_p$ may be required to phase lock stimuli going into different identity neurons (see footnote on page 122).

Finally, it is sometimes necessary to estimate the small phase difference between the actual output spikes and the pseudospikes. In Equation (6.42), $V_{ip}(t)$ is at its maximum value when the argument of $f(-)$ is 1, which corresponds to the occurrence of a pseudospike. By letting $dt$ represent the lead time of the output spike, $V_{ip}(t)$ can be expressed as:

$$V_{ip}(dt) = R_{i0} + R_{ip}\tanh\left[\frac{T_{ip}}{R_{ip}\cdot\kappa\left(\frac{\Delta_p}{\tau_{ip}}\right)}\cdot f\left(1 - \frac{dt}{\tau_{ip}}\right)\right] \tag{6.51}$$

Since the tanh function is always less than one, it is apparent from Equation (6.41) that input spikes applied to each synapse can never cause a neuron's potential to increase by more than $R_{ij}$. Therefore, by setting $V_i(t)$ equal to $\Theta_i$ in Equation (6.41),

and $T_{ij} = \text{Atanh}\left[\frac{0.99}{f\left(\psi\left(\frac{\Delta_p}{\tau_{ij}}\right)\right)}\right]$.

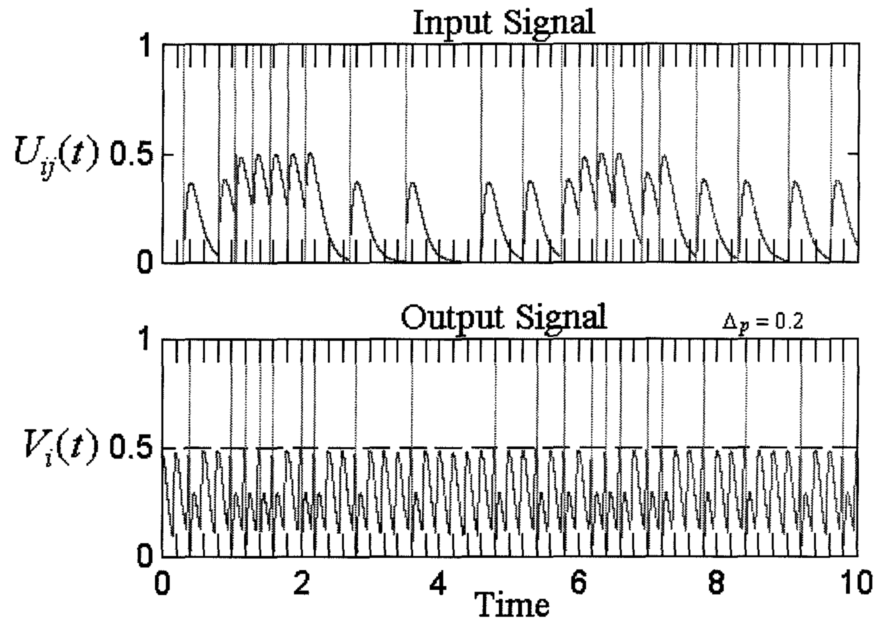


Figure 6.18: **Response of Synchronized Identity Neuron with $\Delta_p = 0.2$.** The plot shows the input and output spikes as well as $U_{ij}(t)$ and $V_i(t)$ for a synchronized identity neuron satisfying the design rules. For this neuron, the parameters are: $R_{i0} = 0.0$, $R_{ip} = 1.0$, $T_{ip} = 1.2792$, $\tau_{ip} = 0.05$, $R_{ij} = 0.02525$, $T_{ij} = 0.29198$, $\tau_{ij} = 0.1$, $\tau_i = 0.04$, $\Theta_i = 0.5$. (The hash marks indicate the times of the pseudospikes.)

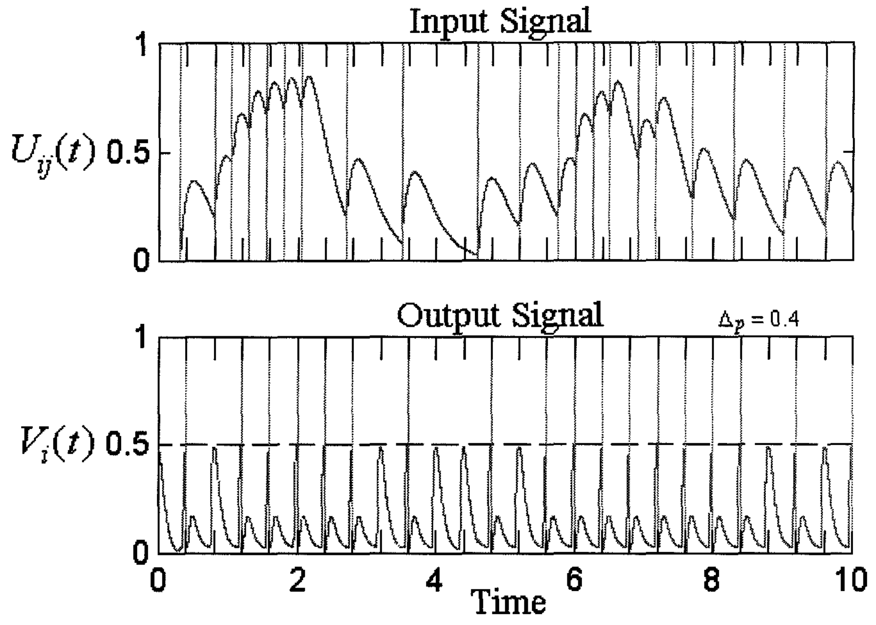


Figure 6.19: **Response of Synchronized Identity Neuron with $\Delta_p = 0.4$.** The plot shows the input and output spikes as well as $U_{ij}(t)$ and $V_i(t)$ for a synchronized identity neuron satisfying the design rules. For this neuron, the parameters are: $R_{i0} = 0.0$, $R_{ip} = 1.0$, $T_{ip} = 1.3998$, $\tau_{ip} = 0.05$, $R_{ij} = 0.02525$, $T_{ij} = 0.29198$, $\tau_{ij} = 0.2$, $\tau_i = 0.08$, $\Theta_i = 0.5$. (The hash marks indicate the times of the pseudospikes.)

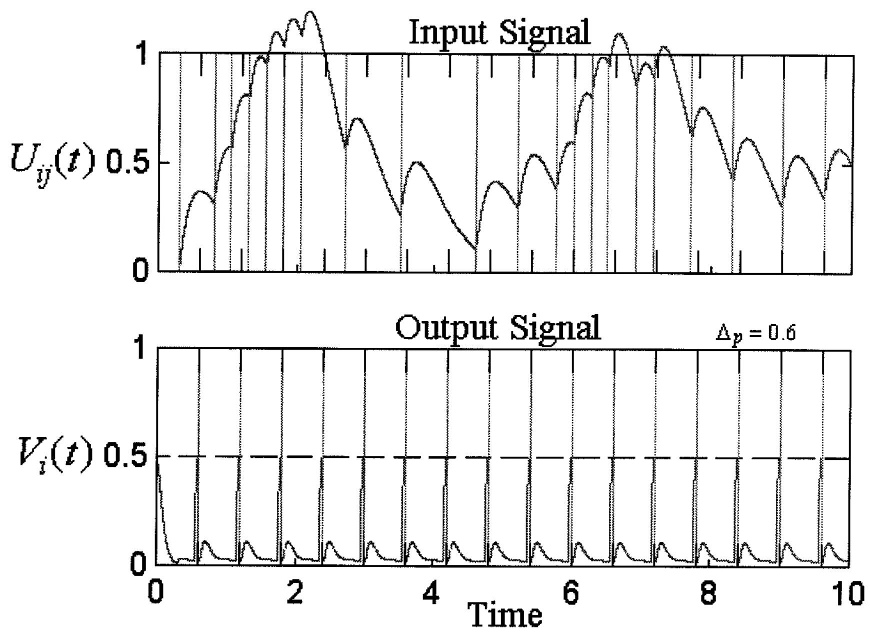


Figure 6.20: **Response of Synchronized Identity Neuron with $\Delta_p = 0.6$.** The plot shows the input and output spikes as well as $U_{ij}(t)$ and $V_i(t)$ for a synchronized identity neuron satisfying the design rules. For this neuron, the parameters are: $R_{i0} = 0.0$, $R_{ip} = 1.0$, $T_{ip} = 1.4039$, $\tau_{ip} = 0.05$, $R_{ij} = 0.02525$, $T_{ij} = 0.29198$, $\tau_{ij} = 0.3$, $\tau_i = 0.12$, $\Theta_i = 0.5$. (The hash marks indicate the times of the pseudospikes.)

the upper bound on the lead time of an output spike is found to be:

$$dt < \tau_{ip} - \tau_{ip} \cdot F^{-1}\left[\frac{R_{ip} \cdot \kappa\left(\frac{\Delta_p}{\tau_{ip}}\right)}{T_{ip}} \cdot \operatorname{Atanh}\left(\frac{\Theta_i}{R_{ip}\tanh^2\left(\frac{\Delta_p}{\tau_i}\right)} - \left(\frac{1}{R_{ip}}\right)\sum_{j=0}^{N_i} R_{ij}\right)\right] \tag{6.52}$$

An output spike will always lead a pseudospikes by less than this upper limit. And for the identity neurons tested in Figures 6.18-6.20, the maximum lead time is bounded by 0.0014085. (Due to the way the parameters where chosen, this number is a constant for all three neurons.) If the phase difference is defined as $\frac{dt}{\Delta_p}$, then the maximum phase differences for the three neurons tested are less than 0.00704, 0.00352, and 0.00235, respectively.

## 6.5.4 Synchronized Inverse Neurons

The synchronized inverse neuron operates the same as the synchronized identity neuron, except that it outputs spikes at opposite times; i.e., during the pseudospike

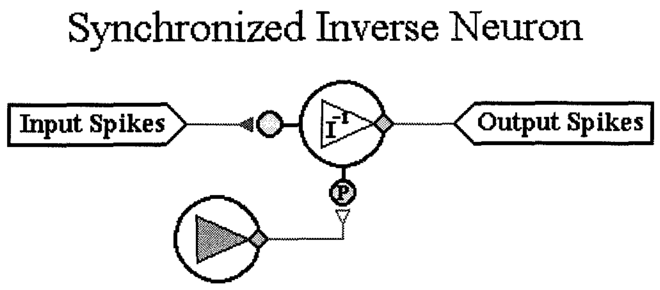


Figure 6.21: **Synchronized Inverse Neuron.** The neuron has an input connection from a pace neuron, which beats at a regular frequency. The inverse neuron shown here has only one other input signal, although more may be used. When there are no input spikes from the signal, the neuron produces an output spike for each spike from the pace neuron. But when a spike is input from the signal, it inhibits the neuron and prevents it from producing a spike at the time of the next spike from the pace neuron. To indicate a synchronized inverse neuron, "$\mathrm{I}^{-1}$" is placed inside the neuron symbol, and "P" is placed on the excitatory synapse from the pace neuron.

events in which the synchronized identity neuron produces an output spike, the synchronized inverse neuron does not, and vice versa. The synchronized identity neuron uses a pace neuron to create pseudospikes, and input spikes have an excitatory effect, which causes the neuron to produce output spikes near the times of the pseudospikes. For the synchronized inverse neuron, the pace neuron's input is sufficient for producing actual output spikes, and input spikes have an inhibitory effect, which causes the neuron to not produce output spikes during the "pseudospikes." Figure 6.21 shows a diagram of a synchronized identity neuron.

While it is possible to design an inverse neuron from scratch using rules similar to those for the identity neuron, since the inverse neuron is usually used in conjunction with an identity neuron, those parameters can be slightly modified to generate an inverse neuron. The remainder of this section discusses one simple method for modifying an identity neuron to produce an inverse neuron. Both neurons are assumed to be receiving the same pace signal, ($\Delta_p$), and employ the same time constants ($\tau_i$, $\tau_{ip}$, and $\tau_{ij}$). To distinguish between the two neurons, $\hat{\imath}$ (instead of $i$) is used for the subscripts on the inverse neuron's parameters.

For the identity neuron, the maximum of $V_{ip}(t)$ is slightly below threshold, $\Theta_i$, and sets the times of the pseudospikes. For the inverse neuron, the maximum of

$V_{\hat{i}p}(t)$ needs to be above threshold by the same amount the identity neuron is below threshold; i.e., assuming $\Theta_{\hat{i}} = \Theta_i$:

$$\max\left\{V_{\hat{i}p}(t)\right\} - \Theta_i = \Theta_i - \max\left\{V_{ip}(t)\right\} > 0 \tag{6.53}$$

Here, the maximum values for $V_{ip}(t)$ and $V_{\hat{i}p}(t)$ are given by Equation (6.43). This condition can be satisfied by using the same values for the neuron's resting potential, $(R_{\hat{i}0} = R_{i0})$, and the synaptic receiver weight from the pace neuron, $(R_{\hat{i}p} = R_{ip})$, if $T_{\hat{i}p}$ is given by:

$$T_{\hat{i}p} = \mathrm{e}\cdot R_{ip}\cdot\kappa\left(\frac{\Delta_p}{\tau_{ip}}\right)\cdot\mathrm{Atanh}\left(\frac{2\left(\Theta_i - R_{i0}\right)}{R_{ip}} - \tanh\left(\frac{T_{ip}\mathrm{e}^{-1}}{R_{ip}\cdot\kappa\left(\frac{\Delta_p}{\tau_{ip}}\right)}\right)\right) \tag{6.54}$$

Since the inverse neuron is above threshold by the same amount that the identity neuron is below, if the neurotransmitter from an input spikes is enough to cause the identity neuron to reach threshold and fire, then the same amount of neurotransmitter must also be enough to prevent the inverse neuron from reaching threshold. Of course the input into the identity neuron must be excitatory while the input into the inverse neuron must be inhibitory, i.e., let $R_{\hat{i}j} = R_{ij}$ and $T_{\hat{i}j} = -T_{ij}$.

When the variables are set as described, the inverse neuron can not produce a spike if the identity neuron is firing; however, since both neurons still have a refractory term, $\tanh^2\left(\frac{t-t_i^K}{\tau_i}\right)$, that depends upon their own spiking history, it is possible that neither fires during a pseudospike. Thus, the input neurotransmitter may be sufficient to cause one of the neurons to fire, but the refractory term may be sufficiently low to inhibit it from reaching threshold. If the neuron's time constant is small, this problem is minimized. But for the identity neurons tested in Figures 6.18-6.20, the time constant was set at $\tau_i = 0.2\Delta_p$, which was slightly too large to prevent this problem. Figures 6.22-6.23 show the results for the corresponding inverse neurons. (The corresponding inverse neuron for the last test neuron is not shown, since the output spike train is trivial, with no output spikes except at $t = 0$.) When the neuron time constants were reduced to $\tau_i = 0.15\Delta_p$, the problem of neither neuron firing

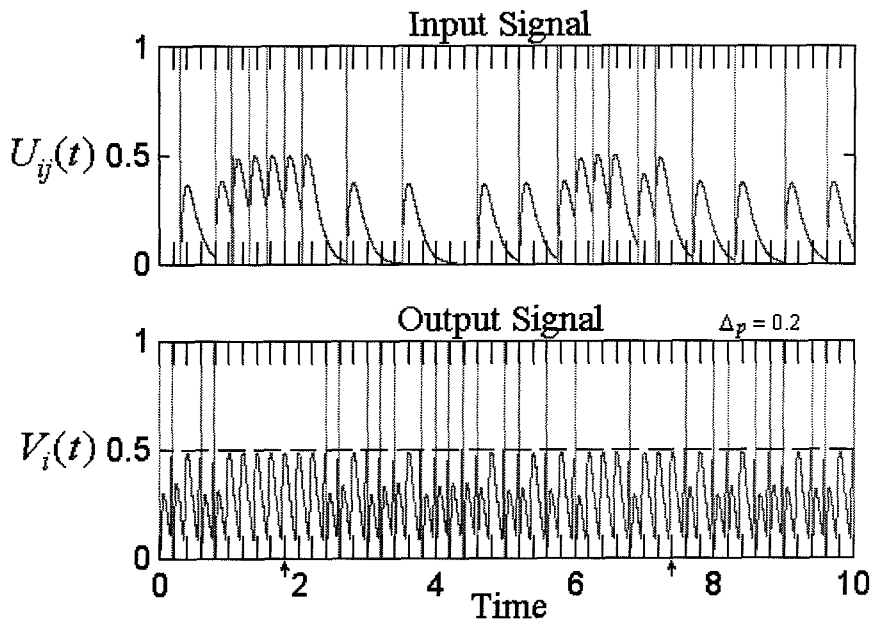


Figure 6.22: **Response of Synchronized Inverse Neuron with $\Delta_p = 0.2$.** The plot shows the input and output spikes as well as $U_{ij}(t)$ and $V_i(t)$ for the synchronized inverse neuron corresponding to the identity neuron shown in Figure 6.18. The parameters are: $R_{\hat{i}0} = 0.0$, $R_{\hat{i}p} = 1.0$, $T_{\hat{i}p} = 1.44439$, $\tau_{\hat{i}p} = 0.05$, $R_{\hat{i}j} = 0.02525$, $T_{\hat{i}j} = -0.29198$, $\tau_{\hat{i}j} = 0.1$, $\tau_{\hat{i}} = 0.04$, $\Theta_{\hat{i}} = 0.5$. (The hash marks indicate the times of the pseudospikes, and the two arrows below the time scale mark the pseudospikes when neither the identity nor the inverse neuron produced a spike.)

was eliminated for the test input spike train, (results not shown). Notice that it is impossible for both neurons to produce an output spike during the same pseudospike, since both can not have a sufficient amount of excitatory neurotransmitter available to produce a spike.

Like the identity neuron, the synchronized inverse neuron also has a lead time between its output spikes and the pseudospikes. But while the identity neuron's lead time increases with strong input stimuli, the inverse neuron's lead time is at its maximum with no input signal. (The input from the pace neuron causes it to fire prematurely.) Therefore, similar to Equation (6.52), the maximum lead time in the inverse neuron is bounded by:

$$dt < \tau_{\hat{i}p} - \tau_{\hat{i}p} \cdot F^{-1}\left[\frac{R_{\hat{i}p} \cdot \kappa\left(\frac{\Delta_p}{\tau_{\hat{i}p}}\right)}{T_{\hat{i}p}} \cdot \operatorname{Atanh}\left(\frac{\Theta_{\hat{i}}}{R_{\hat{i}p} \tanh^2\left(\frac{\Delta_p}{\tau_{\hat{i}}}\right)}\right)\right] \qquad (6.55)$$

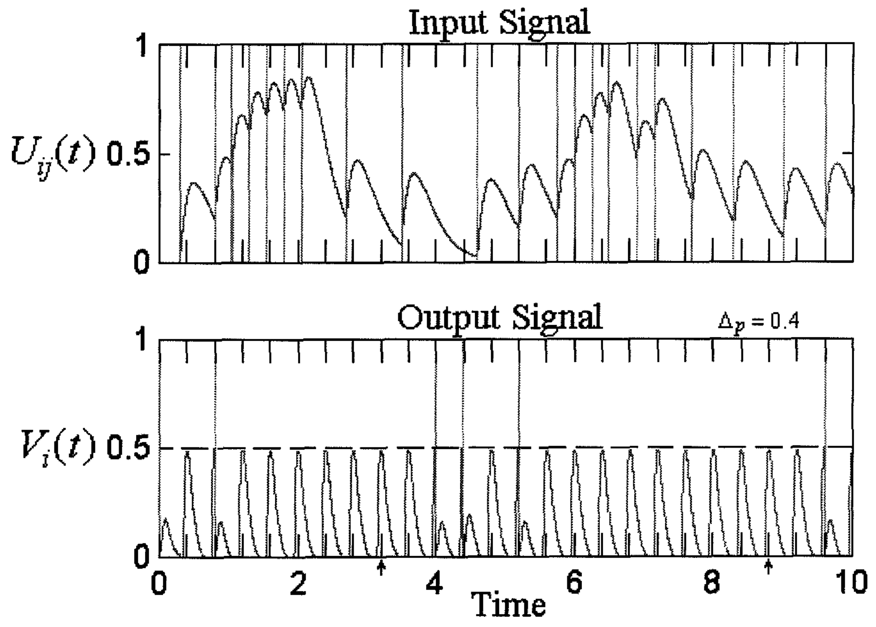


Figure 6.23: **Response of Synchronized Inverse Neuron with $\Delta_p = 0.4$.** The plot shows the input and output spikes as well as $U_{ij}(t)$ and $V_i(t)$ for the synchronized inverse neuron corresponding to the identity neuron shown in Figure 6.19. The parameters are: $R_{\hat{i}0} = 0.0$, $R_{\hat{i}p} = 1.0$, $T_{\hat{i}p} = 1.58057$, $\tau_{\hat{i}p} = 0.05$, $R_{\hat{i}j} = 0.02525$, $T_{\hat{i}j} = -0.29198$, $\tau_{\hat{i}j} = 0.2$, $\tau_{\hat{i}} = 0.08$, $\Theta_{\hat{i}} = 0.5$. (The hash marks indicate the times of the pseudospikes, and the two arrows below the time scale mark the pseudospikes when neither the identity nor the inverse neuron produced a spike.)

The output spikes always lead the pseudospikes by less than this upper bound. And for the inverse neurons tested in Figures 6.22-6.23, the maximum lead time, $dt$, is bounded by 0.015345. (Due to the way the parameters were chosen, this number is a constant for all of the inverse neurons.) This limit can be reduced by using a smaller difference between $\max\left\{V_{\hat{i}p}(t)\right\}$ and $\Theta_{\hat{i}}$ in Equation (6.53). If the phase difference is defined as $\frac{dt}{\Delta_p}$, then the maximum phase differences for the two neurons tested are less than 0.07673 and 0.03836, respectively, (0.02558 for the third inverse neuron not shown).

## 6.6 Summary of Special Neurons

This chapter demonstrated a few examples of some simple functions that individual neurons can perform. It is important to remember that these neurons were designed to deal with information that is noisy or imprecise and provide reasonable outputs, and they are not meant to be precise computational devices. While they usually perform their intended tasks well, their outputs may not be exactly as desired; e.g., the identity neurons were designed to produce an output spike for each input spike, but for some input spike patterns, extra output spikes can be produced.

The high gain neurons of Section 6.2 output spikes at a nearly constant frequency when the input signal exceeds the neuron's threshold level. They are particularly useful for detecting the presence of an input stimulus. Section 7.2 will show how high gain neurons may be combined with synchronized identity neurons to form a stimulus detector, which causes the spikes from several sensor neurons to become phase locked when a sufficiently strong stimulus is presented. Such phase locking is necessary for the pattern recognition networks which are discussed in Section 7.3.

The memory oscillators of Section 6.3 are useful for storing the state of an input signal. Large arrays of such neurons can be used as binary memory units, with each neuron retaining one bit of information. Section 7.6 will show how memory oscillators can be connected to create counting networks.

The bounded threshold neurons of Section 6.4 can be used to measure the strength of an input stimulus. Any stimulus outside of the target range will not evoke output spikes. Also, bounded threshold neurons can be used to signal the start and end of an input stimulus.

The identity and inverse neurons have a wide variety of uses. The unsynchronized identity neuron of Section 6.5.2 can be used to combine several input spike signals into a single output spike train. Several of the synchronized identity neurons described in Section 6.5.3 can be connected to the same pace neuron to phase lock different stimuli. The inverse neurons of Section 6.5.4 are useful for generating inverse spike trains. These neurons fire at opposite times of the identity neurons. Inverse signals are often needed to evaluate logic expressions, which are addressed in Section 7.4.

# Chapter 7 Computing with the SNM

## 7.1 Introduction

This chapter uses the neuron configurations presented in Chapter 6 as building blocks for larger networks, which are capable of accomplishing more complex tasks. Like the neurons from which they are constructed, these networks are not meant to be precise computational devices, but are intended to deal with noisy or imprecise information and provide reasonable outputs. Also, like the neurons in Chapter 6, the parameters are chosen a priori, without any learning. Networks which rely on learning are not discussed until Chapter 9.

While each of the neuron configurations in Chapter 6 was developed in great detail using a set of constraint equations, the networks in this chapter are not. Due to the complexity of the networks involved, a complete set of constraint equations becomes very cumbersome and unrevealing. Instead, each section begins with a schematic diagram of the network configuration, and then proceeds with a general description of the desired behavior for the component neurons. All of the sections include the parameter values for a working network and show a sample output.

The chapter begins by discussing a stimulus detector network (Section 7.2), which is used to phase lock (synchronize) different input spike trains. The network can be configured so that the phase locking only occurs if the total activity on all of the input lines is sufficiently strong, and when the total activity is weak, no signals pass through. Since many of the other networks require synchronized input signals, this network can be thought of as a preprocessor for them.

Section 7.3 presents several different type of networks for pattern recognition. A distinction is made between temporal, spatial, and spatial-temporal patterns. A temporal pattern consists of one input signal with a sequence of spikes. A spatial pattern uses several inputs, but with only one target event on each input line. (Events may be either spikes or non-spikes.) A spatial-temporal pattern uses several input signals with at least two target events on one or more of the input lines. The network configuration depends upon the type of pattern being recognized. Also, the neurons can be arranged in layers for more complex pattern recognition tasks.

Often the number of neurons in a multilayer pattern recognizing network can be reduced to a single logic neuron, which is described in Section 7.4. A logic neuron can be used to evaluate any Boolean logic expression. The expression must first be reduced to conjunctive normal form, which is a product (AND) of sums (OR). Essentially, the neuron uses synaptic clusters to evaluate each of the sums, and the neuron's potential performs the product operation. An output spike is produced only if all of the synaptic clusters are receiving spiking inputs.

Section 7.5 shows how a network with feedback connections can be used to store entire segments of an input spike train. It can be thought of as a short term memory, which makes no modifications to the network parameters; i.e., no learning. The spike pattern is preserved indefinitely, until it is replaced with a new memory.

A counting network can be used to determine the number of spikes within a spike train. Section 7.6 describes two different types of counters. The first is a linear counter, in which the number of neurons oscillating in the network represents the number of input spikes. The second is a sequential counter, in which only one neuron at a time oscillates, but each neuron in the network represents a specific number.

Section 7.7 presents a multiplexer network, which selects one of many input signals to be transferred to a single output. This network can be particularly useful for routing data within a complex nervous system. The section also discusses how a demultiplexer may be constructed from a multiplexer. A demultiplexer routes one input to one of several possible outputs.

While the pattern recognizing networks are useful for determining if an input spike train contains a specific spike sequence, it is sometimes necessary to determine if two or more spike trains are equal to each other. Section 7.8 describes a comparator network, which can be used for such a task. If the input spike trains are equivalent, the network outputs spikes, regardless of the actual input patterns.

And finally, Section 7.9 describes how a Hopfield-like associative memory network can be constructed from spiking neurons. The basic problem is to store a set of patterns in such a way that when the network is presented with a new pattern, it responds by producing whichever one of the stored patterns most closely resembles the input pattern. The stored patterns can be taken to be either a 0 or 1 value at each neuron in the network, where 1 indicates that the neuron is oscillating, while 0 indicates that it is not.

## 7.2 Stimulus Detectors

This section describes a stimulus detector, which uses a high gain neuron (Section 6.2) as the pace neuron for a group of synchronized identity neurons (Section 6.5.3). Each of the identity neurons receives a separate input signal. The network can be configured so that when a large stimulus is presented, the output spikes from the identity neurons become phase locked (synchronized). These spikes can then be used in networks which require synchronized inputs, e.g., the pattern recognition systems discussed in the next section.

Figure 7.1 shows a stimulus detector with three input signals. When a strong input signal arrives on one of the input lines, neurotransmitter is released into the excitatory synapse of the high gain pace neuron and the corresponding synchronized identity neuron. But the synchronized identity neuron can only produce an output spike when it receives an input spike from the pace neuron. Therefore, the input signal must also be sufficiently strong to cause the pace neuron to fire. The parameters of the high gain pace neuron may be chosen so that it only requires a single input spike before producing an output spike, or it may require a much stronger stimulus before firing.

When only one input spike is required for the high gain neuron to fire, any isolated input spike will appear at the corresponding output signal. However, if the high gain pace neuron requires multiple input spikes before it fires, weak signals (isolated spikes) are filtered out. Furthermore, the pace neuron's parameters may be set so

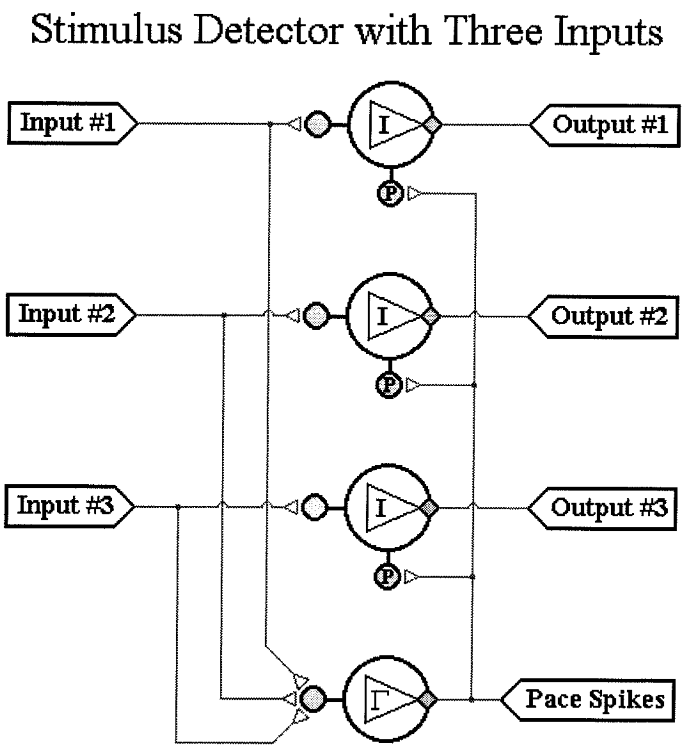


Figure 7.1: **Stimulus Detector.** The plot shows a stimulus detector with three input signals. Each input signal connects to a synchronized identity neuron and the excitatory synapse of a high gain neuron.

that it requires multiple spikes from several of the input signals before any output spikes are produced.

Once the neurotransmitter level at the excitatory synapse of the pace neuron is sufficient for producing output spikes, its high gain causes the output spikes to fire at a nearly constant frequency as long as the input stimulus is maintained. Thus, the output spikes from the identity neurons are produced with a fixed underlying frequency. Those identity neurons with no input spikes do not produce any output spikes, while those with strong input signals produce an output spike with every pseudospike generated from the pace neuron. (Notice that the stimulus detector may also include synchronized inverse neurons, which produce output spikes when the identity neurons are not; however, both types of synchronized neurons can only produce an output spike at the time of a pseudospike from the pace neuron.) Figure 7.2 shows a sample output for a stimulus detector.

While the stimulus detector can produce spikes with a fixed underlying frequency, the starting and stopping of such oscillations is determined by the input stimulus; hence, the output spikes are "unclocked," and the identity neurons are only synchronized with each other. The pattern recognizer networks and logic neurons discussed in the next sections only require that the timing between events remain constant, but the input patterns may start at any time. This is in contradistinction to conventional computers where a master clock drives the entire system.

Other variations on the stimulus detector shown in Figure 7.1 are possible. In one variation, the inputs into the high gain pace neuron form separate synapses. Since the synaptic parameters of a high gain neuron are chosen to cause the neurotransmitter to saturate the input synapse (see Section 6.2), each input signal can produce oscillations. Because the effect on the neuron's potential is independent for each input synapse, the actual output frequency depends upon the number of active inputs. In general, if the time between output spikes is $\Delta_1$ with one saturated input synapse, then the time between output spikes with $N$ saturate input synapses can be

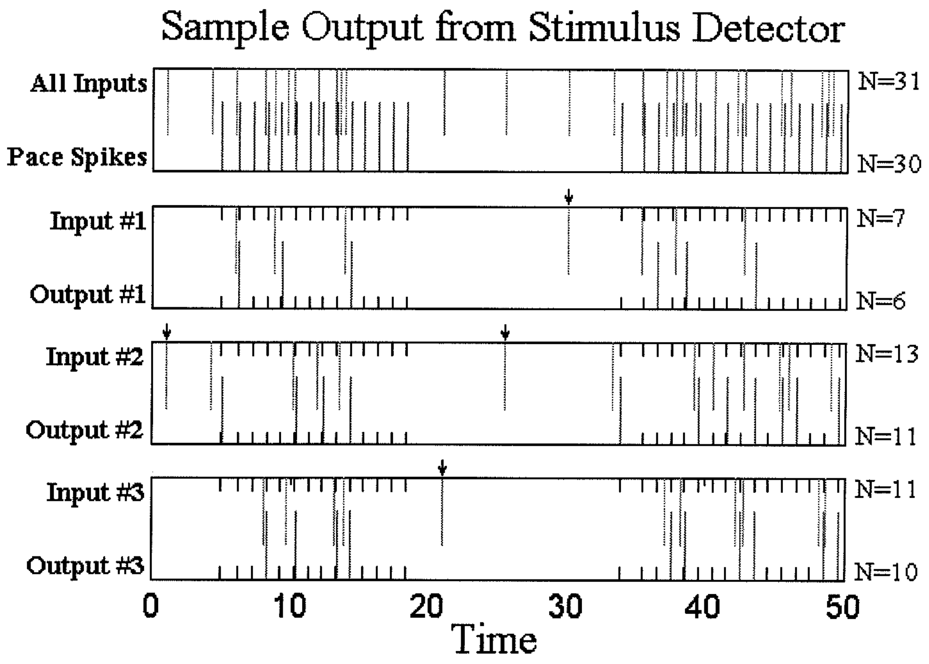


Figure 7.2: **Sample Output for Stimulus Detector.** The plot shows a set of random input spikes for the three input stimulus detector network shown in Figure 7.1. The top box shows the input and output spikes for the high gain pace neuron, while the next three boxes show the input and output spikes for the synchronized identity neurons. The hash marks correspond to the output spikes of the pace neuron. Notice that the identity neurons can only produce output spikes at the times of the pseudospikes. (The pseudospikes correspond to the maximums in neurotransmitter, which occur shortly *after* each pace spike. Since the hash marks represent the times of the pace spikes, many of the output spikes appear slightly after the hash marks.) Notice that when there are only a few widely separated input spikes, the pace neuron does not produce any pace spikes and these input spikes are filtered from the output signals. Arrows mark the input spikes for which the identity neurons did not produce corresponding output spikes. The values of $N$ to the right of each signal represent the number of spikes. The parameters for the high gain neuron were the same as those used in Figure 6.2, and the parameters for all of the identity neurons were: $R_{i0} = 0.0$, $R_{ip} = 1.0$, $T_{ip} = 1.404$, $\tau_{ip} = 0.05$, $R_{ij} = 0.02525$, $T_{ij} = 0.29198$, $\tau_{ij} = 0.5$, $\tau_i = 0.2$, $\Theta_i = 0.5$. (These parameters are the same as those used for the synchronized identity neurons of Figures 6.18-6.20, but with $\Delta_p = 1.0$. See footnote on page 129.)

approximated by:

$$\Delta_N = \text{Atanh}\left(\frac{\tanh(\Delta_1)}{\sqrt{N}}\right) \tag{7.1}$$

A second variation has the inputs into the pace neuron and the identity neurons coming from different sources. Thus, the pace neuron does not oscillate due to the actual input stimulus, but rather responds to a controlling stimulus. Alternatively, the pace neuron can simply be a beating neuron, which does not use any inputs. In this case, all input signals are synchronized, regardless of their strength.

In a third variation, the pace neuron's output signal is transmitted to each of the synchronized identity neurons with different delays. This configuration causes the synchronized identity neurons' output spikes to have a predetermined phase difference.

## 7.3 Pattern Recognition

### 7.3.1 Introduction

The SNM may be used to recognize patterns. Much of the artificial neural network research has been motivated by the desire to implement computational tasks that are difficult for conventional computers, but which are routinely performed by humans with relative ease. Perhaps the two problems where this difference in ability is most evident is in visual object recognition and speech recognition. Humans are able to easily identify objects in a visual scene, even if they are partially obstructed or at an unusual orientation. In speech recognition, humans are able to quickly recognize a wide variety of phonons and associate words and meanings to them. This recognition is largely independent of the individual characteristics of the speaker's voice. Even when some of the phonons or words are obscured by noise, the listener can usually interpret the speaker's intended meaning. Both of these problems fall into the category of pattern recognition, for which serial computers are less adept.

Since a neuron receives spiking inputs from other neurons or sensors, the problem of pattern recognition is defined as having the neuron respond to a specific input spiking pattern, referred to as the target pattern. Here, each of the spiking inputs is assumed to have a fixed underlying frequency, with the spike or non-spike events occurring at regular intervals. Thus, any bit pattern of 1's and 0's can be directly mapped into a sequence of spikes, where spike events represent 1's and non-spike events represent 0's. Since the input spikes have an underlying frequency, they may be assumed to be the output spikes from synchronized neurons, but each of the input signals does not need to have the same frequency.

All spiking patterns may be classified as one of three types: temporal, spatial, or spatial-temporal. A temporal pattern consists of one input signal with a sequence of spikes. A spatial pattern uses several inputs, but with only one target event on each input line. The events may be time delayed from each other. A spatial-temporal pattern is the most general type of pattern. It uses several input signals with at least two target events on one or more of the input lines. Examples of temporal, spatial, and spatial-temporal patterns are illustrated in Figures 7.3-7.5. (In this thesis, temporal patterns are written as row vectors, spatial patterns are written as column vectors, and spatial-temporal patterns are written as matrices with each row specifying one of the input lines.[1])

## 7.3.2 Recognizing Temporal Patterns

When attempting to recognize temporal patterns, the goal is to produce an output spike when one of the target patterns is presented on the single input line. Since the input is assumed to have a fixed frequency of events, the target patterns may be represented as row vectors of 1's and 0's.

A neuron designed to recognize a temporal spiking pattern uses $N$ separate input synapses branching out from the single input line, where $N$ is the number of target

[1]When vectors or matrices are written within text, commas will be used to separate elements in rows and semi-colons will be used to separate elements in columns. Thus, the matrix of:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}$$

can be written as: $[[a_{11}, a_{12}, a_{13}, a_{14}]; [a_{21}, a_{22}, a_{23}, a_{24}]; [a_{31}, a_{32}, a_{33}, a_{34}]]$. (This notation is consistent with that used in MATLAB.)

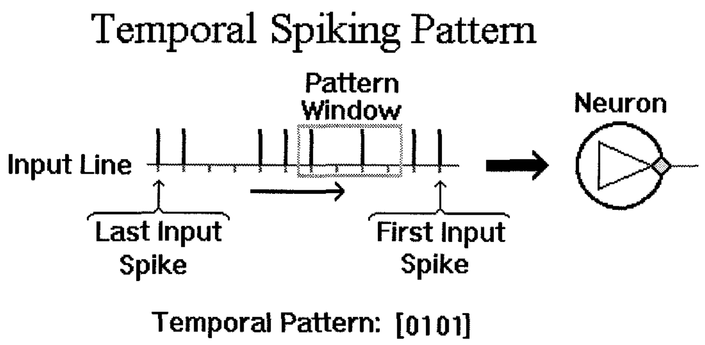


Figure 7.3: **Temporal Spiking Pattern.** The plot shows an input temporal spiking pattern with a fixed frequency of events. The spike on the right is the first to arrive at the neuron, with the four windowed events representing the pattern of $[0, 1, 0, 1]$.

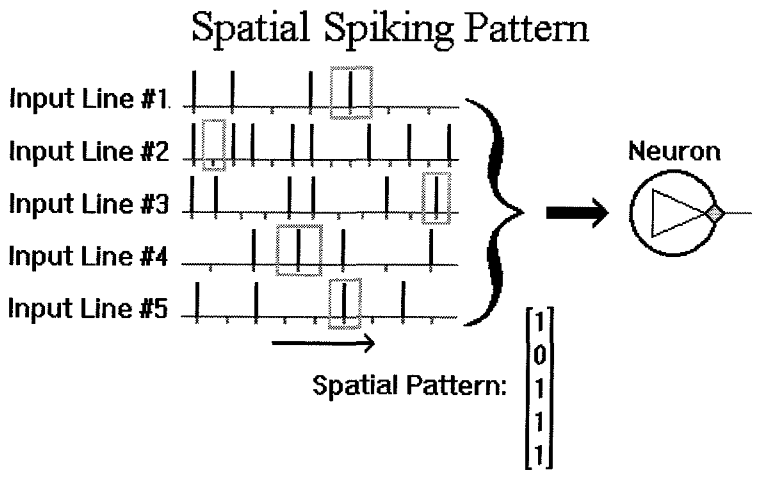


Figure 7.4: **Spatial Spiking Pattern.** The plot shows an input spatial spiking pattern with each input line having a fixed frequency of events. The spikes on the right are the first to arrive at the neuron, but each input line may have a different delay associated with it, allowing spikes generated at different times to form the input pattern. This is shown in the figure by the positions of the pattern windows on each of the input lines, with the windowed events representing the pattern of $[1; 0; 1; 1; 1]$. (Each pattern window may contain only one event.) Notice that the input signals need not have the same frequency, nor must they be phase locked.

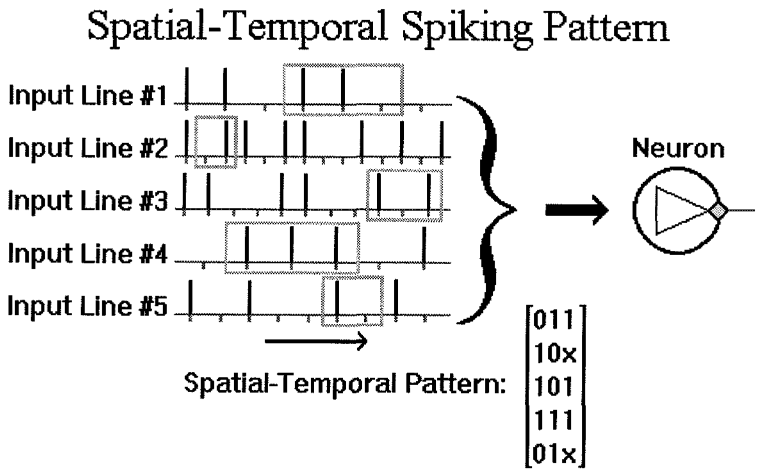


Figure 7.5: **Spatial-Temporal Spiking Pattern.** The plot shows an input spatial-temporal spiking pattern with each input line having a fixed frequency of events. The figure is the same as the spatial spiking pattern; however, the pattern windows may contain more than one event. Since the number of windowed events on each input line does not need to be the same, an "x" is used to fill the spatial-temporal matrix on rows representing inputs with less than the maximum number of windowed events. The windowed events represent the pattern of [[0, 1, 1]; [1, 0, x]; [1, 0, 1]; [1, 1, 1]; [0, 1, x]].

bits.[2] Each synapse is associated with one of the target bits and has a delay of $(N - n + 1)$ time units, where $n$ corresponds to the $n^{\text{th}}$ bit of the pattern. (One time unit represents the time between input events.) If the target pattern has $p$ bits equal to 1, then the synapses associated with the 1's should be excitatory and increase the neuron's potential by a maximum of approximately $\left(\frac{\Theta}{p}\right)$. Similarly, the synapses associated with the 0's should be inhibitory and decrease the neuron's potential by approximately $-\left(\frac{\Theta}{p}\right)$. Notice that a non-spike event does not actually release neurotransmitter, but the inhibitory synapses serve as penalty terms for input spikes occurring when there should be none.

As a simple example, consider the target pattern of $[0, 1, 0]$. The solution to this problem requires 3 synaptic connections with delays of 0, 1, and 2 time units, and strengths of $-\Theta$, $\Theta$, and $-\Theta$. Thus, the neuron can only reach threshold, $\Theta$, and

[2]The target pattern may also contain "x" terms, which represent "do not care" bits. In this case, the neuron is to respond if all of the specified target bits are presented, regardless of the value of each input bit which corresponds to an "x" term. The neuron only requires one synapse for each of the specified bits and those which correspond to an "x" do not require a synapse.

respond when the target pattern is presented. This is illustrated in Figure 7.6.

Now consider the problem of recognizing several target patterns. The set of all possible $N$-bit patterns comprise the vertices on an $N$-dimensional cube centered at $0.5\cdot[1,\ldots,1]$. If the synaptic time constants are small enough compared to the underlying event frequency so that the effects from the input spikes do not accumulate, then each spike may be considered an independent event. Thus, the maximum neurotransmitter in a synapse, is approximately $\mathrm{e}^{-1}$, which occurs at time $(d_{ij}+\tau_{ij})$ following an input spike. Now, assume that the neuron is such that: (1) the delay of each synapse is $(N-n+1)$ time units for the corresponding input bits; (2) all of the synapses have the same time constants, $(\tau_{ij}=\tau)$; and (3) and the ratio of $\left(\frac{|T_{ij}|}{R_{ij}}\right)$ is the same for each synapse. Therefore, the voltage response of the neuron at the maximums in neurotransmitter is given by:

$$\begin{aligned} V_i(t) &= \tanh^2\left(\frac{t-t_i^K}{\tau_i}\right)\cdot\left\{R_{i0}+\alpha_i\cdot\left[\vec{X}_i\cdot\vec{S}_i\right]\right\} & (7.2)\\ \text{where:}\quad \vec{X}_i &\equiv [x_1;\, x_2;\, \ldots;\, x_N] = \text{Input bit pattern} & (7.3)\\ \vec{S}_i &\equiv [\operatorname{sgn}(T_{i1})\cdot R_{i1};\ \operatorname{sgn}(T_{i2})\cdot R_{i2};\ \ldots;\ \operatorname{sgn}(T_{iN})\cdot R_{iN}] & (7.4)\\ \alpha_i &\equiv \tanh\left(k_i\mathrm{e}^{-1}\right) & (7.5)\\ k_i &\equiv \left(\frac{|T_{ij}|}{R_{ij}}\right) = \text{Constant for all } j & (7.6) \end{aligned}$$

This equation assumes that the neurotransmitter is 0 for those synapses not currently receiving an input spike, and is $\mathrm{e}^{-1}$ for those synapses that do. The firing condition for the neuron is:

$$\text{if:}\qquad \left[\vec{X}_i\cdot\vec{S}_i\right] \geq \left(\frac{\Theta_i^{\text{effective}}-R_{i0}}{\alpha_i}\right) \qquad \Rightarrow \qquad n_i \text{ fires} \qquad (7.7)$$

where $\Theta_i^{\text{effective}}$ is defined in Equation (4.42). But this equation represents an $(N-1)$-dimensional plane in the input space of $\vec{X}_i$. The unit vector normal to the plane is given by:

$$\hat{n} = \frac{\vec{S}_i}{\sqrt{\sum_{j=1}^{N} R_{ij}^2}} \qquad (7.8)$$

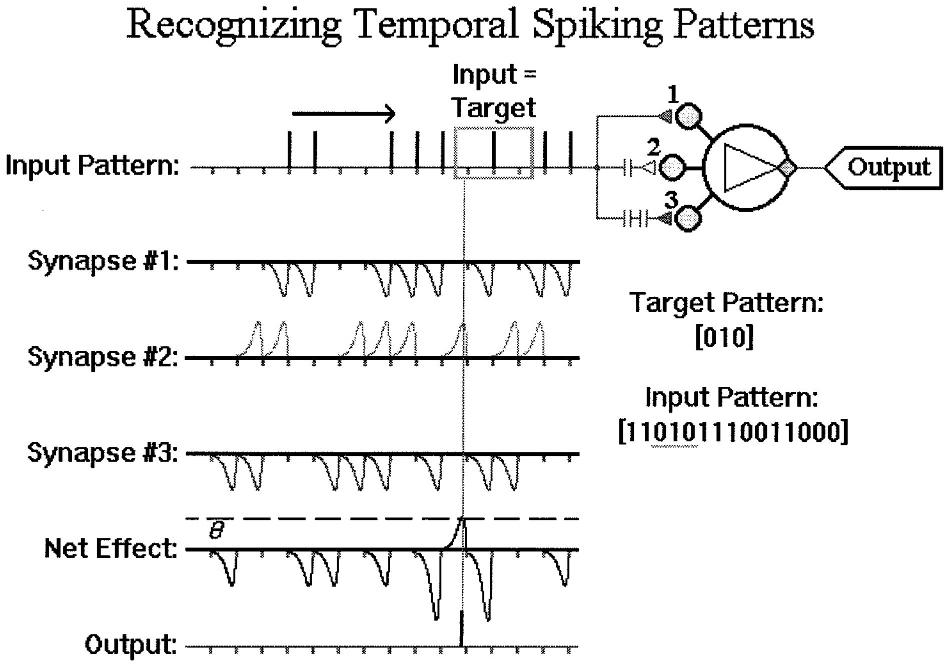


Figure 7.6: **Recognizing a Temporal Spiking Pattern.** The plot shows an example of a neuron being used to recognize the temporal pattern of $[0, 1, 0]$. The single input line forms three synaptic connections. The first synapse is inhibitory and has no delay. The second synapse is excitatory and has a delay of one time unit. The third synapse is inhibitory and has a delay of two time units. (Normally, any time delay is represented by a single delay symbol, but when recognizing temporal patterns, the delays are multiples of each other. Consequently, the number of delay symbols can be used to indicate the relative delay times.) Also shown are the effects from each synapse on the neuron's potential. The net effect is given by the sum of all of the synaptic responses, and it only reaches threshold when the target pattern is presented. Since the spike on the right is the first to arrive at the synapse, the neurotransmitter functions actually appear backward, with time increasing from right to left. Notice that except for the sign and time delay, all three synaptic responses are identical. All possible 3-bit combinations can be found within the sample input pattern, verifying that the neuron only responds to the target pattern. For this example, the time between events was 1.0 with the synaptic delays being: $d_{ij} = 0.0$, 1.0, and 2.0, respectively. All synapses used the parameter values of: $R_{ij} = 1.0$, $T_{ij} = \pm 1.0$, and $\tau_{ij} = 0.15$. The neuron parameters were: $\Theta_i = 0.32$, $\tau_i = 1.0$, and $R_{i0} = 0.0$.

and the distance from the origin is:

$$d = \frac{\Theta_i^{\text{effective}} - R_{i0}}{\alpha_i \cdot \sqrt{\sum_{j=1}^{N} R_{ij}^2}} \tag{7.9}$$

Thus, the input space is linearly divided into two regions, and those inputs on one side of the hyperplane produce output spikes, while those on the other side do not. Of course, the inputs are still restricted to be the vertices of a hypercube, but the neuron parameters may be chosen so that the neuron is able to selectively respond to any linearly separable group of such vertices. The overall operation of this neuron is similar to that of a simple perceptron with threshold units. (See [75] or [42, pp. 92-101] for details on perceptrons.)

Of course, one key difference between the SNM and the perceptron neuron model is that the state of the SNM does not just depend upon its current inputs, but also depends upon its previous activity. Thus, the refractory time constant can be set so that the neuron only produces an output spike for target patterns sufficiently separated in time. (From a geometric standpoint, immediately after the neuron produces an output spike, the refractory coefficient pushes the separating hyperplane out to infinity before moving it back to its initial location. The normal vector does not change; only the distance from the origin varies.) This feature can be particularly useful for preventing the neuron from firing multiple times due to overlapping target patterns. E.g., assume that the neuron is to fire when the 9-bit target pattern of [011000110] is input. If the neuron has a short refractory time coefficient, it fires twice when presented with the 14-bit pattern of [01100011000110]. But if the neuron is only to fire with independent presentations of the target pattern, the time constant can be made sufficiently long so that the neuron is only capable of firing a second time after another 9 bits have been input. Thus, the neuron would only produce an output spike after the 9th bit, and not again after the 14th bit.

The synaptic time constants can also be used to reduce the required number of synaptic connections. For the derivation of Equation (7.2), the synaptic time constants were sufficiently small so that the effects from the input spikes could not accumulate. With this assumption, the maximum value of the neurotransmitter at a synapse was approximately $e^{-1}$. However, if the target pattern has repeated bits, then only one connection is needed to account for the identical, consecutive bits. Using the previous example 9-bit target pattern of [011000110], the neuron would only require 5 input synapses, since the bit combinations of "11," "000," and "11" could each be accounted for with a single synapse. The time constants associated with each of the multiple bit synapses must be chosen so that the synapse's contribution to the neuron potential is only sufficient for producing spikes when all the repeated bits are presented; e.g., if a neuron is only to respond when $n$ sequential 1's are input into a particular synapse, then the neuron's parameters must be chosen so that it can only fire when the neurotransmitter at that synapse approaches the maximum value generated by $n$ sequential input spikes. For consecutive input spikes, the maximum value of the neurotransmitter concentration is given by Equation (E.12), which was derived to explain the latency effect in the SNM; however, since the input spikes have the same time separation between them, the result still holds.[3] The delays associated with the multiple bit synapses must be set so that the maximum neurotransmitter concentration occurs simultaneously at all of the synapses when the *entire* target pattern is presented. For synapses that recognize multiple 0's, the parameters must be chosen so that the inhibitory effect from even one erroneous input spike anywhere within the repeated 0's of the target pattern, prevents the neuron from firing.

Like the simple perceptron neuron model, the SNM can be used in a layered network to respond to bit patterns within different regions of the input space. Each of the neurons in the first layer responds to inputs on one side of a hyperplane, and its output spikes can then go into neurons of a second layer, which perform logical AND functions. These neurons can be designed to respond only when the network's input pattern is within a convex region of the input space. Thus, a single neuron within the second layer is sufficient for responding to any linearly bounded, simply

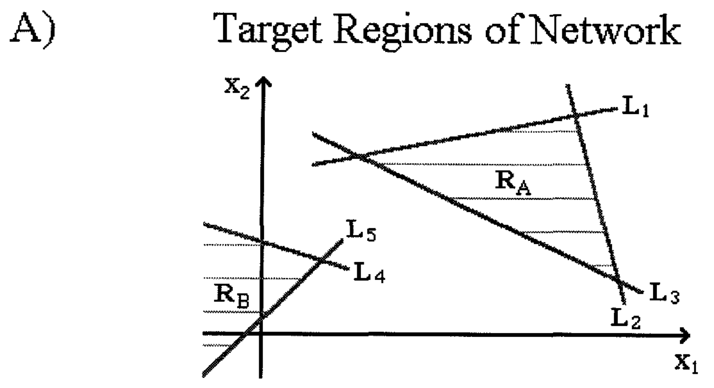
[3]When using Equation (E.12), $x$ represents the time after the last input spike arrives at the synapse. Also, the first expression for $\max\{U(n;x)\}$ should always be used, since after the last input spike there is not another one to further bolster the maximum neurotransmitter level.

connected, finite region of the input space, and the number of neurons required in the first layer depends upon the number of hyperplanes required to define the targeted region. To extend the network's response to arbitrarily shaped regions, a third layer, consisting of a single neuron, can be used to perform the logical OR function on the outputs from the AND neurons of the second layer. Thus, when an input pattern falls within the convex target region of one of the second layer neurons, the neuron in the third layer produces an output spike. (See [35, pp. 86-102] for a geometric description of pattern recognition in networks of perceptrons.) Figure 7.7 shows a three layered network connected to implement a two dimensional pattern recognition task.

In theory, when the inputs are restricted to lie on the vertices of a hypercube, the target region need not be defined by oblique boundaries, as shown in Figure 7.7. But regardless, the neurons can be designed to implement any linear boundary. Since the bits are presented to the neuron sequentially, any mismatch in the timing between events and the synaptic delays can create errors in the values detected. Each input spike causes *every* synapse to reach a maximum/minimum in potential value. It is the time delays between the synapses that determine when the extrema occur. Thus, oblique target boundaries may be used to allow for slight timing discrepancies.

Since every Boolean logic expression can be reduced to conjunctive normal form, no classification task requires more than three layers of neurons. (Notice that non-convex regions can be considered to be the union of convex regions.) Furthermore, Section 7.4 shows how any logic function can be performed with a *single* neuron.[4] Thus, all of the neurons needed for the second and third layers can actually be reduced to one neuron if desired. However, the layered approach is usually easier to design and implement.

[4]In conjunctive normal form, a logic expression is written as *either* a product (AND) of sums (OR) or a sum (OR) of products (AND). The order in which the three neuron layers perform the pattern recognition task uses a sum of products approach, while the single logic neuron described in Section 7.4 uses a product of sums approach; however, it is possible to switch between the conjunctive normal forms.



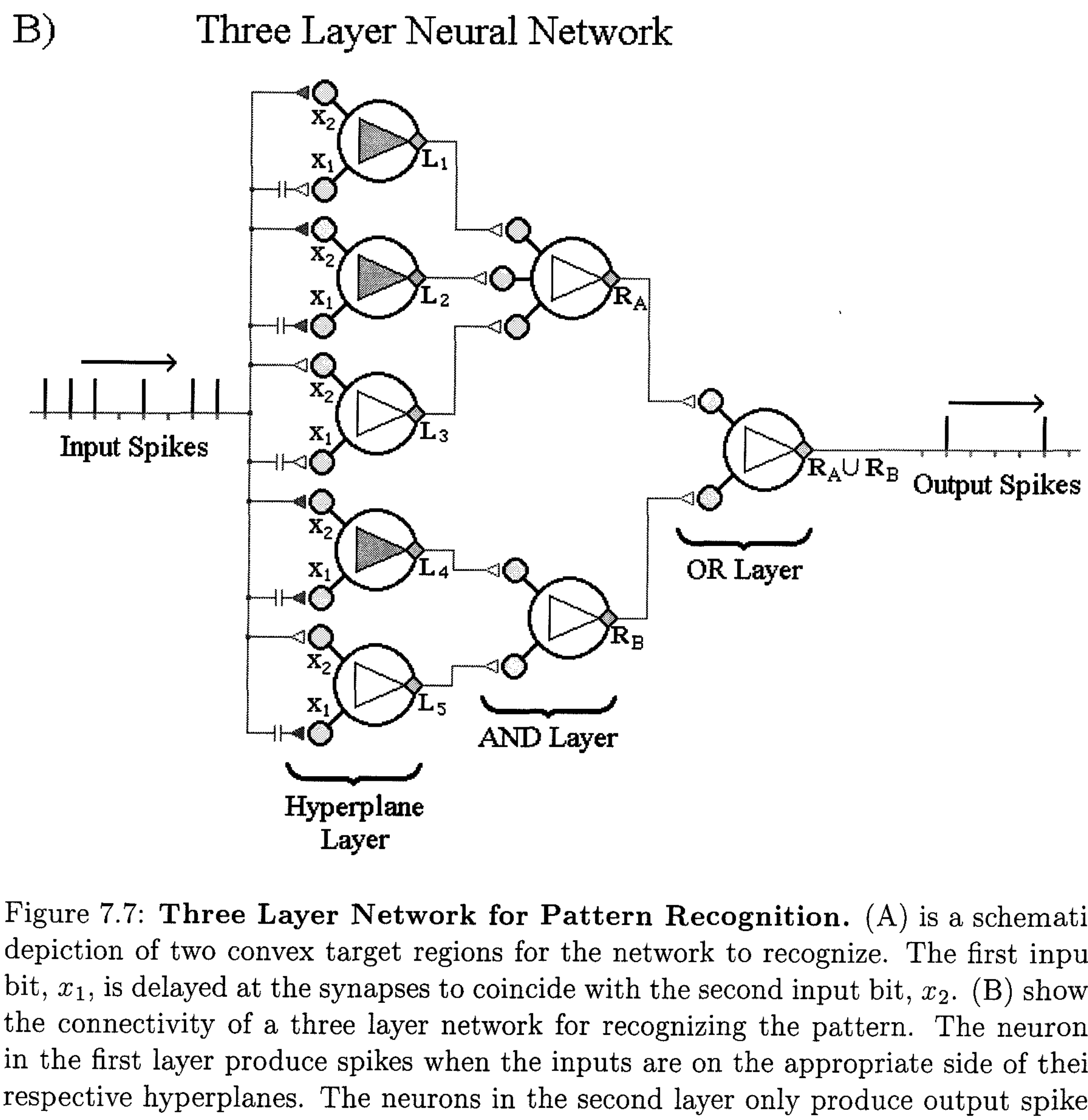


Figure 7.7: **Three Layer Network for Pattern Recognition.** (A) is a schematic depiction of two convex target regions for the network to recognize. The first input bit, $x_1$, is delayed at the synapses to coincide with the second input bit, $x_2$. (B) shows the connectivity of a three layer network for recognizing the pattern. The neurons in the first layer produce spikes when the inputs are on the appropriate side of their respective hyperplanes. The neurons in the second layer only produce output spikes when all of their inputs are spiking. The neuron in the third layer produces an output spike if any of its inputs are spiking. (Notice that the first layer neurons with the origin in their target region must have an intrinsic spike frequency; i.e., $R_{i0} > \Theta_i$.)

### 7.3.3 Recognizing Spatial Patterns

For spatial pattern recognition, the goal is to produce an output spike when one of the target patterns is presented. The target patterns consist of one event on each of several input lines, but the events do not need to occur simultaneously; i.e., each of the input signals may have different synaptic delays. If the neuron has $N$ input signals, then each target pattern may be represented as an $N$-bit column vector of 1's and 0's. As with the temporal pattern recognition task, the events on each of the input lines should have a fixed underlying frequency, but each line may have a different characteristic frequency.

Each input line forms a separate synapse into the neuron. Excitatory synapses correspond to 1's in the target pattern, while inhibitory synapses correspond to 0's in the target pattern. The time delay associated with each synapse depends upon the time difference between the targeted event on that input line, and the last targeted event in the pattern. Thus, all of the input events must be delayed enough so that their *effects* on the neuron occur nearly simultaneously. For the example pattern shown in Figure 7.4, the synapse associated with the third input signal would have the most delay, while the synapse associated with the second input signal would have no delay.

In addition to the different delays associated with each input line, there may also be different window sizes associated with the targeted events on each line. The windows may even be large enough to accommodate more than one event; however, the target pattern may have only a 0 or 1 specified for each window, with *any* input spikes occurring within the window being considered a 1. Notice that since the frequencies differ between the input lines, the relative locations of events within the windows change; e.g., in Figure 7.4 all of the events are initially centered in their respective time windows; however, the second input line has the highest frequency, and when the next event is centered within its time window, the events on the other input lines will not be centered within their windows.

While the time window represents when an event may occur within a target pattern to influence the neuron's potential, it does not represent the time during which a spike is *sufficiently influencing* the neuron's potential; e.g., a spike occurring at the end of the time window will not significantly alter the neuron's potential until after the time window has passed. Thus, the synaptic time constants and delays must be chosen appropriately. (It is important to realize that the time windows are not "real" parameters, but are useful for setting the actual neuron parameters.) If a neuron receives $N$ input lines and is only to produce an output spike when the neurotransmitter is above the level $\Upsilon^+$ at each of its excitatory synapses and below the level $\Upsilon^-$ at each of its inhibitory synapses, then the neurotransmitter contribution from a single isolated spike must remain above $\Upsilon^\pm$ for a period of time equal to the time window, $w_j$. This condition is necessary to ensure that any spike occurring within the window contributes a sufficient amount of neurotransmitter to either excite or inhibit the neuron. Using the definition of $\psi(-)$ in Figure C.1 this condition becomes:

$$\Upsilon^\pm = F\left(\psi\left(\frac{w_j}{\tau_{ij}}\right)\right) \tag{7.10}$$

Solving for $\tau_{ij}$ gives:

$$\tau_{ij} = \frac{w_j}{\psi^{-1}\left(F^{-1}\left(\Upsilon^\pm\right)\right)} \tag{7.11}$$

(Algorithms for calculating $F^{-1}(-)$ and $\psi^{-1}(-)$ are given in Appendix D.) After an input spike arrives, the neurotransmitter level reaches the necessary value of $\Upsilon^\pm$ when the elapsed time reaches:

$$t^{\min} = \tau_{ij} \cdot \psi\left(\frac{w_j}{\tau_{ij}}\right) = \tau_{ij} \cdot F^{-1}\left(\Upsilon^\pm\right) \tag{7.12}$$

This represents the only time after the window when the neurotransmitter level is guaranteed to be at its necessary level if an input spike occurs during the window; i.e., spikes that arrive at the beginning of the time window have already reached their maximum neurotransmitter level and are about to decay below $\Upsilon^\pm$ at $t^{\min}$ after the window, while spikes that arrive at the end of the time window have increasing neurotransmitter levels, and just reach $\Upsilon^\pm$ at $t^{\min}$ after the window. Thus, *the synaptic* *delays must be set so that the occurrences of* $t^{\min}$ *after each window are aligned.*

To summarize, each input line has a time window associated with it, and the synaptic time constants must be chosen so that the effect from each input spike remains sufficiently strong for the length of the time window. However, since it takes time $t^{\min}$ for the neurotransmitter to reach the necessary level, spikes arriving at the end of the time window do not immediately influence the neuron. Therefore, the synaptic delays should not be chosen to align the starting time of the windows, but instead chosen so that the times of $t^{\min}$ after each window occur simultaneously.

When the neuron parameters are chosen appropriately, the neuron acts as a spatial pattern recognizer. However, since the synaptic time constants must be chosen so that the neurotransmitter concentrations can remain at sufficiently high levels for the length of the time windows, there can be an accumulation of neurotransmitter at a synapse with successive input spikes. Consequently, the input spikes can not be considered independent events as they were for temporal patterns. Thus, if one of the input signals has been continuously producing spikes, and then stops, the first few non-spike events occurring within the time window may be incorrectly perceived as spikes due to the previous accumulation of neurotransmitter. This problem is minimized when the length of the time windows are chosen to be less than the time between events in the underlying signal frequencies. However, this choice creates periods when no events occur within one or more of the time windows. The net result is that 1's in the target pattern are harder to recognize with smaller time windows, but larger windows require longer time constants, which can cause non-events to be misinterpreted as spikes, making 0's in the target pattern harder to recognize.

Another problem that can occur is that the neurotransmitter level at one synapse may be sufficiently high to compensate for the lack of an input spike at another synapse; i.e., while the neurotransmitter concentration level is supposed to be above $\Upsilon^+$ at all of the neuron's excitatory synapses, it is possible that it is significantly above this level at one synapse and below at another. This problem is minimized by setting the parameters so that the synapses' tanh functions are well saturated at the required neurotransmitter level. Thus, large amounts of neurotransmitter at one synapse can not significantly influence the neuron's potential more than a synapse with a neurotransmitter level of only $\Upsilon^+$. Also, $\Upsilon^-$ should be chosen to be close to zero so that the contribution from the residual neurotransmitter at the synapses used to recognize 0's can not significantly influence the neuron's potential.

As a design example, consider using a neuron with three input lines to recognize the spatial pattern of $[1; 0; 1]$, where the spike on the first input line is to occur 2 time units after the spike on the third line, and the non-spike event on the second line is to occur 3.5 time units after the spike on the third line. (Time units are measured relative to the time between events on the first input line.) Also, the times between events on the input lines are: 1.0, 1.67, and 2.33, respectively. For this problem to be completely specified, it is also necessary to know the time windows in which the targeted events must occur. Here, they are set equal to the time between events. All of the parameters depend upon the required neurotransmitter levels, which are chosen to be: $\Upsilon^+ = 0.2$, and $\Upsilon^- = 0.05$. Using Equation (7.11), the synaptic time constants, $\tau_{ij}$, are calculated to be: 0.44, 0.38, and 1.02. The values for $t^{\min}$ in Equation (7.12) are: 0.11, 0.02, and 0.26. To determine the necessary synaptic delays, all events are referenced to the first occurring time window, which is on the third input line. Adding the delay before the start of each window and the length of each window to $t^{\min}$, yields times of: 3.11, 5.19, and 2.59. Since the events in the window of the second input are detected last, the other inputs must be delayed to coincide with the second signal; i.e., $d_{ij}$ is: 2.08 $(= 5.19 - 3.11)$, 0.00 $(= 5.19 - 5.19)$, and 2.60 $(= 5.19 - 2.59)$, respectively. To ensure that the tanh functions are well saturated at the required neurotransmitter levels, all of the synaptic receiver weights, $R_{ij}$, are set to 0.26, and the synaptic transmitter weights, $T_{ij}$, are set to: 2.41, -9.64, and 2.41, respectively. The neuron's parameters are:, $\Theta_i = 0.5$, $R_{i0} = 0$, and $\tau_i = 1.0$. The design results are shown in Figure 7.8.

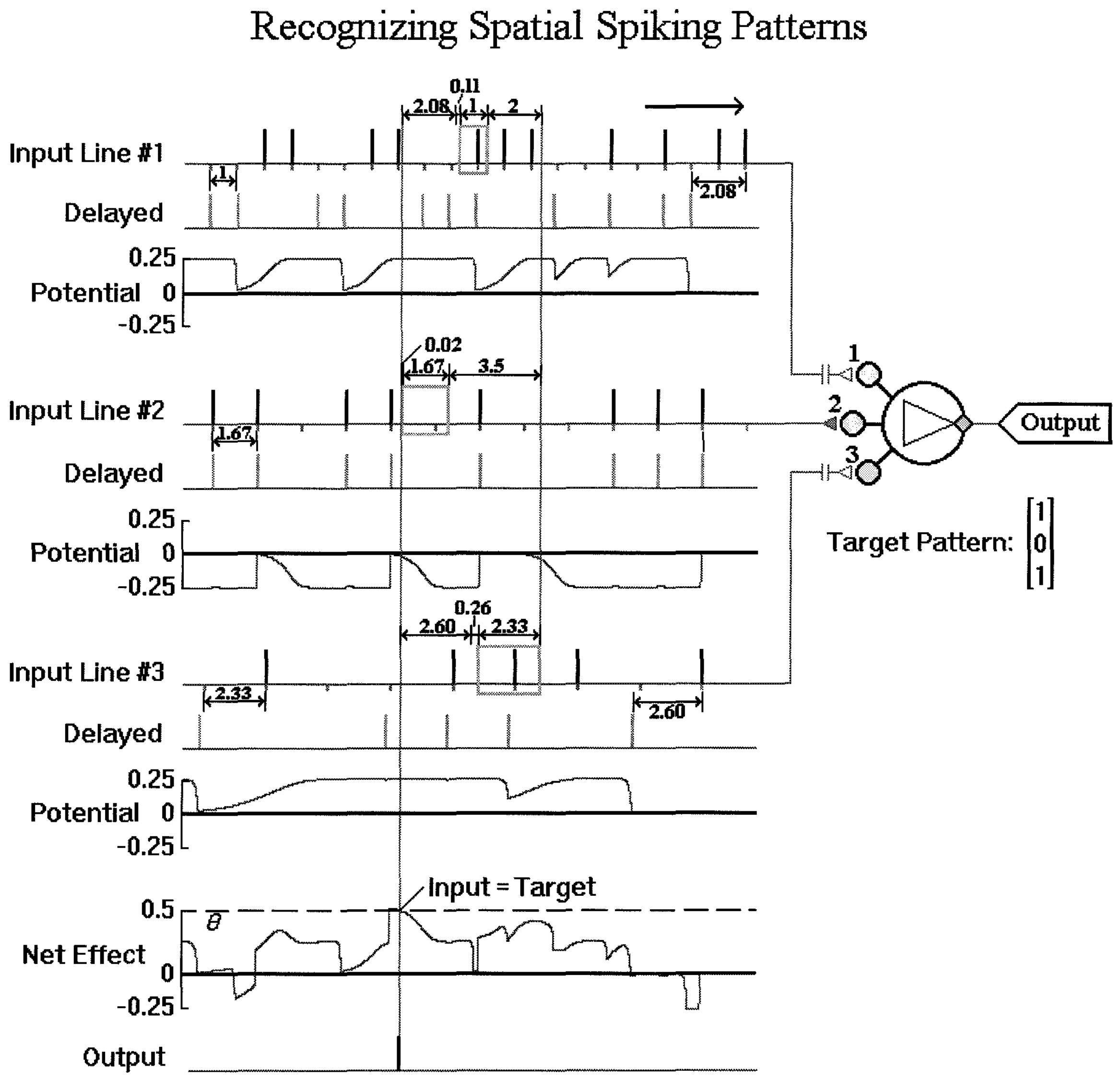


Figure 7.8: **Recognizing a Spatial Spiking Pattern.** The plot shows a neuron being used to recognize the spatial pattern of [1; 0; 1]. There are three input lines, with different underlying frequencies and windows. In addition to the inputs, the diagram also shows a delayed version of the spikes and their effect on the neuron's potential. The sum of the synaptic responses only reaches threshold when the target pattern is presented. Since the spikes on the right are the first to arrive at the synapse, the potential functions appear backward, with time increasing from right to left. While the time windows could have been shown anywhere along the input lines, they are at the location which produced an output spike, and their widths were set equal to the times between events (1.0, 1.67, and 2.33, respectively). The first and second time windows start 2 and 3.5 time units after the third time window. The synaptic time delays were chosen so that the times of $t^{\min}$ after each window align ($t^{\min} = 0.11$, 0.02, and 0.26, respectively); i.e., the first and third inputs required delays of 2.08 and 2.60.

### 7.3.4 Recognizing Spatial-Temporal Patterns

A network designed to recognize spatial-temporal patterns has several input lines, and produces an output spike when one of the target patterns is presented. A target pattern consists of at least two events on one or more input lines. (All input lines should have at least one specified event.) As with spatial patterns, the events need not occur simultaneously, but each of the input lines should have a fixed frequency of events. (The frequencies may differ between inputs.) If the recognizing neuron has $N$ input signals with a maximum of $M$ events specified for any one input line, then the target pattern may be represented by an $N$x$M$ matrix of 1's and 0's.

In principle, a single neuron can be use to recognize any spatial-temporal pattern, with one synapse associated with each target event. However, when the input signals have different frequencies and window sizes, longer synaptic time constants are need to align their effect on the neuron's potential, and these longer time constants make it difficult to identify the temporal aspects of an individual input line. Consequently, the task is best accomplished by using a two layer network, separating the temporal and spatial recognition tasks.

At first it appears that it is possible to use *either* temporal pattern recognizers in the first layer and a spatial pattern recognizer in the second layer *or* spatial pattern recognizers in the first layer and a temporal pattern recognizer in the second. However, both types of recognizers require the input signal to have a fixed frequency of events, and while the output from the temporal pattern recognizer has the same underlying frequency as its input signal,[5] the output from the spatial pattern recognizer does not have a fixed frequency if the frequencies differ between inputs. Thus, only temporal pattern recognizers can be used in the first layer to provide output spikes to a spatial pattern recognizer.

Figure 7.9 shows how a two layer network can be use as a general spatial-temporal

[5]The end of Section 7.3.2 discussed the possibility of using a long refractory time constant to prevent the neuron from firing multiple spikes when presented with overlapping patterns. This forces the minimum time between output spikes to be greater than the input frequency. But since any one input event might be the final bit in the target pattern and produce a spike, provided the neuron has not fired recently, the underlying frequency of *events* is still the same as the input frequency.

pattern recognizer. The neurons in the first layer are temporal pattern recognizers, where each neuron receives spikes from only a single input line. If there are several valid target patterns for one row, then a multilayer feedforward network, as shown in Figure 7.7, can be use to replace any single neuron of the first layer. The neuron in the second layer does the spatial pattern recognition task using the outputs from the neurons in the first layer. Since the neurons in the first layer only spike when presented with their respective target temporal patterns, all of the synapses into the spatial pattern recognizer are excitatory.

As a design example, consider using a two layer network to recognize the five windowed patterns used in Figure 7.9; i.e., $[[0,1,1];[1,0,\mathrm{x}];[1,0,1];[1,1,1];[0,1,\mathrm{x}]]$. Assume that the times between events are given by: $[8;4;5;9;6]$, and the window widths are equal to the number of targeted events multiplied by the time between events; i.e.: $[24;8;15;27;12]$. (Notice that for a spatial-temporal pattern, each window width must be greater than $(M_j-1)\cdot\Delta_j$ and less than or equal to $M_j\cdot\Delta_j$, where $M_j$ is the number of targeted events and $\Delta_j$ is the time between events. This is necessary to ensure that the window is wide enough to contain all targeted events, but can never have more events than the target pattern.) Also, the window on the third input line starts first, with the other windows starting $[8;42;0;17;12]$ time units later, respectively. For each of the temporal pattern recognizers, the parameters can be chosen as they were in Figure 7.6, with the delays, $d_{ij}$, equal to the times between events, the time constants, $\tau_{ij}$, equal to 0.15 of the times between events ($\tau_{ij}=[1.2;0.6;0.75;1.35;0.9]$), the synaptic transmitter weights, $T_{ij}$, equal to $\pm\left(\frac{1}{p}\right)$, where $p$ is the number of target bits equal to one ($T_{ij}=\pm[0.5;1.0;0.5;0.33;1.0]$), and the synaptic receiver weights, $R_{ij}$, equal to 1. Also, the refractory time constants, $\tau_i$, are set equal to half of the times between events ($\tau_i=[4;2;2.5;4.5;3]$).

Now, while the windows represent when the temporal patterns on each input line can occur, they do not represent when the spatial pattern recognizer neuron can expect to receive an input spike from the temporal pattern recognizers in the first layer. Since each *targeted pattern* is only $(M_j-1)\cdot\Delta_j$ time units in length, the temporal pattern recognizer can first output a spike at $(M_j-1)\cdot\Delta_j$ units after the *start*

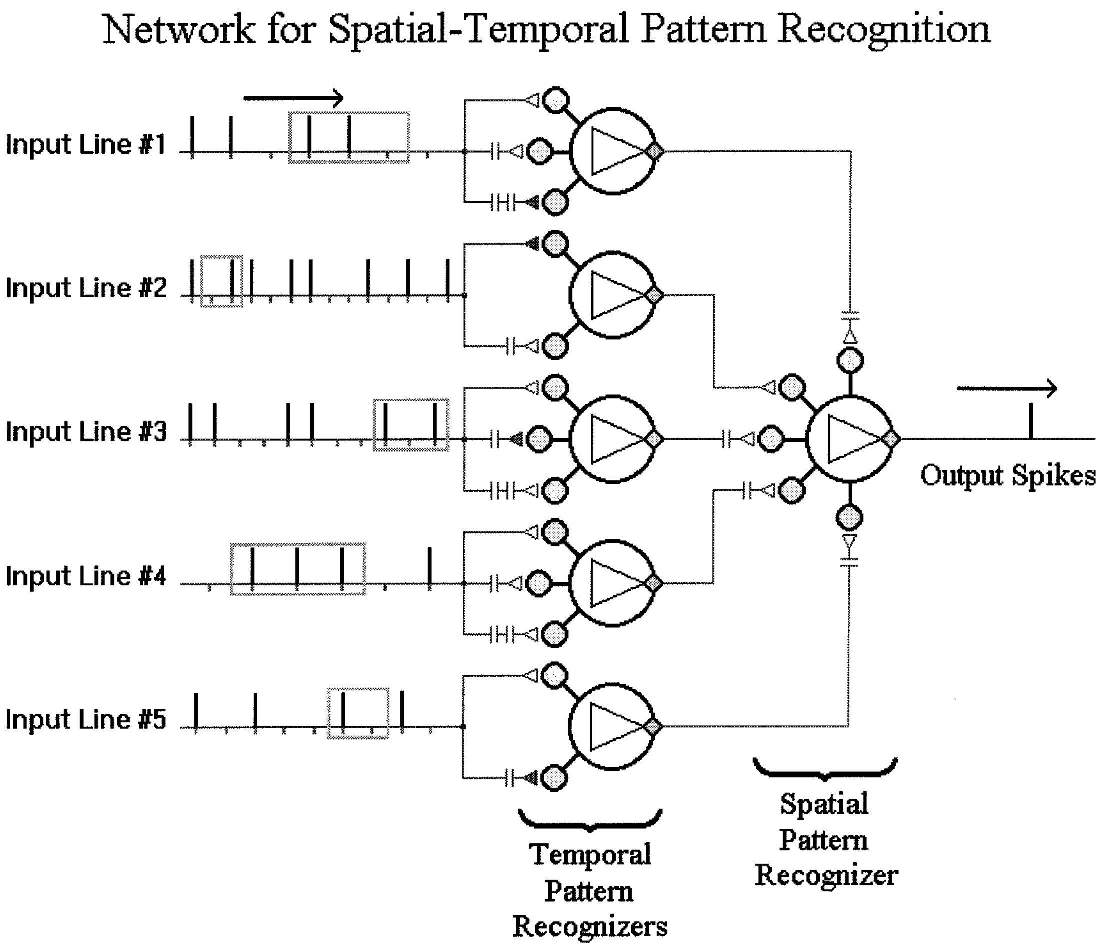

Network for Spatial-Temporal Pattern Recognition
Input Line #1
Input Line #2
Input Line #3
Input Line #4
Input Line #5
Output Spikes
Temporal Pattern Recognizers
Spatial Pattern Recognizer

Figure 7.9: **Two Layer Network for Spatial-Temporal Pattern Recognition.** The plot shows the connectivity of a two layer network for spatial-temporal pattern recognition. Each neuron in the first layer receives input spikes from one input line, and acts as a temporal pattern recognizer. The number of synapses into each neuron depends upon the number of bits to be recognized in the corresponding row of the target pattern, and the synaptic delays are based on the input frequency. The neuron in the second layer acts as a spatial pattern recognizer, and fires only if the correct temporal patterns are found on *all* of the input lines. Its synaptic delays are determined by the time differences between the window locations in the target pattern.

of the window. And when the target pattern occurs at the very end of the time window, the temporal pattern recognizer should produce a spike before the neurotransmitter contribution from the last possible spike in the target pattern reaches its maximum at $\tau_{ij}$ time units after the *end* of the window. Therefore, the relative start times for the windows associated with the spatial pattern recognizer are found by summing the start times for the windows of the temporal pattern recognizers with $(M_j - 1)\cdot\Delta_j$; i.e.: $[24; 46; 10; 35; 18]$. Similarly, the relative end times for the windows associated with the spatial pattern recognizer are found by summing the end times of the windows of the temporal pattern recognizers with the synaptic time constants; i.e.: $[33.2; 50.6; 15.75; 45.35; 24.9]$. Obviously, the difference between the start and end times of these windows provides their widths; i.e.: $[9.2; 4.6; 5.75; 10.35; 6.9]$. Notice that the spatial windows are considerably smaller than the temporal windows, and all of the target inputs are 1's. With $\Upsilon^+ = 0.2$ as in Figure 7.8 and using Equation (7.11), the synaptic time constants for the spatial inputs are: $[4.03; 2.01; 2.52; 4.53; 3.02]$. Using Equation (7.12), the values for $t^{\min}$ are: $[1.04; 0.52; 0.65; 1.17; 0.78]$. Adding $t^{\min}$ to the end times of the windows gives: $[34.24; 51.12; 16.40; 46.52; 25.68]$. Therefore, to align the times of $t^{\min}$ after each window, the spatial neuron's synaptic delays are: $[16.88; 0.00; 34.72; 4.60; 25.44]$ $(= 51.12 - [34.24; 51.12; 16.40; 46.52; 25.68])$. Finally, the synaptic weights can be chosen as they were in Figure 7.8. (Here, all inputs are excitatory and correspond to a 1 in the target pattern.) Previously, with two excitatory inputs, $R_{ij} = 0.2632$, so with five excitatory inputs, $R_{ij} = 0.1053$ $(= 2\cdot 0.2632/5)$. And to keep the same ratio between $T_{ij}$ and $R_{ij}$, $T_{ij}$ is set to 0.9642. Also, the neuron's parameters are: $\Theta_i = 0.5$, $R_{i0} = 0$, and $\tau_i = 6$. Figure 7.10 shows the simulation results from this network.

### 7.3.5 Problems with Variable Frequencies

The design of all of the pattern recognizers previously discussed assumes that the input signals have a fixed frequency of events, which is possible with a stimulus detector, provided that the pace neuron is operating in its high gain region. But

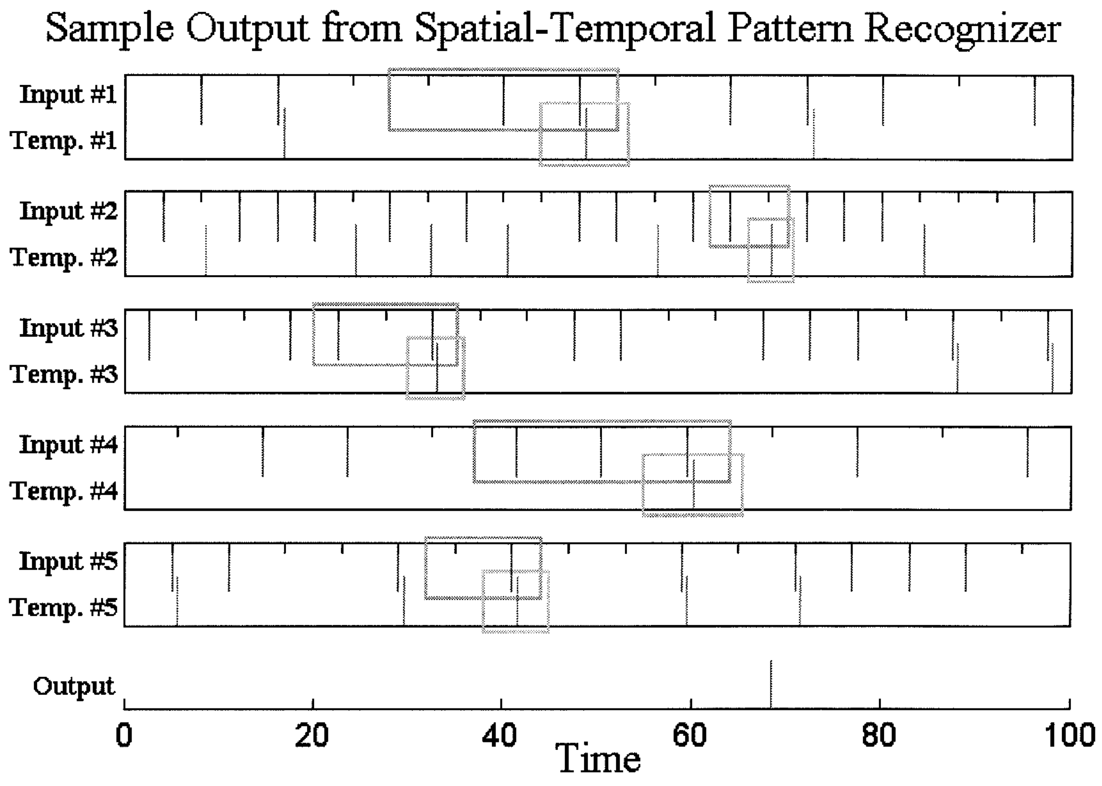


Figure 7.10: **Recognizing Spatial-Temporal Spiking Patterns.** The plot shows a sample output for the spatial-temporal pattern recognizing network in Figure 7.8. The five long boxes show the input spike trains and the output spikes from the temporal pattern recognizers of the first layer. The hash marks on the top lines indicate the times of the input events. The last line represents the output spikes from the spatial pattern recognizer of the second layer. Also shown are the windows around the events which matched the target pattern, and led to the single output spike. The windows on the input lines are the same as those shown in Figure 7.8, while the windows on the temporal neurons' outputs represent the effective time windows for the spatial neuron's inputs. Notice that since time increases from left to right, the spike trains are opposite of those shown in Figure 7.8, but the input spike patterns exactly match from 20 to 75.

what if the input spikes do not have a fixed frequency? The purpose of the frequency assumption was to set the synaptic delays and time constants.

For the spatial pattern recognizer, the frequency can vary since the widths and relative locations of the time windows remain fixed. In fact, because there is only one target event per input line and spikes occurring anywhere within the window are considered 1's, the constraint of a fixed frequency of events can be relaxed. However, higher input frequencies increase the chance of a non-spike event being misinterpreted as a spike event. This is because the neurotransmitter contribution from a series of closely spaced spikes accumulates and does not dissipate as rapidly as the neurotransmitter from a single isolated spike. Thus, even though there may not be any spikes within a time window, the residual neurotransmitter from a previous burst of spikes can have the same effect on the neuron's potential as an actual spike within the time window. In some circumstances, this may have some possible advantages, with short bursts of spikes being used to represent a regularly spaced series of spikes. Also, at higher input frequency there are several events within each window, so several consecutive non-spike events are required before the input signal is actually perceived as a 0. If the synapse is excitatory (input line corresponds to a 1 in the target pattern), the higher input frequency causes the neuron to become more likely to fire, and if the synapse is inhibitory (input line corresponds to a 0 in the target pattern), the neuron becomes less likely to fire. Therefore, higher input frequencies can be thought of as causing an expansion in the *effective* time window, making 1's more likely to be detected.

When the input frequency for a spatial pattern recognizer is lower than anticipated, it is possible for the width of the time window to be less than the time between input events. In this case, there will often be occasions when there are no events occurring within the window. Since the parameters are chosen so that a spike must occur sometime within the window for the input signal to be considered a 1, the input patterns will be perceived to contain 0's even if there are spikes occurring at every possible event. Thus, lower input frequencies can be thought of as causing a contraction in the *effective* time window, making 0's more likely to be detected.

In general, the temporal pattern recognizer is more sensitive to input frequency variations than the spatial pattern recognizer. The underlying frequency sets the necessary time delay of the synapses, and any changes in the frequency affect the timing between the bits of the input pattern. Essentially, each synapse can be thought of as having a time window associated with it, which is equal in width to the time between spikes for which the neuron was designed. But since each synapse is delayed by exactly one time unit more than the previous one, the windows do not overlap. Different frequencies can cause the events being compared at the input synapses to not be aligned; e.g., with a higher frequency, the first synapse may be affected by both the first and second input events instead of just the first, while the second synapse may be affected by the third and fourth input events instead of the second. (As with the spatial recognizer, high frequency spikes can lead to a buildup of neurotransmitter which can cause non-spike events to be perceived as spikes; however, since the synapses usually have smaller time constants, the problem is less severe in temporal pattern recognizers.) Therefore, when a target pattern arrives at an incorrect frequency, it is often not recognized. Figure 7.11 shows that when the same input pattern for the temporal pattern recognizer of Figure 7.6 is presented at a higher frequency, the neuron is unable to respond when the target pattern arrives.

While input spikes arriving at higher frequencies usually go unrecognized, one important exception to this is when the temporal target pattern consists of all 1's. Such a pattern can be used to test when an input signal is spiking at or *above* a specified frequency. Since the maximum neurotransmitter level depends upon the time between incoming spikes, a neuron with a single input synapse can be used to test the input spike frequency; however, a short burst of relatively few spikes causes the neuron to reach threshold and fire. By using a temporal pattern recognizer, with all 1's in the target pattern, the input spikes must continue over the period of time specified by the maximum synaptic delay. Thus, high frequency spikes are only recognized if they continue for a sufficient duration of time. The neuron responds when the *average* frequency of the input spikes is above a preset limit, but it is not susceptible to errant high frequency bursts.

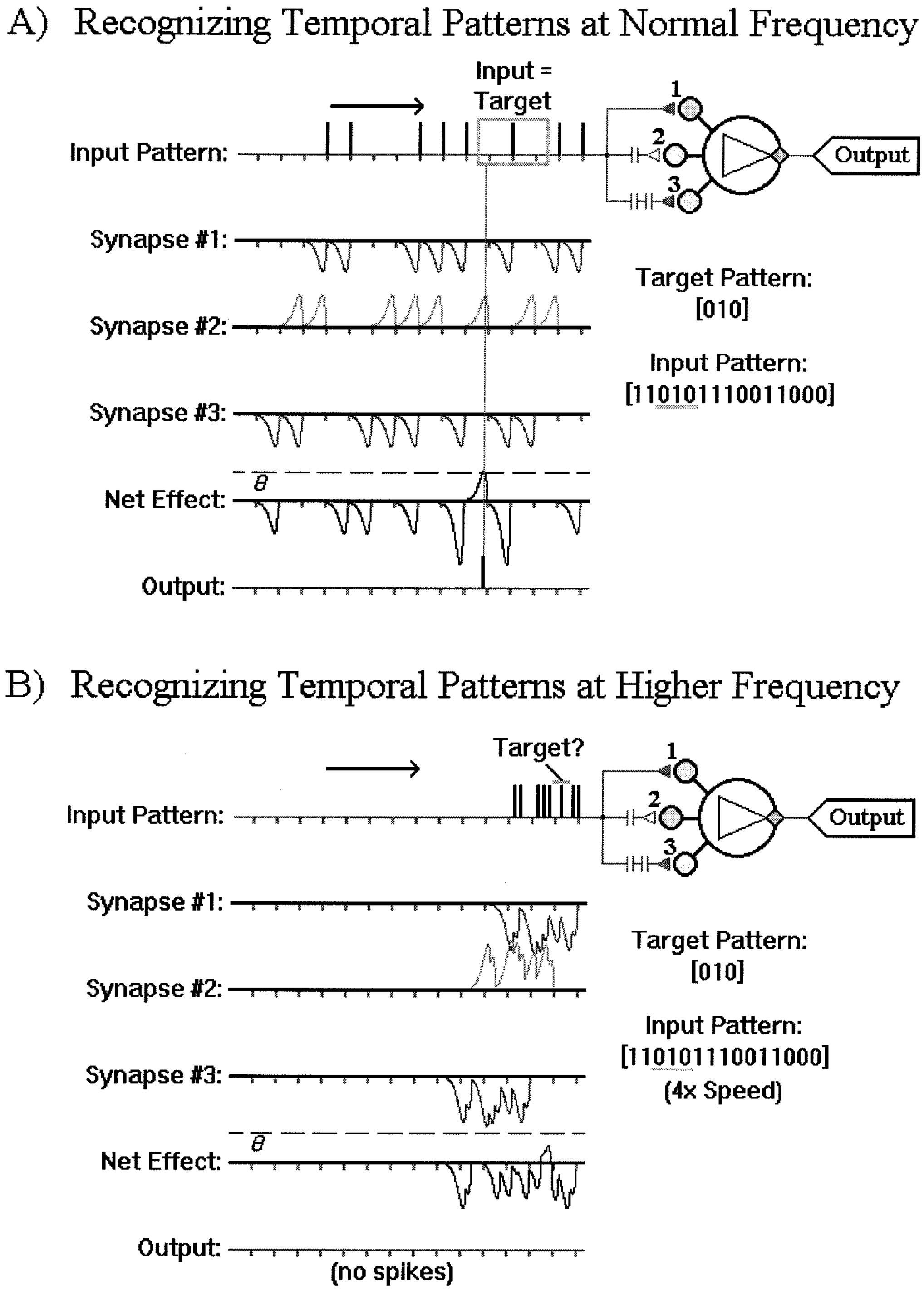


Figure 7.11: **Recognizing Temporal Patterns at Different Frequencies.** The plot shows two neurons being used to recognize the temporal pattern of $[0, 1, 0]$. (A) is the same as Figure 7.6. When the input pattern matches the target pattern, an output spike is produced. (B) uses the same neuron parameters, but the input spikes are presented at four times the frequency of (A). Since the synaptic delays were chosen for a lower frequency, the neuron is unable to recognize the target pattern.

## 7.4 Evaluating Logic Expressions

The synaptic clusters discussed in Section 3.4.3 can be used to evaluate any Boolean logic expressions. When the $T_{ijl}$ variables of Equation (3.14) are sufficiently large (and positive), the tanh function associated with each cluster will saturate near the value of one if any of the cluster's inputs are spiking. In this sense, a synaptic cluster performs the logical OR function. Furthermore, the $R_{ij}$ coefficients of the clusters can be set so that the neuron can not produce an output spike unless all of the clusters are saturated. This allows the neuron to perform the logical AND function. Since all Boolean expressions can be written in conjunctive normal form as a product (AND) of sums (OR), a single neuron can evaluate *any* logic expression. A neuron design to evaluate a specific Boolean expression for a given set of input variables is called a "logic neuron."

While different weight values can be used for each input connection, $T_{ijl}$, and each synaptic cluster, $R_{ij}$, to implement a "fuzzy-like logic" scheme where some variables and clauses have more importance than others, in the simplest case (standard Boolean logic), all of the $T_{ijl}$ variables are set equal ($T_{ijl} = T$), and all of the $R_{ij}$ variables are set equal ($R_{ij} = R$). (Notice that all of the inputs are excitatory.) Also, all of the synapses can be assumed to have the same time constant, ($\tau_{ijl} = \tau$), and no delay, ($d_{ijl} = 0$). Now, when $R_{i0} = 0$, the neuron's potential is given by:

$$V_i(t) = \tanh^2\left(\frac{t - t_i^K}{\tau_i}\right) \cdot \left[R \cdot \sum_{j=1}^{N_i} \tanh\left(\left(\frac{T}{R}\right) \cdot \sum_{l=1}^{M_{ij}} U_{ijl}(t)\right)\right] \tag{7.13}$$

Here, $N_i$ is the number of clusters (clauses) and $M_{ij}$ is the number of input connections (literals) into the $j^{\text{th}}$ cluster.

The problem in designing a logic neuron is deciding how to choose the appropriate neuron parameters. Assume that all of the input events are actually the synchronized outputs from a stimulus detector (Section 7.2), whose pace neuron has a time between spikes of $\Delta_p$. While each spike may lead the pseudospikes by a slightly different amount, all of the occurring spikes are within *dt* of each other (see Equation 6.52).

Since the neuron is to respond if just one of the inputs into each of the clusters receives a spike, the minimum value of $\sum_{l=1}^{M_{ij}} U_{ijl}(t)$ for each cluster to which the neuron must respond is given by:

$$\sum_{l=1}^{M_{ij}} U_{ijl}(t) \geq f\left(\psi\left(\frac{dt}{\tau}\right)\right) \tag{7.14}$$

Here, the lower bound on the total neurotransmitter concentration is given by the contribution from just one spike, $f(-)$, and the spread of the input spikes reduces the maximum from $\mathrm{e}^{-1}$ to $f\left(\psi\left(\frac{dt}{\tau}\right)\right)$ (see the definition of $\psi(\Delta)$ implied by Figure C.1).

The neuron's potential should be above threshold when all of the synaptic clusters have at least one input spike. This applies even if the logic neuron already produced an output spike at its last opportunity, which occurred $\Delta_p$ time units ago. Therefore, the minimum value for $V_i(t)$ that should result in an output spike is given by:

$$\tanh^2\left(\frac{\Delta_p}{\tau_i}\right)\cdot\left\{N_i\cdot R\cdot\tanh\left[\left(\frac{T}{R}\right)\cdot f\left(\psi\left(\frac{dt}{\tau}\right)\right)\right]\right\} \geq \Theta_i \tag{7.15}$$

It is easy enough to choose the neuron's parameters to make this equation hold, but the trick is to make sure that the neuron does not fire even if only one synaptic cluster does not receive an excitatory spike input. The worst case scenario occurs when all of the possible inputs have been continuously spiking, and at one synaptic cluster they suddenly stop. When this occurs, the neuron should stop producing output spikes. Using Equation (4.23), the neurotransmitter available at the synaptic cluster which stops receiving input spikes is given by:

$$\sum_{l=1}^{M_{ij}} U_{ijl}(t) = \left(\frac{M_{ij}}{\kappa\left(\frac{\Delta_p}{\tau}\right)}\right)\cdot f\left(\frac{t+\Delta_p}{\tau} + \psi\left(\frac{\Delta_p}{\tau}\right)\right) \tag{7.16}$$

Here, $t = 0$ refers to the time of the last set of input spikes at the *other* synapses. The last input spikes at this synapse occured $t + \Delta_p$ time units ago. Notice that the previous input spikes are assumed to have been periodic with $P^1 = \Delta_p$; i.e., this synapse was receiving input spikes at every possible pseudospike until now.

For all of the synaptic clusters that are still receiving input spikes, the tanh function must be less than one. Therefore, the neuron is guaranteed not to produce an erroneous output spike when there are no input spikes into one of the clusters if:

$$(N_i - 1)\cdot R + R\cdot\tanh\left[\left(\frac{T\cdot\max\{M_{ij}\}}{R\cdot\kappa\left(\frac{\Delta_p}{\tau}\right)}\right)\cdot f\left(1+\frac{\Delta_p}{\tau}\right)\right] < \Theta_i \tag{7.17}$$

In this equation, $(N_i - 1)\cdot R$ represents an upper bound on the maximum contribution to the neuron's potential from the $N_i - 1$ synaptic clusters that are still receiving input spikes. Also, for a worst case scenario the synaptic cluster that stops receiving input spikes is assumed to be the cluster with the most input synapses. Notice that the maximum neurotransmitter concentration occurs when the argument of $f(-)$ is 1, and while the other synapses are at there maximum level, the "off" synapse, which did not received any input spikes at the last pseudospike, had its last maximum neurotransmitter level $\frac{\Delta_p}{\tau}$ time units ago. Thus, the argument of its neurotransmitter function is $\left(1 + \frac{\Delta_p}{\tau}\right)$.

Any set of neuron parameters that satisfies constraint Equations (7.15) and (7.17) allows the neuron to evaluate a particular Boolean logic expression. Notice that when a set of parameters is found for a given value of $\Delta_p$, any larger value for $\Delta_p$ may also be used; i.e., increasing $\Delta_p$, increases the RHS of Equation (7.15) and decreases the LHS of Equation (7.17). Thus, while the neuron has a maximum input frequency (minimum value of $\Delta_p$) for which it is guaranteed to work, lower input frequencies may be used without any problem. This is in contradistinction to the temporal pattern recognizer, which was designed for a particular input frequency.

As a simple example, assume that a neuron is to evaluate the expression of:

$$\begin{aligned} B(x_1,\ldots,x_5) &= (x_1+x_2+x_3)\cdot(\bar{x}_1+x_4+\bar{x}_5)\cdot(\bar{x}_2+\bar{x}_3+x_5)\cdot(\bar{x}_2+\bar{x}_4+\bar{x}_5) \\ &\quad \cdot(\bar{x}_1+\bar{x}_4+\bar{x}_5)\cdot(x_3+\bar{x}_4+x_5)\cdot(\bar{x}_2+x_3+x_4) \\ &\quad \cdot(x_1+x_2+\bar{x}_3)\cdot(\bar{x}_1+x_2+x_5)\cdot(x_1+\bar{x}_2+\bar{x}_4) \end{aligned} \tag{7.18}$$

Here, $x_i$ is a binary variable that takes the value of 0 or 1, and $\bar{x}_i$ represents its inverse; i.e., if $x_i$ is 0, then $\bar{x}_i$ is 1 and vice versa. (This problem has a unique solution of $\vec{X} = [0,1,1,0,1]$.) Notice that $N_i = 10$, which represents the number of clauses, and $\max\{M_{ij}\} = 3$.[6] The network can be connected as shown in Figure 7.12.

All of the input spikes into the logic neuron are first synchronized. Assume that the pace neuron has $\Delta_p = 0.4$, and that the identity and inverse neurons have the same parameters as the neurons shown in Figures 6.19 and 6.23. Of these neurons, the inverse one has the larger possible lead time with $dt < 0.015345$. (In general, inverse neurons always have larger possible lead times – compare Equations (6.52) and (6.55).) To satisfy the constraint equations, the logic neuron's parameters were chosen as follows: $\Theta_i = 0.5$, $\tau = 0.01$, $\tau_i = 0.08$, $T = 10$, and $R = 0.05105$. The results from a set of random input spike trains are shown in Figure 7.13. The logic neuron correctly evaluates the Boolean expression and only produces an output spike when the solution vector is presented.

Finally, it should be noted that as the number of clauses and literals increases, the region in parameter space for which Equations (7.15) and (7.17) may be satisfied shrinks. In theory, for any finite number of clauses and literals, a solution can always be found, but the neuron becomes very susceptible to noise and rounding errors in the simulation; i.e., the difference in the logic neuron's potential voltage is small when there is only one true literal in each clause and the neuron should fire (Equation (7.15)) versus when all of the literals in all of the clauses are true except for one clause which is all false and the neuron should not fire (Equation (7.17)). Thus, for large expressions, a two layered approach similar to Figure 7.7 is best. (This is actually a three layer network, but only the last two layers are needed to evaluate a logic expression. Also, this figure performs the AND functions before the OR function, but the neurons may be switched to evaluate the expression in the same order as the logic neuron. See footnote on page 153.) With the two layer network, each clause is evaluated independently of the others. And only if a clause is true does

[6]Any Boolean expression can be reduced to conjunctive normal form, and any expression in conjunctive normal form can be reduced to one containing only 3 variables per clause; however, additional Boolean "dummy" variables may need to be used. Expressions with only 3 variables per clause are referred to as 3-SAT problems. To use a neuron to evaluate a logic expression, it is not necessary to reduce it into the 3-SAT formulation.

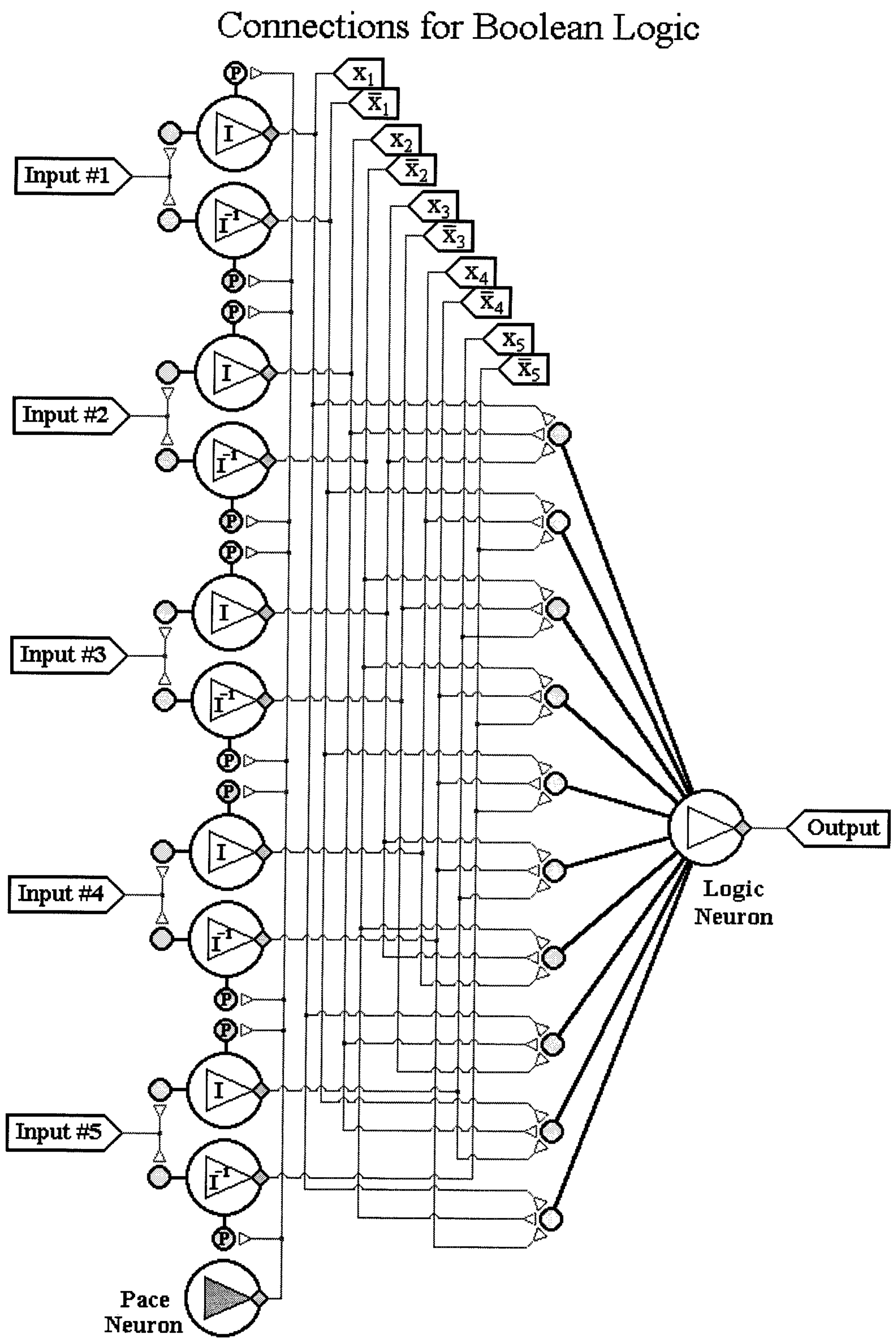


Figure 7.12: **Connections for Boolean Logic.** The diagram shows how the five inputs may be connected to ten synaptic clusters to evaluate $B(x_1, \ldots, x_5)$ in Equation (7.18). The inputs go through a stimulus detector to synchronize their signals and calculate their inverses.

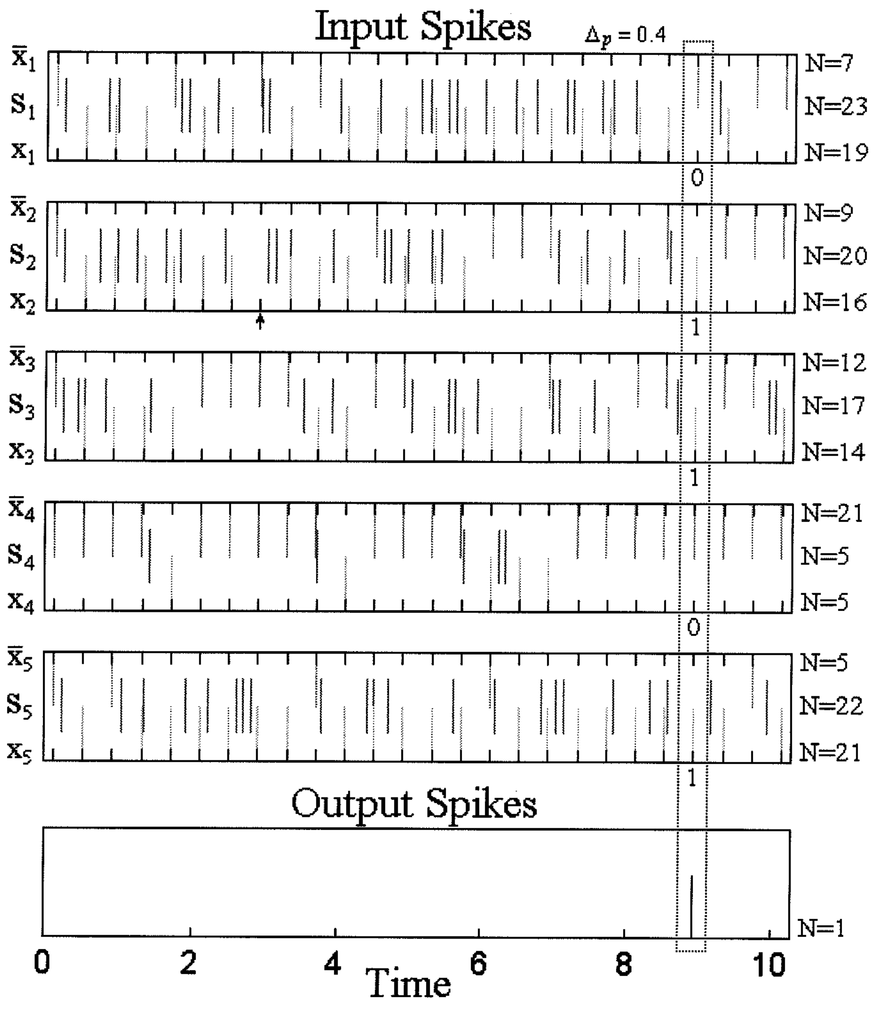


Figure 7.13: **Response of Logic Neuron.** The first five boxes show the input spikes, while the last box shows the output from the logic neuron. The input spike trains were randomly generated and input into synchronized identity and inverse neurons. For each of the input boxes, the spikes in the bottom half, $x_i$, represent the output from the identity neuron; the spikes in the top half, $\bar{x}_i$, represent the output from the inverse neuron; and the spikes in the middle, $S_i$, which partially overlap both the identity and inverse spikes, are the randomly generated input signals. The logic neuron only produced one output spike, which occurred when the synchronized inputs corresponded to $\vec{X} = [0, 1, 1, 0, 1]$, the solution to the Boolean expression. (The hash marks indicate the times of the pseudospikes, and the arrow below one of the pseudospikes in the second box indicates a time when neither the identity nor the inverse neuron produced a spike. The values of $N$ to the right of each signal represent the number of spikes.)

the corresponding neuron in the first layer output a spike to the neuron in the second layer. The second layer neuron only produces spikes when all of its input clauses are firing; i.e., the entire expression is true.

## 7.5 Memory Networks

Previously, Section 6.3 showed how a neuron could be used to save the state of a single input bit. This section describes how entire segments of an input spike train can be stored. The basic idea is to have a network with feedback, so that a series of input spikes continues to oscillate indefinitely through the network. Memory networks may be constructed in a variety of ways, depending upon the length of the memory and how the storage is controlled. When an input line contains a series of spikes, a connection with delay can be used to temporarily store all input spikes. But to permanently store the spikes within any finite memory, it is necessary to limit the time range in which the memory is sensitive to new input spikes. This sensitivity time window determines the length of the memory, with spikes occurring before or after the time window having no effect on the stored memory. And when it is necessary to store a new memory, all information from the previous memory is lost.

Figure 7.14 shows a synchronized memory network. The network uses a pace neuron to synchronize all network spikes. There are two inputs: one contains the input spike train to be stored, and the other is a controlling signal, which indicates when the previous memory should be erased and a new memory formed. Both input signals are first synchronized before connecting to the two neurons actually responsible for storing the memories. Only one of the neurons is used to store the memory, while the other is used to indicate the start of the spike pattern. Since the memory is created by using a delayed feedback connection, the memory neuron is continuously producing output spikes. Therefore, there is no way to determine which output spike is the initial spike in the memorized pattern. The second neuron uses the same delayed feedback connection, but it only spikes at the time of the initial memorized event, thus indicating the start of the memorized pattern.

Since all of the network's spikes are synchronized by the pace neuron, the memory network can be described in terms of the number of events that it holds in memory, $N$. Thus, the time length of the memory is actually $(N-1)\cdot\Delta_p$, where $\Delta_p$ is the time between spikes from the pace neuron.

When designing a synchronized memory network, the parameters must be chosen so that a control spike erases the current memory and only allows the input spikes occurring within the sensitivity time window to be stored in memory. Ideally, an input control spike is able to precisely regulate the network by turning off the feedback synapse that allows the currently stored spikes to continue, and turning on the input synapse so that new input spikes can be stored in memory. And when the elapsed time is equal to the length of the memory, the input synapse is turned off while the feedback synapse is turned on again. The problem with actually implementing such a design is that the effects from a control spike do not instantaneously turn on or off. The neurotransmitter released at a presynapse takes the same functional form as the neurotransmitter released at a synapse, $te^{-t}$; thus, after a control spike arrives at a presynapse, its effect on the synapse increases to a maximum before gradually decaying away. For a synchronized network, this is not a significant problem since the presynaptic parameters and delays can be chosen so the transition regions, from off to on and on to off, occur between pseudospikes.[7]

Now, the presynaptic connection from the control identity neuron onto the input synapse of the memory neuron must increase the effective weight so that input spikes are able to produce output spikes. The weight must be sufficiently elevated for at least $N-1$ time units, but less than $N$ time units. If any shorter, not all possible input spikes would be stored in memory, and if any longer, later input spikes would be erroneously included in the stored memory. Of course, the time delays for the

[7]If the total length of the memory is long, while the time between pseudospikes is short, presynaptic clusters can be used to provide for faster transitions and less susceptibility to noise. In this case, all of the connections into the cluster still come from the control signal, but they each have different delays, and their time constants are smaller than would be needed for a single connection. Thus, several copies of the same spike arrive at different times, with each copy only able to influence the synapse a short duration. Essentially, the idea is to allow a single control spike to have the same effect as a burst of spikes.

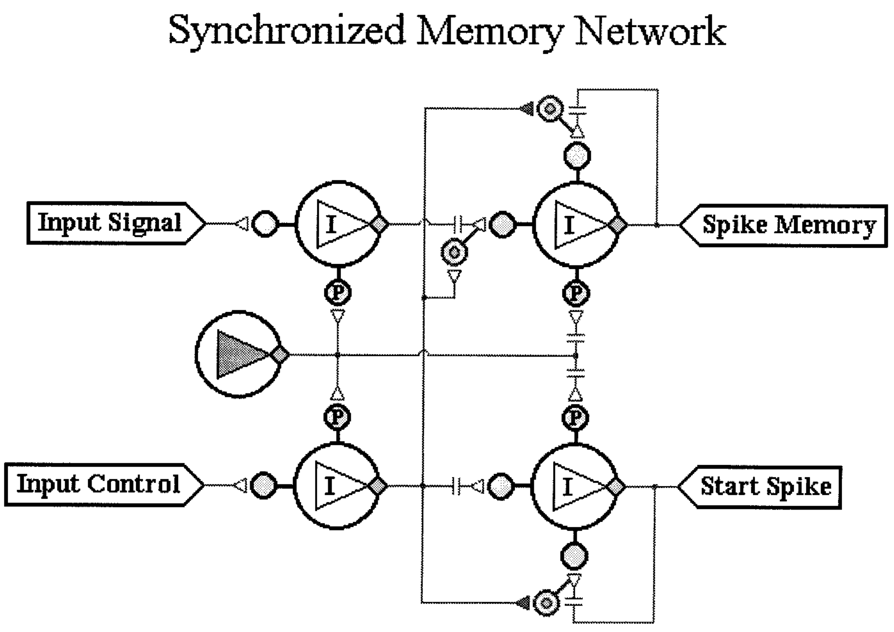


Figure 7.14: **Synchronized Memory Network.** The plot shows the connectivity of a synchronized memory network. The network uses a pace neuron, with $R_{i0} > \Theta_i$ to synchronize all network spikes. The top right neuron is used to store the memorized pattern, while the bottom right neuron is used to store the start spike, which indicates the beginning of the memorized pattern. The output from the synchronized control neuron forms an inhibitory presynaptic connection onto the feedback synapse, and an excitatory presynaptic connection onto the input synapse. The inhibitory presynapse prevents the currently stored memory from producing any more output spikes. Notice that the length of the feedback delay determines the maximum number of events that can be stored in memory. The excitatory presynapse allows the input spikes to produce new output spikes, which are then store in memory. When there is not an initial control spike, the effects from the input spikes are too weak to produce any output spikes, and the current pattern in memory remains unaltered. The neuron used to store the start spike is identical to the one used to store the spike pattern, except that the control spikes are used as the input signal, and there is no presynaptic connection on the input synapse. Thus, each control spike automatically creates a new start spike.

input and pace signals must be chosen so that the elevated level begins before the next possible input spike.

Similarly, the presynaptic connection from the control identity neuron onto the feedback synapse of the memory neuron must decrease the effective weight so that the feedback spikes are unable to produce output spikes. The weight must be sufficiently depressed for at least $N - 1$ time units, but less than $N$ time units. If any shorter, feedback spikes from the previous memory could erroneously produce output spikes and be included in the new memory. If any longer, the new spikes would be unable to produce output spikes and would be lost from memory. The delay on the feedback synapse must be chosen so the neuron is able to store $N$ events, with the maximum contributions from the feedback spikes occurring simultaneously with the pseudospikes of the output neuron so as to produce new output spikes.

Figure 7.15 shows a sample output for the synchronized memory network of Figure 7.14. The pace neuron's parameters were: $\Theta_i = 0.5$, $R_{i0} = 1.0$, and $\tau_i = 0.4538$, which yielded an output spike every 0.4 time units. The network was designed to store eight events ($N = 8$), or a spike train of $7 \cdot (0.4) = 2.8$ time units. Notice that with this type of memory, the initial time between input spikes may be slightly altered by the synchronization procedure, but then the time between spikes is accurately preserved within the memorized pattern. Of course, the network may be designed for a faster sampling rate (time between spikes from the pace neuron) to minimize the initial errors. Conversely, an unsynchronized memory network can be used, but since the input and control spikes can occur at any time relative to each other, the transition regions in controlling the presynapses can cause errors in erasing old memories and storing new patterns; i.e., input and feedback spikes may occur before the synapses are turned completely on or off. Also, an unsynchronized network is unable to accurately preserve the timing between the spikes in memory; consequently, the timing error becomes significant the longer the pattern is stored.

The parameters for all of the other neurons were: $\Theta_i = 0.5$, $R_{i0} = 0.0$, and $\tau_i = 0.08$, which are the same as the parameters used for the synchronized identity neuron in Figure 6.19. Likewise, the parameters from Figure 6.19 were used to set the

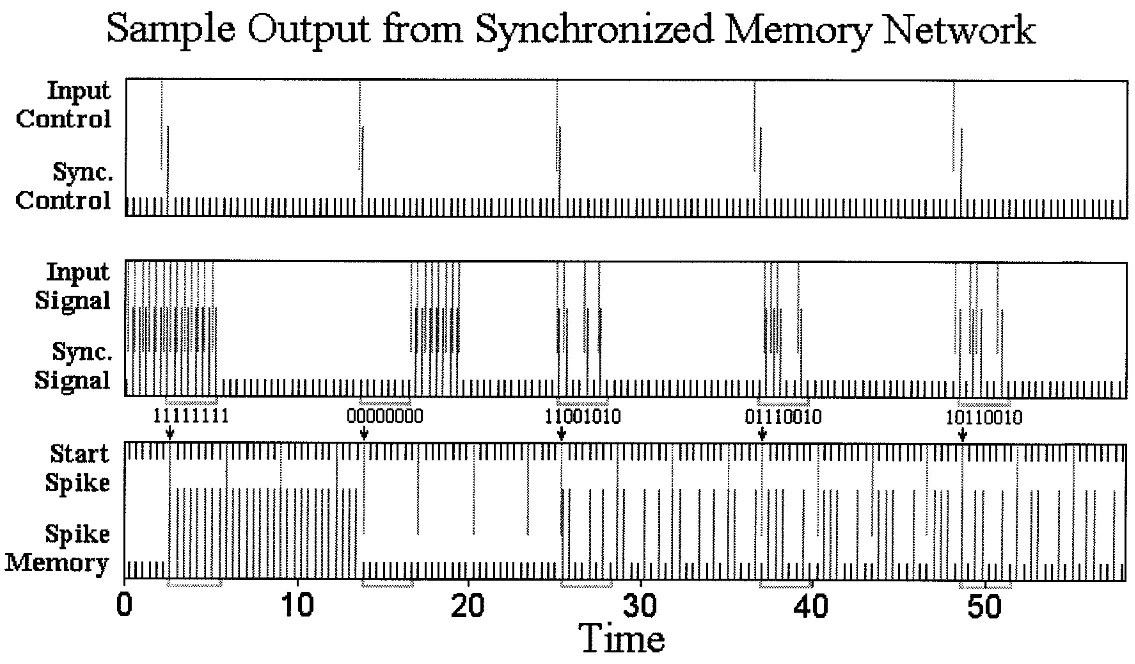


Figure 7.15: **Sample Output for Synchronized Memory Network.** The plot shows the results when five different input spike patterns are stored. The top box shows the control spikes and the output from the synchronized control neuron. The middle box shows the input spikes and the output from the synchronized input neuron. The numbers underneath indicate the binary code for the first eight events after a control spike. These are the spikes which are to be stored in memory. The bottom box shows the output spikes from the start spike neuron and the memory neuron. Arrows designate the start times of new memories, while the hash marks indicate the times of the corresponding pseudospikes for each of the neurons. (Since the input from the pace neuron is slightly delayed into the start and memory neurons, their pseudospikes do not exactly align with the pseudospikes for the synchronized control or input neurons.) Initially, the network was receiving input spikes at the maximum possible rate, but without a control spike, no output spikes were produced. Only after the first control spike were there any output spikes. Even when the input spikes stopped immediately after the last memory event ($8^{\text{th}}$ event after the control spike), the memory neuron continued to produce output spikes at every pseudospike. When the next control spike arrived, there were no input spikes until nine pseudospikes later. Thus, these late input spikes had no effect on the neuron, and the stored pattern was all non-spike events. The next three control spikes stored the patterns of "11001010," "01110010," and "10110010." Notice that these are the binary numbers for the values of 83, 78, and 77 (first bit is least significant) which representation the ASCII characters of "SNM." The network was able to correctly store all five test patterns, and ignored any input spikes that occurred outside of the sensitivity time windows, following each control spike.

synaptic parameters on the input and control identity neurons; i.e.: $R_{i1} = 0.02525$, $T_{i1} = 0.292$ and $\tau_{i1} = 0.2$ for the input signal; $R_{ip} = 1.0$, $T_{ip} = 1.4$, and $\tau_{ip} = 0.05$ for the pace signal.

The synaptic parameters of the memory neurons were: $R_{i1} = 0.21$, $T_{i1} = 0.21$, $\tau_{i1} = 0.0694$, and $d_{i1} = 0.1306$ for the synchronized input signal; $R_{i2} = 0.33$, $T_{i2} = 0.33$, $\tau_{i2} = 0.0848$, and $d_{i2} = 3.1152$ for the feedback signal; and $R_{ip} = 1.136$, $T_{ip} = 1.136$, $\tau_{ip} = 0.05$, and $d_{ip} = 0.2$ for the pace signal. Notice that with these small time constants, each synchronized spike acts as an isolated event, with no significant accumulation of neurotransmitter possible. Because of this, the maximum neurotransmitter concentration from each spike is approximately $\mathrm{e}^{-1}$. And when $R_{ij} = T_{ij}$, the parameters must be chosen so that:

$$R \cdot \tanh\left(\mathrm{e}^{-1}\right) + R_p \cdot \tanh\left(\mathrm{e}^{-1}\right) \geq \Theta_i$$

or:

$$\tfrac{\Theta_i}{R+R_p} \leq \tanh\left(\mathrm{e}^{-1}\right) \approx 0.3521 \qquad (7.19)$$

Only if this equation is satisfied can an input spike produce an output spike. Thus, for the parameters used, *when there was no significant effect from the presynapses*, the feedback synapse's weight values caused an output spike to be produced for each feedback spike, while the input synapse's weight values were unable to produce output spikes as a result of any input spikes. The synaptic parameters for the start spike neuron were the same as for the memory neuron except on the input synapse, whose parameters were: $R_{i1} = 0.33$, $T_{i1} = 0.33$, $\tau_{i1} = 0.05$, and $d_{i1} = 0.15$, which allowed all synchronized control spikes to produce output spikes.

The parameters of the excitatory presynaptic connection onto the input signal's synapse were: $R_{i2} = 3.0$, $T_{i2} = 3.0$ and $\tau_{i2} = 0.9$. These values allowed input spikes to create new output spikes during the next eight possible events after a control spike. The parameters of the inhibitory presynaptic connection onto the feedback synapses were: $R_{i3} = 3.0$, $T_{i3} = -3.0$ and $\tau_{i3} = 0.75$. These values prevented any spikes previously in memory from producing output spikes during the next eight possible events after a control spike occured.

## 7.6 Counters

### 7.6.1 Introduction

For some computing problems, it is useful to have a network which can count the number of incoming spikes. This section presents two such networks. Both networks encode the total number of spikes via memory oscillators. Section 7.6.2 describes a linear counter, in which the number of oscillating neurons increases with each input spike. It can not count higher than the number of neurons it has in the network, and once that number is reached, additional input spikes have no effect on the network until it receives a reset signal. Section 7.6.3 describes a sequential counter, in which the *next* neuron starts oscillating with each input spike, and all other neurons are silent. When the last neuron in the network is oscillating and another input spike arrives, the first neuron starts oscillating and the cycle repeats. By cascading several sequential counters together, large numbers can be encoded with relatively few neurons, with each counter encoding the value for one digit.

### 7.6.2 Linear Counters

A linear counter network uses two inputs. One is a reset signal which stops all network oscillations. The other is the input signal, which contains the spikes to be counted. Each input spike causes another neuron in the network to start oscillating. Thus, the total number of oscillating neurons in the network represents the number of input spikes since the last reset spike. Obviously, the network can not count higher than the number of neurons it contains. Figure 7.16 shows a linear counter network which contains five neurons.

Each neuron in the linear counter network has a similar behavior to the Dual Control Memory Neuron of Section 6.3.2. The reset input acts as the "OFF" input, while the feedback and the input signal act as the "ON" inputs. Each neuron (except

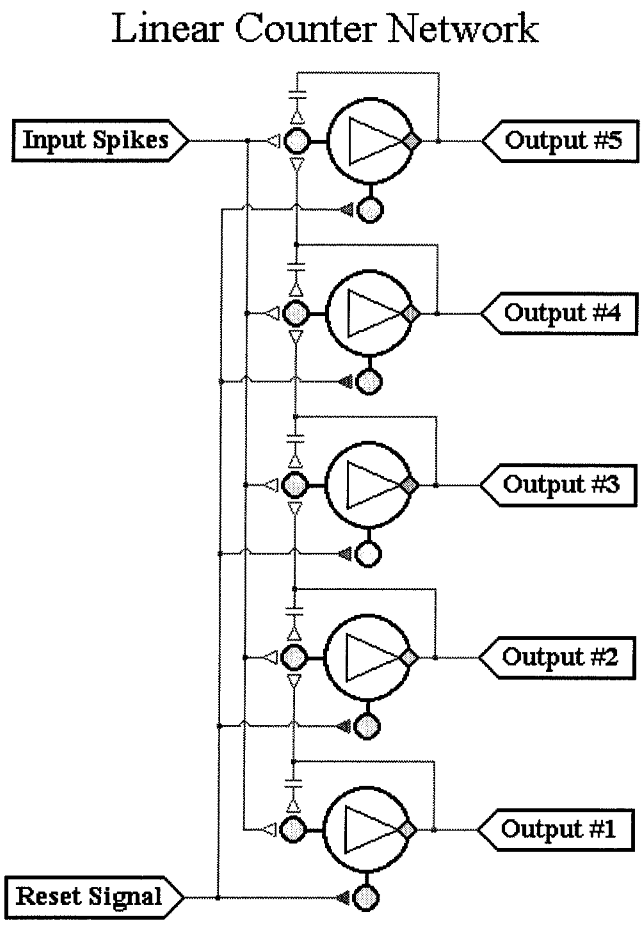


Figure 7.16: **Linear Counter Network.** The plot shows the connectivity of a linear counter. The network is composed of a chain of Dual Control Memory Neurons. The reset signal inhibits the neurons and stops all oscillations. Each neuron (except the first neuron) has three excitatory connections. The first is from the input signal. The second is the feedback signal required for oscillation. And the third is from the previous memory neuron. Each neuron is able to sustain an oscillation once it starts producing output spikes, but it is incapable of producing any output spikes unless it receives spikes from both the input signal *and* the previous neuron.

the first neuron) has three excitatory connections. The first connection is from the input signal. The second connection is the feedback signal required for oscillation. And the third connection is from the previous memory neuron. The weights are chosen so that each neuron is able to sustain an oscillation once it starts producing output spikes, but it is incapable of producing any output spikes unless it receives a spike from both the input signal *and* the previous neuron. Thus, with each input spike, the next neuron in the chain starts oscillating. (The first neuron starts oscillating with the initial input spike.)

Figure 7.17 shows a sample output for the five neuron linear counter of Figure 7.16. When the network receives a reset spike, all of the neurons stop oscillating. Notice that after the second reset spike, the first input spike to arrive is not counted. This is because there is a "dead time" associated with each reset, in which all of the neurons are insensitive to input spikes. The parameters for all of the neurons were: $\Theta_i = 0.5$, $R_{i0} = 0.0$, and $\tau_i = 0.2$. The reset input's synaptic parameters were: $R_{i1} = 0.10$, $T_{i1} = -0.10$, and $\tau_{i1} = 1.0$. The excitatory synaptic cluster's weight value was $R_{i2} = 0.53$, and the excitatory connection parameters were: $T_{i21} = 2.0$ and $\tau_{i21} = 0.375$ for the input signal; $T_{i22} = 2.75$, $\tau_{i22} = 0.375$ and $d_{i22} = 0.2$ for the feedback signal; and $T_{i23} = 0.265$ and $\tau_{i23} = 1.0$ for the input from the previous neuron. (For the first neuron, $T_{i21} = 2.75$ and $\tau_{i21} = 0.375$ for the input signal, and there was no connection from a previous neuron.)

### 7.6.3 Sequential Counters

The sequential counter network is very similar to the linear counter network, but rather than just having the next neuron in the network start oscillating with each input spike, the previous neuron stops oscillating. Thus, only one neuron oscillates at a time.

Like the linear counter, the network uses two inputs. One is a reset signal which stops all network oscillations, and the other is an input signal, which contains the spikes to be counted. Each input spike causes the next neuron in the network to start

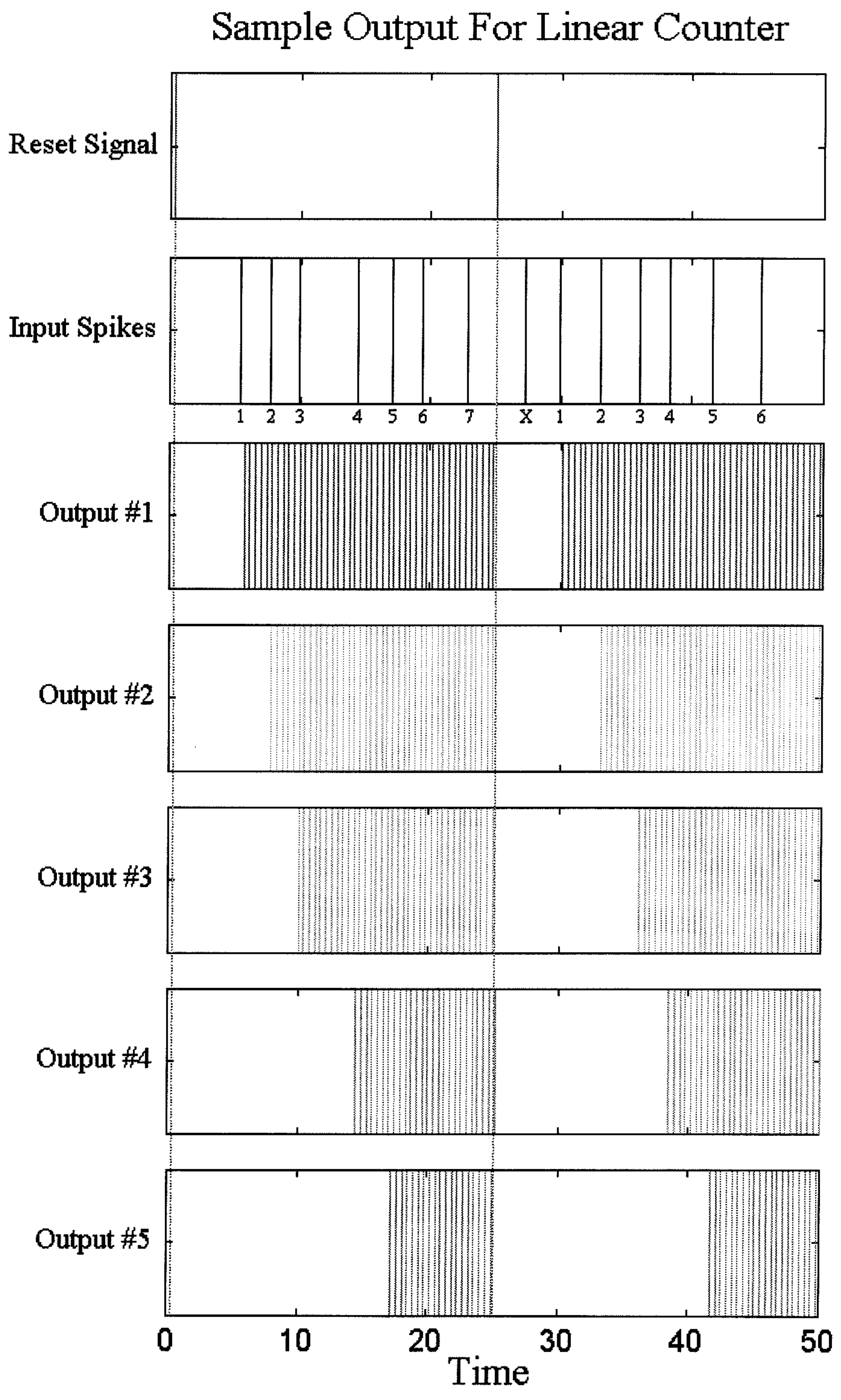


Figure 7.17: **Sample Output for Linear Counter.** The plot shows a set of random input spikes for a five neuron linear counter. The network receives a reset signal at $t = 0.1$ and $t = 25$. With each input spike the next neuron starts oscillating, until all are oscillating. Additional input spikes do not affect the network.

oscillating. Figure 7.18 shows a sequential counter network with five neurons.

Like the linear counter, each neuron (except the first neuron) has three excitatory connections. The first excitatory connection is from the input signal. The second excitatory connection is the feedback signal required for oscillation. And the third excitatory connection is from the previous memory neuron. The weights are chosen so that each neuron is able to sustain an oscillation once it starts producing output spikes, but it is incapable of producing any output spikes unless it receives a spike from both the input signal *and* the previous neuron. Thus, with each input spike, the next neuron in the chain starts oscillating. However, each neuron has a delayed inhibitory connection from the output signal of the next neuron. Therefore, once that neuron starts oscillating, it then turns off the neuron whose oscillations allowed it to get started.

The first neuron does not receive an excitatory input signal from any of the other neurons but instead receives an inhibitory signal from all but the last neuron. When none of the neurons in the network are oscillating, it is uninhibited and an initial input spike causes it to start oscillating. (The weights from the input signal are larger for the first neuron than the others, which also require the previous neuron to be oscillating.) The next input spike causes the second neuron to start oscillating, and its inhibitory input into the first neuron stops it from oscillating. Each subsequent input spike causes the next neuron to start oscillating, but since all of the neurons inhibit the first, it is unable to start oscillating. When the last neuron in the network starts oscillating, the first neuron no longer receives an inhibitory signal, and the next input spike causes it to start oscillating again. Its output inhibits the last neuron, which then stops oscillating, leaving only the first neuron to produce spikes. With each additional input spike, the next neuron starts oscillating and the cycle repeats until a reset spike is received and all of the neurons stop oscillating.

Figure 7.19 shows a sample output for the five neuron sequential counter of Figure 7.18. Notice that after the second reset spike, the first input spike to arrive is not counted. This is because there is a "dead time" associated with each reset, in which all of the neurons are insensitive to input spikes. The parameters for all of the neurons

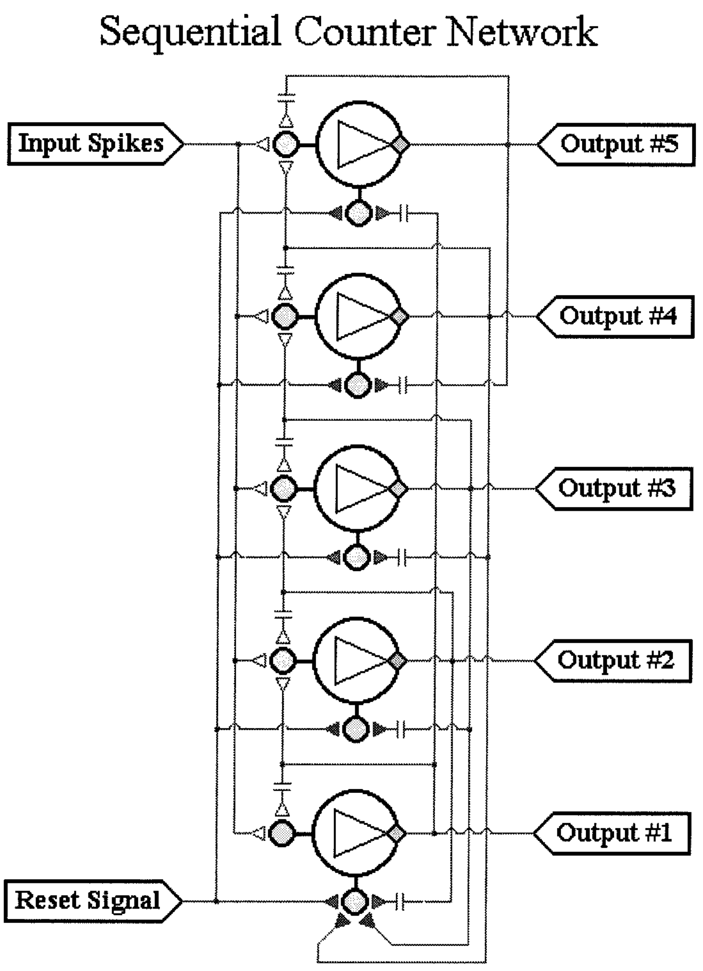


Figure 7.18: **Sequential Counter Network.** The plot shows the connectivity of a sequential counter. The network is composed of a chain of Dual Control Memory Neurons. The reset signal inhibits the neurons and stops all oscillations. Each neuron (except the first neuron) has three excitatory and two inhibitory connections. The first excitatory connection is from the input signal. The second excitatory connection is the feedback signal required for oscillation. And the third excitatory connection is from the previous memory neuron. Each neuron is able to sustain oscillation once it starts producing output spikes, but it is incapable of producing any output spikes unless it receives spikes from both the input signal *and* the previous neuron. The first inhibitory connection is from the reset signal. The second inhibitory connection is the delayed output from the *next* memory neuron. Therefore, once the next neuron starts oscillating, the oscillation in the previous neuron stops.

were: $\Theta_i = 0.5$, $R_{i0} = 0.0$, and $\tau_i = 0.2$. The inhibitory synaptic cluster's weight value was $R_{i1} = 0.10$, and the inhibitory connection parameters were: $T_{i11} = -0.10$ and $\tau_{i11} = 1.0$ for the reset signal; and $T_{i12} = -0.10$, $\tau_{i12} = 1.0$ and $d_{i12} = 0.2$ for the input from the next neuron. (For the first neuron, $T_{i1j} = -0.10$ and $\tau_{i1j} = 0.2$ for the inputs from neurons #3 and #4.) The excitatory synaptic cluster's weight value was $R_{i2} = 0.53$, and the excitatory connection parameters were: $T_{i21} = 2.0$ and $\tau_{i21} = 0.375$ for the input signal; $T_{i22} = 2.75$, $\tau_{i22} = 0.375$ and $d_{i22} = 0.2$ for the feedback signal; and $T_{i23} = 0.265$ and $\tau_{i23} = 1.0$ for the input from the previous oscillator. (For the first neuron, $T_{i21} = 2.75$ and $\tau_{i21} = 0.375$ for the input signal, and there was no excitatory connection from a previous neuron.)

With the linear counter, the network could not count higher than the total number of neurons it had, and once this limit was reached, additional input spikes had no effect on the network. For the sequential counter, no input spikes are lost, since the network continues to cycle through its neurons. Because of this, it is possible to use several cascaded sequential counter networks to encode totals containing several digits. The number of neurons in each counter determines the base unit for the counting system; e.g., for a decimal system (base 10) each counter requires 10 neurons to represent the digits of 0 to 9. When the last neuron in the network is oscillating and an additional input spike arrives, the first neuron starts oscillating and a carry spikes is generated, which becomes the input signal into the next counter.

Figure 7.20 shows an example of how sequential counters may be cascaded together to represent two digits. Since each of the sequential counting networks has only 5 neurons, this network encodes the total number of spikes using a base 5 counting system. The first counter represents the $5^0$ digit, and the second counter represents the $5^1$ digit. Additional counters may be cascaded together to represent digits for higher powers of 5. Notice that the first neuron no longer represents "1," as it did in the sequential counter of Figure 7.18, but instead represents "0."

Figure 7.21 shows a sample output from the base 5 counter of Figure 7.20. All of the parameters for the neurons in each of the two sequential counters were the same as those used for Figure 7.19, except for the first neuron's synaptic parameters. All of

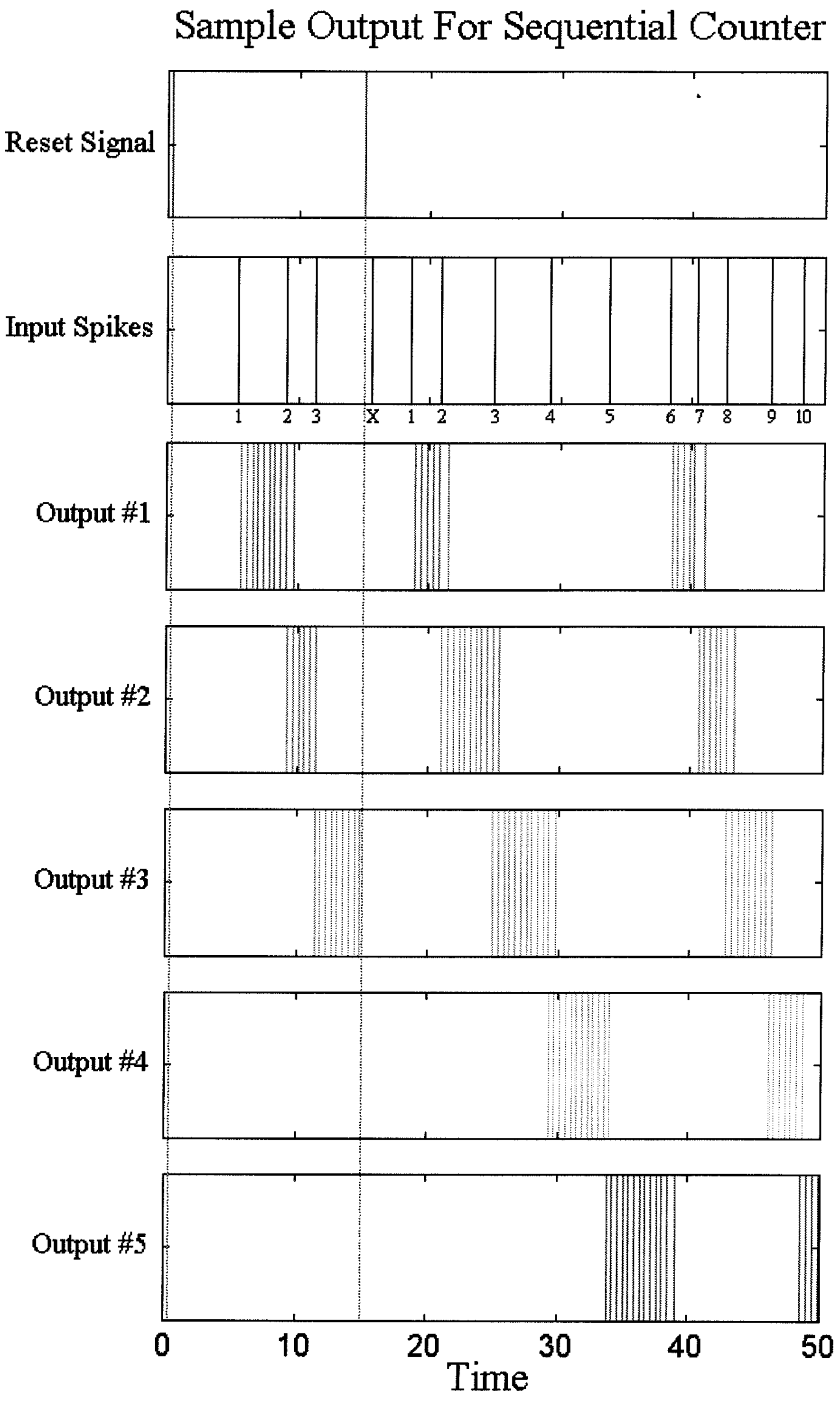


Figure 7.19: **Sample Output for Sequential Counter.** The plot shows a set of random input spikes for a five neuron sequential counter. The network receives a reset signal at $t = 0.1$ and $t = 15$. With each input spike the next neuron starts oscillating, until the last neuron is oscillating and the cycle repeats itself.

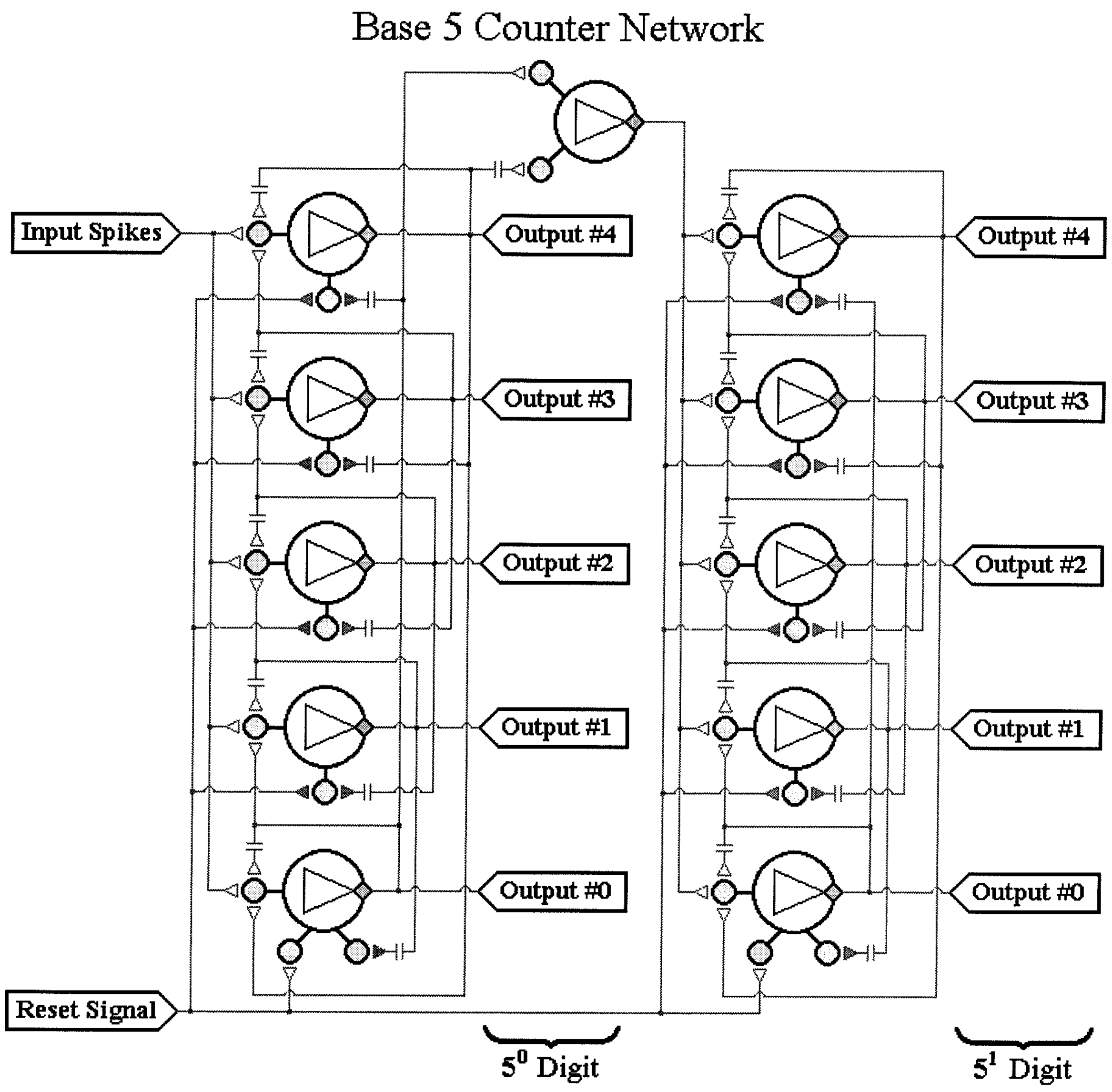


Figure 7.20: **Base 5 Counter Network.** The diagram shows the connectivity of a base 5 counter network, which consists of cascaded sequential counters with 5 neurons each. Each counter operates independently of the others, but a "carry bit" is produced when both its first neuron and the delayed output from the last neuron are oscillating. This is the case only when the last neuron is oscillating and another input spike arrives, causing the counter to repeat the cycle. The carry spike is then used as the input signal to the next counter in the cascade. Except for the first neuron, all of the neurons in each counter are connected exactly as they were for the sequential counter of Figure 7.18. But now the first neuron represents "0" instead of "1"; consequently, the reset signal causes the first neurons in each counter to oscillate while still inhibiting all of the other neurons from oscillating.

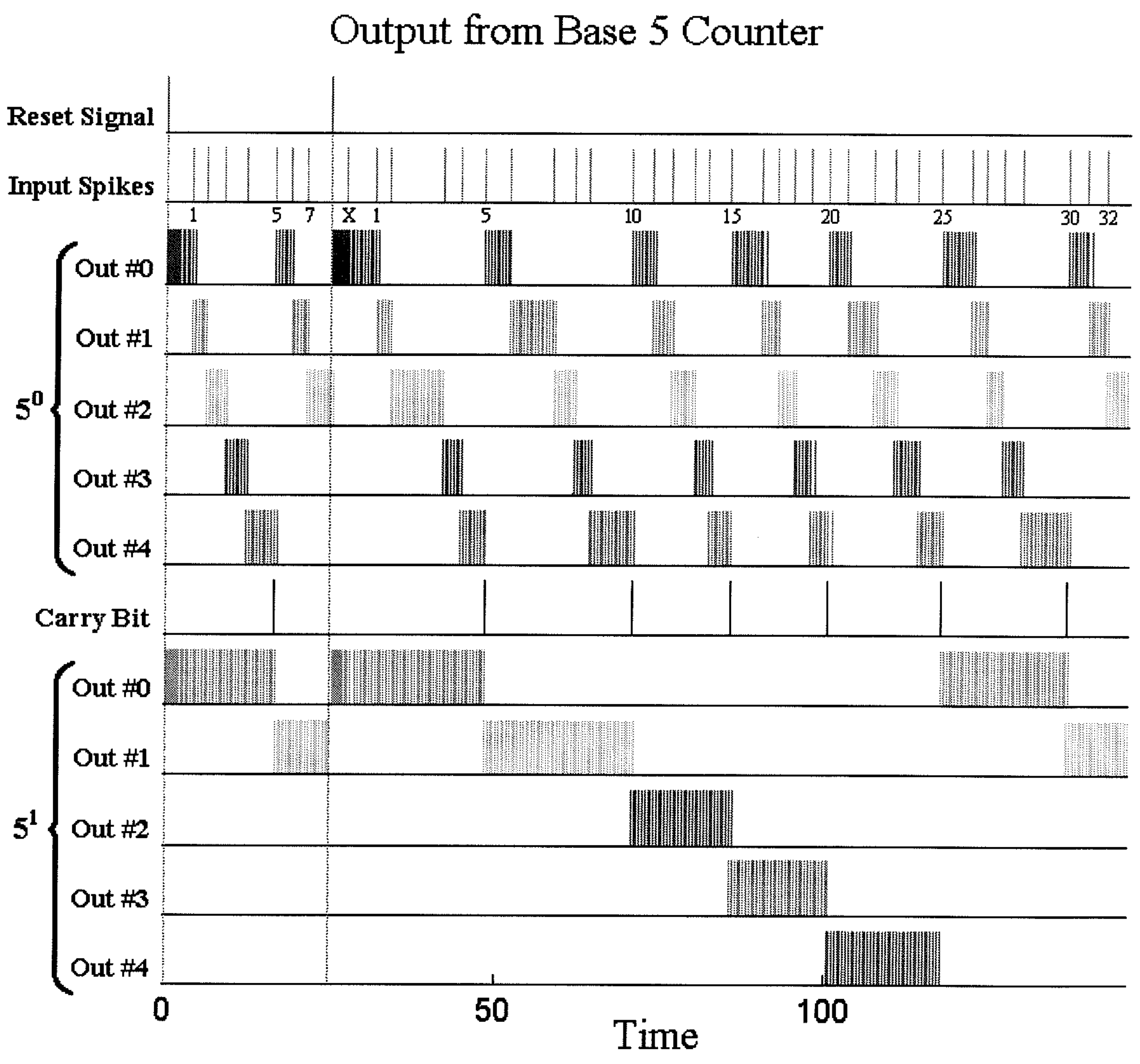


Figure 7.21: **Sample Output for Base 5 Counter Network.** The plot shows the output spikes from the two digit, base 5 counter of Figure 7.20. The input spikes being counted are random. The network receives a reset signal at $t = 0.1$ and $t = 25$. The reset spikes cause the first neurons ($0^{\text{th}}$ numbers) in each sequential counter (digit) to start oscillating. Notice that after the second reset spike, the first input spike to arrive is not counted. This is because there is a "dead time" associated with each reset, in which the neurons are insensitive to input spikes. With each input spike the next neuron starts oscillating, until the last neuron is oscillating and the cycle repeats itself, but not before producing a carry bit, which is the input signal for the next sequential counter.

its parameters were the same as the other neurons in the counter network except that its reset signal was excitatory, with the parameters being: $R_{i3} = 0.70$, $T_{i3} = 2.50$ and $\tau_{i3} = 0.50$. Also, the parameters for the neuron which produced the carry bit were: $\Theta_i = 0.5$, $R_{i0} = 0.0$, and $\tau_i = 1.0$; and its synaptic parameters were: $R_{ij} = 0.27$, $T_{ij} = 2.00$ and $\tau_{ij} = 0.20$ for both synapses, but the connection from the last neuron had a delay of $d_{i2} = 0.2$.

## 7.7 Multiplexers

A multiplexer network selects one of many inputs to be transferred to a single output. Multiplexers can be used to interleave data from several sources. One of its possible uses is to route sensory data to a pattern recognizer network. For example, if the system is to recognize a particular object in its visual field, it is not necessary to have multiple copies of the pattern recognizing network associated with every possible location in the visual field. Instead, the spike trains from all locations can be alternately routed to the pattern recognizing network to search for the desired pattern in the entire field.

Figure 7.22 shows a multiplexer with three input signals and one output signal. To control which signals are passed to the output, each input must first go through a passing neuron, whose sole function is to reproduce the input spikes if that signal has been selected. An input signal is selected when the corresponding controlling oscillator, a dual control memory neuron, is producing spikes. (A single control memory neuron may also be used to generate the necessary spikes.) The passing neurons work like *unsynchronized* identity neurons, but with negative resting potentials, ($R_{i0} < 0$). The excitatory input spikes from the oscillating memory neuron boost the neuron's potential enough to compensate for the negative bias, but not enough to produce any output spikes. The memory neuron's feedback parameters determine its oscillation frequency. Thus, when the corresponding memory neuron is oscillating, the passing neuron reproduces the input spikes. The parameters are chosen so that the contribution from the controlling oscillator remains relatively constant (no pseudospikes).

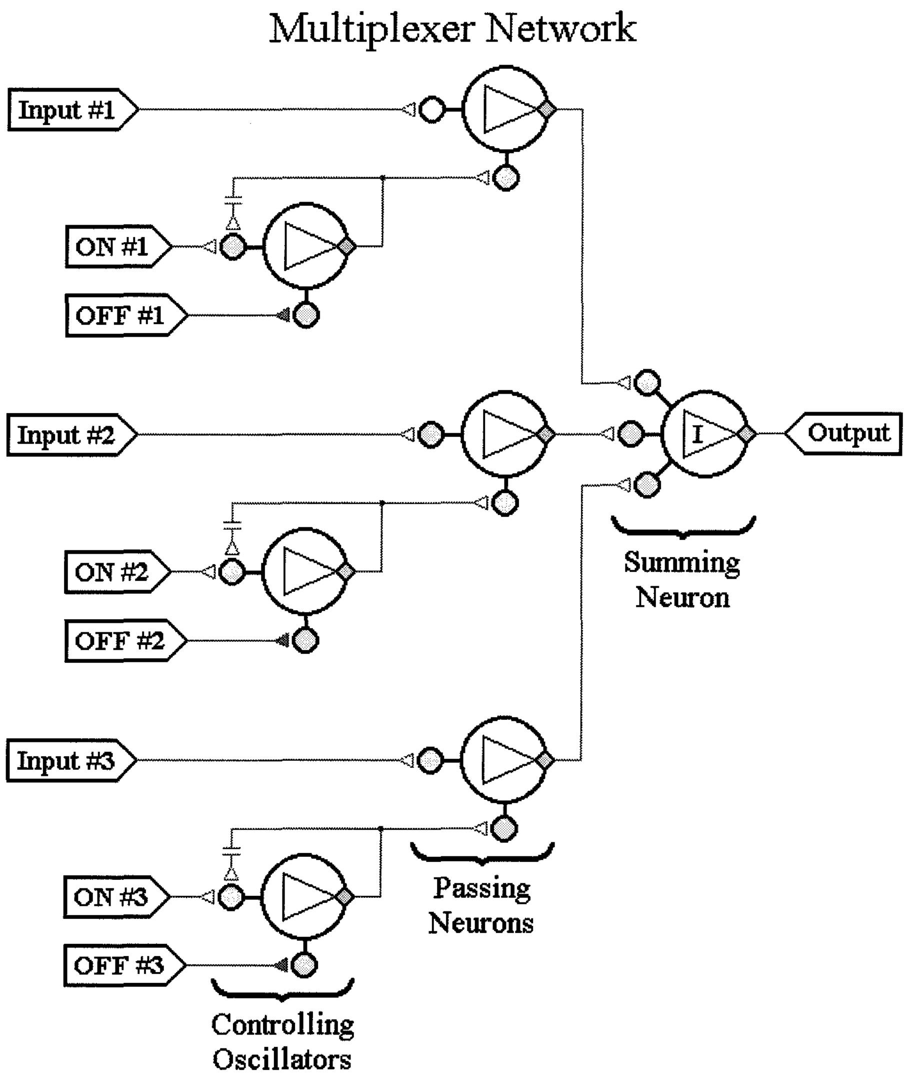


Figure 7.22: **Multiplexer Network.** The plot shows the connectivity of a three input multiplexer network. If more than one input signal is selected, the output is the sum of the selected spike trains. Each input connects to a passing neuron, which only reproduces the spikes when that signal has been selected. An input is selected when the corresponding controlling oscillator is producing spikes. The output spikes from all of the passing neurons go into a single identity neuron which sums the spike trains together.

This allows the timing of the output spikes to depend upon the input spikes and remain independent of the controlling spikes.

In the network shown, it is possible for several of the memory neurons to oscillate. In this case, the output spike train is the sum of all input spike trains whose corresponding memory neurons are oscillating. But since the summing neuron is also an unsynchronized identity neuron, it is possible for some of the input spikes to be lost when spikes from several of the summed inputs are closely correlated in time (see Section 6.5.2).

Figure 7.23 shows a sample output for the multiplexer of Figure 7.22. The parameters for the controlling oscillators (dual control memory neurons) were: $\Theta_i = 0.5$, $R_{i0} = 0.0$, and $\tau_i = 0.2$; the parameters for the passing neurons were: $\Theta_i = 0.5$, $R_{i0} = -0.2$, and $\tau_i = 0.42$; and the parameters for the summing neuron were: $\Theta_i = 0.5$, $R_{i0} = 0.0$, and $\tau_i = 0.42$. For the controlling oscillators, the excitatory connection parameters were: $R_{i1} = 0.53$, $T_{i1k} = 2.75$ and $\tau_{i1k} = 0.375$ for both the "ON" inputs and feedback connections, but the feedback connections also had a delay of $d_{i12} = 0.2$. The parameters for the inhibitory "OFF" inputs were: $R_{i2} = 0.1$, $T_{i2} = 0.1$ and $\tau_{i2} = 1.0$. When these memory neurons receive an "ON" spike, they start oscillating, with the time between output spikes converging to 0.425. For the passing neurons, the synaptic parameters for the input connections were: $R_{i1} = 1.0$, $T_{i1} = 1.5$ and $\tau_{i1} = 0.333$; and the synaptic parameters for the oscillator inputs were: $R_{i2} = 0.21$, $T_{i2} = 0.35$ and $\tau_{i2} = 0.5$.[8] The excitatory synapses of the summing neuron used the same parameters as the input signals into the passing neurons, allowing it to behave as an unsynchronized identity neuron as well.

When a memory neuron first starts oscillating, its contribution to the passing neuron has not yet reached its asymptotic level. Consequently, input spikes arriving immediately after the controlling oscillations start are not passed through to the

[8]When the memory neuron is oscillating, its maximum contribution to the passing neuron's potential is 0.203, and its minimum contribution is 0.200. (These values can be calculated from Equations (4.25), (4.26), and (3.1).) Thus, it is able to compensate for $R_{i0} = -0.2$. The parameters for the input connections were the same as those used for the unsynchronized identity neuron of Figures 6.12-6.15, but with the time scale reduce by a factor of 3.

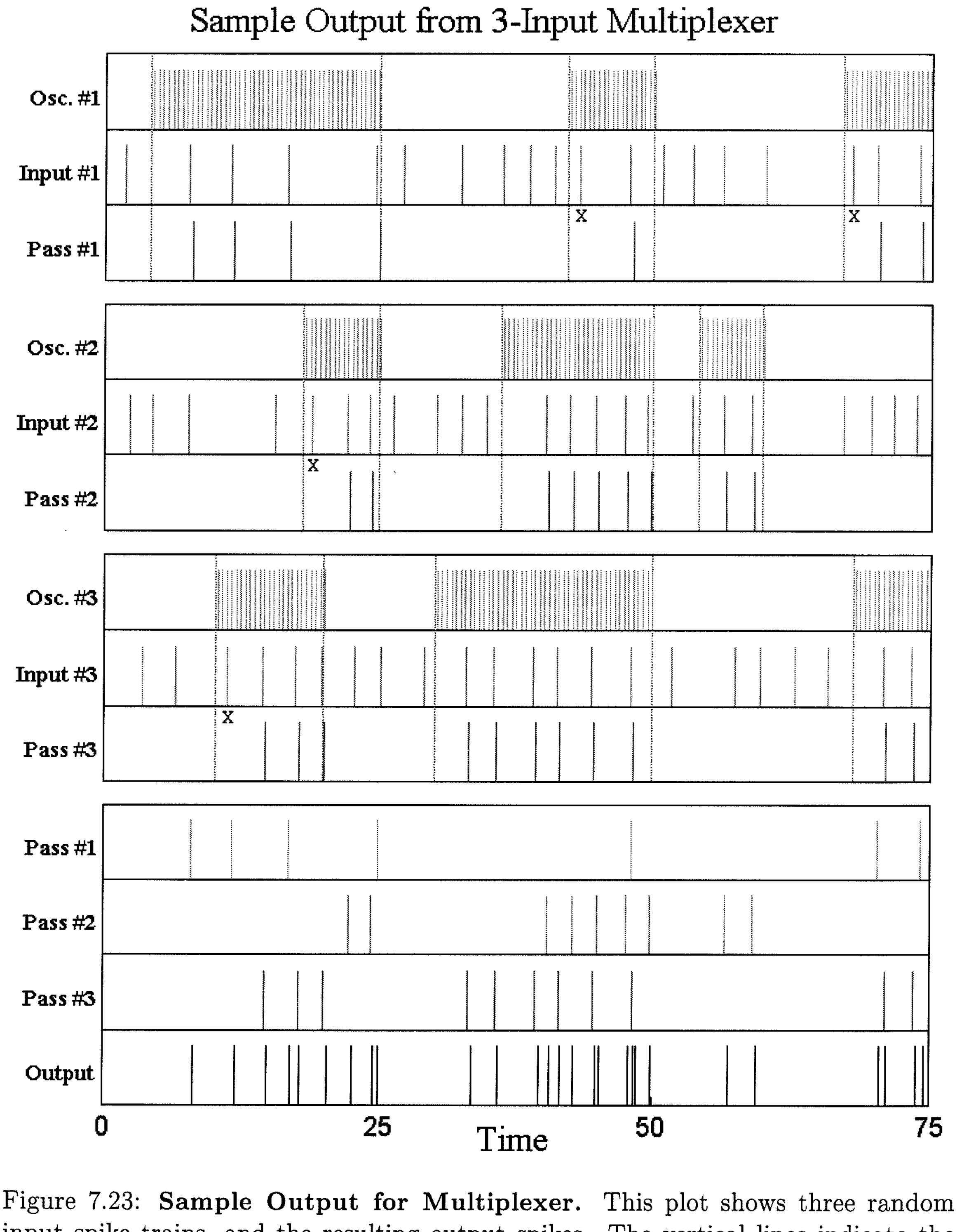


Figure 7.23: **Sample Output for Multiplexer.** This plot shows three random input spike trains, and the resulting output spikes. The vertical lines indicate the times of the "ON/OFF" input spikes to the controlling oscillators. Only the input spikes that occurred during the oscillation state were passed to the summing neuron. The "X" under some input spikes indicate that they were not passed because they occurred too close to the onset of the oscillation. The last box repeats the output spikes from the passing neurons and shows the network's output, which is their sum.

summing neuron. For the example shown, all of the spikes that actually passed to the summing neuron produced an output spike. While both the passing neurons and the summing neuron introduced delay into the system, the relative timing between spikes was fairly well preserved. See Section 6.5.2 for details on the timing between input and output spikes from an unsynchronized identity neuron. In general, stronger input stimuli (those with closely space spikes) have less delay than weak input stimuli. For the sample output shown in Figure 7.23 the minimum delay between an input and output spike was 0.39 and the maximum delay was 0.68.

While the multiplexer shown in Figure 7.22 represents one configuration, other variations are possible. Rather than use an independently controlled memory neuron for each input signal, it is possible to use a sequential counter as shown in Figure 7.18 to control all of the inputs. In such a case, only one input is passed at a time, and the selected input is chosen by cycling through the different controlling oscillators using the appropriate number of spikes into the counter. Also, the summing identity neuron can be replaced by a synchronized identity neuron, connected to a pace neuron. In this case, the output spikes are synchronized.

Although not shown, it is trivial to construct a demultiplexer from a multiplexer. A demultiplexer routes one input to one of several possible outputs. The summing neuron can be removed from the multiplexer network of Figure 7.22 and all of the inputs connected together to form a single input. The passing neurons then become the many possible output signals, and the controlling oscillators determine which of the outputs receive a copy of the single input spike train.

## 7.8 Comparators

Previously, Section 7.3 presented several networks for recognizing specific spike patterns; however, it is sometimes necessary to determine if two or more spike trains are equal to each other. A comparator network can be used for such a task.

Figure 7.24 shows a network for comparing three bits from two input spike trains. The network requires the input spike trains to be synchronized with each other, so

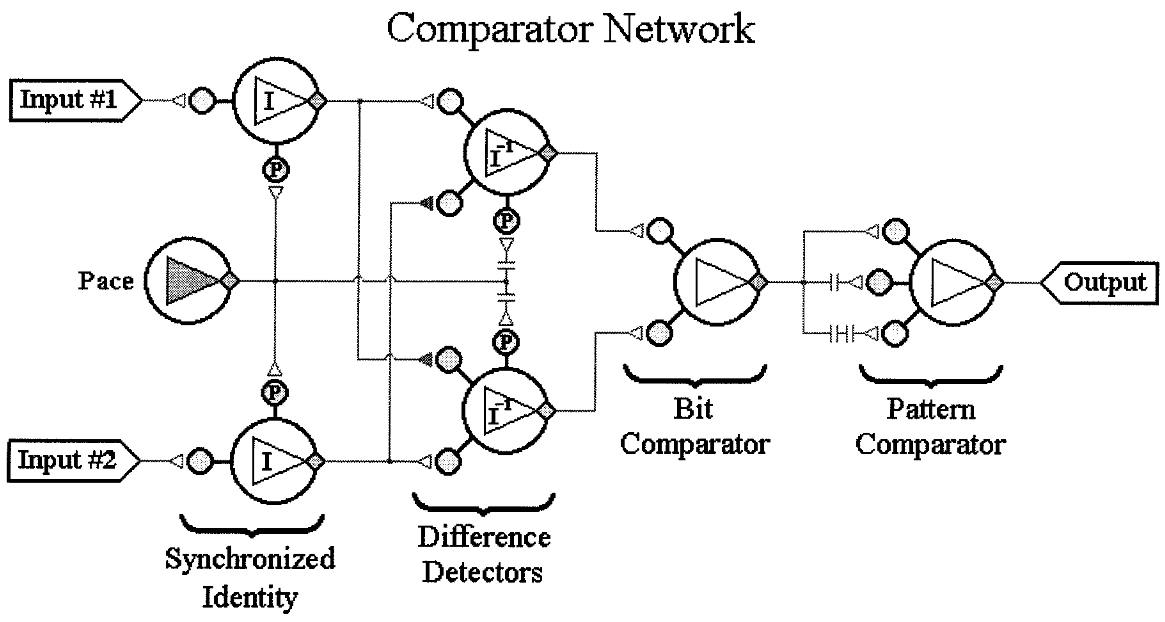


Figure 7.24: **Comparator Network.** The plot shows the connectivity of a network for comparing three bits from two input spike trains. The input spike trains first go through the synchronizing identity neurons before connecting to the inverse neurons. Each inverse neuron subtracts one of the input spike trains from the other, and outputs a spike if its net input is positive or zero. Only when both spikes trains have equal input bits do both inverse neurons produce output spikes. Their outputs connect to a bit comparator neuron, which acts as an "AND" gate and only produces a spike when all of its inputs are spiking. The final neuron is the pattern comparator. It only fires when all of its input synapses are receiving spikes. Since each of its inputs is a delayed copy of the output from the bit comparator neuron, the last three input events must match.

each of the input signals first go through a synchronized identity neuron. After becoming synchronized, both spike trains connect to two inverse neurons which act as difference detectors. The inverse neurons use the same pace neuron as the identity neurons, but its input is slightly delayed so that the maximum contributions from the output spikes of the identity neurons occur at the times of the pseudospikes in the inverse neurons. In addition to the signal from the pace neuron, the inverse neurons receive a connection from each input. One is excitatory while the other is inhibitory. This allows each inverse neuron to subtract one of the input spike trains from the other. The parameters are chosen so that the inverse neuron outputs a spike if its net input is positive or zero. The first difference detector outputs a spike at each pseudospike unless the first input does not have a spike while the second input does. The second difference detector outputs a spike at each pseudospike unless the second input does not have a spike while the first input does. Thus, only if both spikes trains have equal input bits do both inverse neurons produce output spikes.

More input signals can be compared by increasing the number of difference detector neurons. Two additional inverse neurons are needed for each extra input; e.g., for 4 input signals, 6 neurons are needed to calculate: #1-#2, #2-#1; #2-#3, #3-#2; #3-#4, #4-#3. Notice that when the difference detectors are connected in this manner, all possible combinations of signals are not actually compared. But obviously, if and only if #1 = #2, #2 = #3, #3 = #4 can *all* of the input spike trains equal each other.

The outputs from the difference detector neurons connect to a single bit comparator neuron which only produces spikes when all of its inputs are spiking. It acts like an "AND" gate, since the last events on both input spike trains must match for there to be an output spike.

The bit comparator neuron then connects to a pattern comparator neuron through multiple synapses. Like the bit comparator, the pattern comparator produces an output spike only when all of its input synapses receive a spike. Its number of input synapses equals the number of bits being compared. The delays associated with each synapse are set so that the effects from the last three events are aligned. Longer patterns may be compared by using more synapses. The threshold on the pattern comparator can be set so that the patterns need not match exactly to produce an output spike. The refractory time constant can be set so that successive output spikes are not produced, thus eliminating comparisons between overlapping patterns. (See discussion of overlapping patterns on page 151.)

Figure 7.25 shows the output from the comparator network of Figure 7.24 for two random input spike trains. The pace neuron's parameters were: $\Theta_i = 0.5$, $R_{i0} = 0.538$, and $\tau_i = 0.5$, which produced a time interval between output spikes of 1.0. The parameters for all of the other neurons in the network were: $\Theta_i = 0.5$, $R_{i0} = 0.0$, and $\tau_i = 0.2$. The synchronized identity neurons' synaptic parameters were: $R_{ip} = 1.0$, $T_{ip} = 1.404$, $\tau_{ip} = 0.05$, $R_{ij} = 0.0253$, $T_{ij} = 0.292$, and $\tau_{ij} = 0.5$, which were chosen using the method described in Section 6.5.3. The inverse identity neurons' synaptic parameters were: $R_{ip} = 1.0$, $T_{ip} = 1.58$, $\tau_{ip} = 0.05$, $d_{ip} = 0.15$, $R_{ij} = 0.03$, $T_{ij} = \pm 0.3$, and $\tau_{ij} = 0.15$. The bit comparator neuron's synaptic parameters were: $R_{ij} = 0.8$, $T_{ij} = 0.8$, and $\tau_{ij} = 0.15$. And the pattern comparator neuron's synaptic parameters were: $R_{ij} = 0.55$, $T_{ij} = 0.55$, $\tau_{ij} = 0.15$, and $d_{ij} = 0, 1, 2$.

The total length of the pattern comparison time window equals the number of bits being compared, multiplied by the period of the pace neuron, $\Delta_p$. Notice that due to possible timing differences between the input spikes and the pseudospikes of the identity neurons, it is possible for an input spike to cause the next pseudospike to become an actual spike, while a slightly later spike on a different input signal does not produce an actual spike until the following pseudospike. (Also, the synchronized identity neurons are not perfect at retaining a one-to-one correlation between their input and output spikes – see Section 6.5.3.) On average, two input spikes on different input lines will cause the same pseudospikes in the identity neurons to become actual spikes if they both occur within time $\frac{\Delta_p}{2}$ of each other. By reducing $\Delta_p$ and increasing the number of bits, the accuracy of the pattern match increases; i.e., the input spikes must occur nearly simultaneously for the output neuron to recognize the patterns as matching.

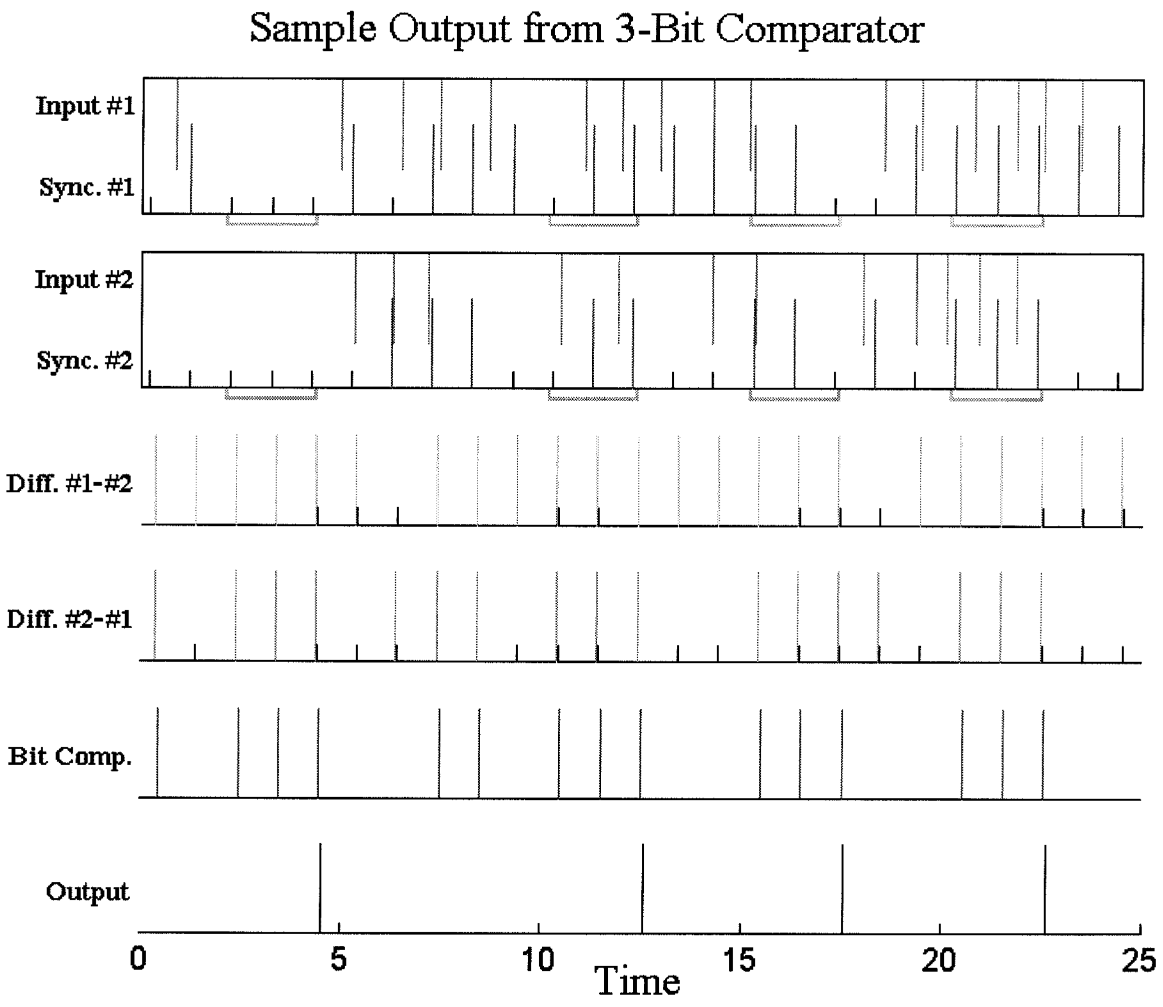


Figure 7.25: **Sample Output from 3-Bit Comparator.** This plot shows two sets of random input spike trains, and the resulting output spikes from each neuron of the comparator network shown in Figure 7.24. The top two boxes show the actual input spikes and the resulting output spikes from the synchronized identity neurons. The underlines indicate the times when three synchronized events matched. The hash marks indicate the times of the pseudospikes produced by the pace neuron. The next two lines show the output spikes from the difference detector neurons, which output spikes when their net input is positive or zero. The bit comparator only outputs spikes when both of the difference detectors are firing. Finally, the pattern detector outputs a spike when the bit comparator produces three consecutive spikes.

## 7.9 Hopfield Networks

This section describes how to use a network of spiking neurons as an associative memory. The basic problem is to store a set of $P$ patterns in such a way that when the network is presented with an input pattern, it responds by producing whichever one of the stored patterns most closely resembles the input signal [42, p. 11]. Thus, the stored patterns are attractors, and the dynamics of the system carry the starting point into one of the attractors. A common application for an associative memory is the recognition and reconstruction of images, in which an entire image can be correctly recalled from a partial image. The stored patterns can be taken to be either 0 or 1 at each neuron in the network, where 1 indicates that the neuron is firing, while 0 indicates that it is not. This type of network is usually referred to as a "Hopfield Network" (see [45]).

Hopfield used neurons based on the McCulloch-Pitts model [71], in which the output of each neuron was determined by:

$$S_i = \begin{cases} 1 & \text{if } \sum_j w_{ij} \cdot S_j > 0 \\ 0 & \text{if } \sum_j w_{ij} \cdot S_j \leq 0 \end{cases} \tag{7.20}$$

where $S_i$ represents the output state of the $i^{\text{th}}$ neuron. All of the neurons in the network are fully interconnected, with each neuron receiving inputs from all of the others.

With this type of neuron, to store just one pattern the weights can be chosen as:

$$w_{ij} = (2\xi_i - 1)(2\xi_j - 1) \tag{7.21}$$

where $\vec{\xi}$ represents the stored pattern with $\xi_i = 0$ or 1, and $w_{ij}$ represents the connection strength from the $j^{\text{th}}$ to the $i^{\text{th}}$ neuron. (Usually, the self-connection terms, $w_{ii}$, are set to zero.) Notice that the connection strength between two neurons is +1 if the neurons are correlated (both 1 or both 0) and -1 if the neurons are uncorrelated (one is 0 and the other is 1). This method for connecting weights is often referred to as

"Hebb's rule" because of its similarity with the hypothesis made by Hebb [42, p. 16], who suggested that synaptic strengths change according to the correlation between the firing of pre- and post-synaptic neurons [37]. Also notice that the weight matrix is symmetrical; i.e., $w_{ij} = w_{ji}$. Hopfield [45] showed that any network with symmetric weights will eventually converge to a stable state.

Now, when the network is started in state $\vec{S}$, the net input into each neuron is given by:

$$h_i \equiv \sum_j w_{ij} \cdot S_j = (2\xi_i - 1) \cdot \sum_j (2\xi_j - 1) \cdot S_j \tag{7.22}$$

If more than half of the neurons with $S_j = 1$ are initially correct, then this summation is positive, and the output state of a neuron is determined by the sign of $(2\xi_i - 1)$, which causes $S_i$ to become equal to $\xi_i$. Thus, the network converges to the stored pattern.

Notice that there are actually two attractors in this case; the other is the reversed state, where those neurons whose output should be at 1 are at 0 and those whose output should be at 0 are at 1; e.g., if 010011... is a stored memory, then 101100... is also a stable state. In general, the reversed state can be eliminated by removing the connections between neurons with a desired output of 0; i.e., let:

$$w_{ij} = 3\xi_i \cdot \xi_j - \xi_i - \xi_j = \begin{cases} +1 & \text{if } \xi_i = \xi_j = 1 \\ 0 & \text{if } \xi_i = \xi_j = 0 \\ -1 & \text{if } \xi_i \neq \xi_j \end{cases} \tag{7.23}$$

Now, neurons with a desired state of 0 only receive inhibitory inputs. With the previous weight selection method of Equation (7.21), the zero state (all 0's) was an unstable equilibrium point, but with this method, the zero state becomes one of the stable attractors. Notice that Equation (7.23) can be viewed as a correction to "Hebb's rule," since it is probably not physiologically reasonable for the weights between neurons to change positively when neither is firing [42, p. 16].

When there are several input patterns to be stored, the weights can be chosen as the superposition of the contributions from each pattern; i.e.:

$$W_{ij} = \sum_{p=1}^{P} w_{ij}^{p} \tag{7.24}$$

where $w_{ij}^{p}$ represents a weight associated with one of the $P$ memories, which is chosen using either Equation (7.22) or (7.23). However, adding memories can cause them to interfere with each other and become unstable. Also, spurious states are created, which do not correspond to any of the stored memories. The stability of the stored patterns limits the number of memories that can be recalled. For random patterns, the number of memories is approximately $0.138N$, where $N$ represents the total number of neurons in the network. (See [42, pp. 16-41] for detail on pattern stability, storage capacity, and spurious states.) If the stored patterns are correlated, the overlap between them becomes even more of a problem. When this occurs, a pseudo-inverse approach can be used to calculate the weights. But because inverse calculations require a knowledge of all neuron states, these weight selection methods are considered non-local and biologically improbable. (See [42, pp. 49-52] for details.)

Figure 7.26 shows a nine neuron Hopfield network composed of SNM's. Each neuron has three synapses. The first synapse receives an inhibitory reset signal, which stops all network oscillations. The second synapse receives an excitatory input signal. The third synapse is a cluster which receives inputs from all of the neurons in the network including itself. The input connections may be either inhibitory or excitatory depending upon the stored memories. The neurons' parameters are chosen so that the resting potentials are just slightly below the threshold levels. Thus, if the net input into a neuron is positive, it tends to fire. Conversely, if the net input is negative it does not. When the network is in a stable state, the contributions from the firing neurons cause their synaptic clusters to saturate at a positive value, and the output spikes become self-sustaining. Those neurons not firing have no effect on the other neurons in the network, and the inhibitory inputs from the firing neurons prevent them from reaching threshold.

The nine neuron Hopfield network was tested by storing the three patterns shown

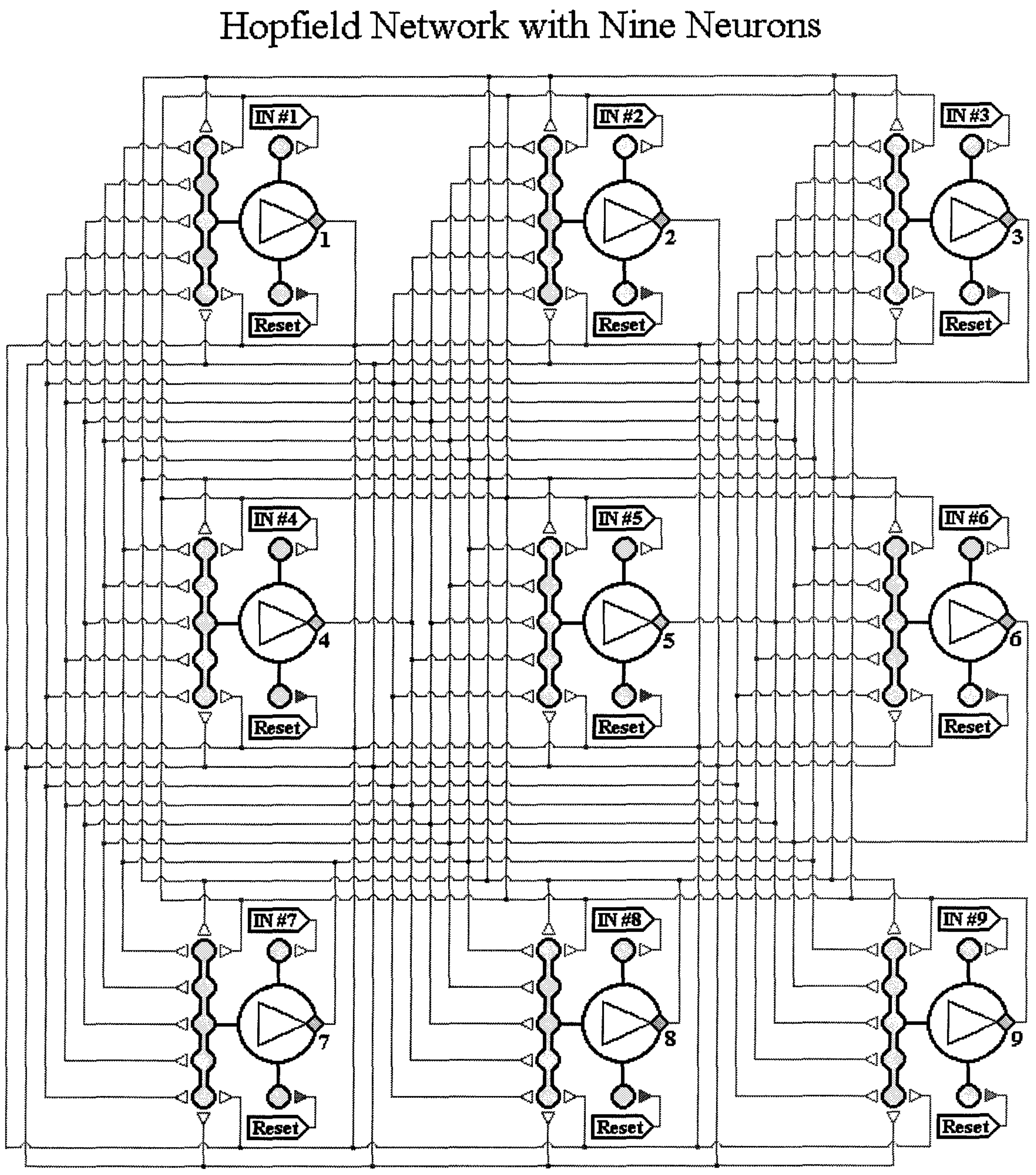


Figure 7.26: **Hopfield Network.** The plot shows a fully connected Hopfield network with nine spiking neurons. Each neuron has three synapses. The first synapse receives the inhibitory reset signal, which stops all network oscillations. (Although the connecting wires are not shown, all of the neurons are assumed to receive the *same* reset signal.) The second synapse receives an excitatory input signal. (All of the neurons have separate input signals.) The third synapse is a cluster which receives inputs from all of the neurons in the network including itself. The input connections may be either inhibitory or excitatory depending upon the stored memories.

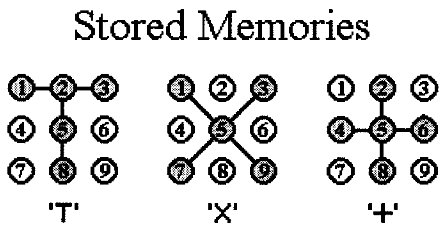


Figure 7.27: **Memories Stored in Hopfield Network.** The plot shows the three different patterns stored in the nine neuron Hopfield network of Figure 7.26. The colored neurons indicate those that oscillate when one of the memories is correctly retrieved. The shapes formed by the patterns can be described by: 'T', 'X', and '+'.

in Figure 7.27. To store all three patterns in the network, the weights were chosen according to Equations (7.23) and (7.24), with the self-connecting weights, $w_{ii}$, included. To eliminate all spurious states, a constant value was added to the weights between neurons and another constant was added to the self-connecting weights;[9] i.e.:

$$W_{ij} = K_{ij} + \sum_{p=1}^{3} w_{ij}^{p} \tag{7.25}$$

$$\text{where:} \quad K_{ij} = \begin{cases} 1 & \text{if } i \neq j \\ -3 & \text{if } i = j \end{cases} \tag{7.26}$$

Notice that each neuron in the Hopfield network has a similar behavior to the Dual Control Memory Neuron of Section 6.3.2. The reset input acts as the "OFF" input, while the feedback and the inputs from the other neurons act as the "ON" input. (The input signal into each neuron acts as another "ON" input.) Since the neurons should oscillate when their "ON" inputs are excitatory, the parameters were chosen to closely match those used in Figure 6.4. For all tests, the neurons' parameters were: $\Theta_i = 0.5$, $R_{i0} = 0.48$, and $\tau_i = 0.2$. The reset input parameters were: $R_{i1} = 1.5$, $T_{i1} = -3.0$, and $\tau_{i1} = 2.0$. (The parameters were chosen so that a reset spike would inhibit the neurons, and the neurotransmitter at the synaptic clusters would decay away before any of the neurons could start firing again. Thus, with no other input

[9]Adding constants to the weights can often help reduce the number of spurious states, particularly when the weights are chosen using Equation (7.23). With the patterns used to test the network, all spurious states can be eliminated, and the memories recalled exactly. When different patterns are used, this is not always possible.

spikes, a reset spike forces the network into the zero state, where it remains.) The input signal's synaptic parameters were: $R_{i2} = 0.6$, $T_{i2} = 0.6$, and $\tau_{i2} = 0.2$. And for the synaptic clusters receiving inputs from the other neurons, the parameters were: $R_{i3} = 0.53$ and $\tau_{i3j} = 0.375$. The $T_{i3j}$ values were set equal to the weights found with Equation (7.25), but normalized so that the magnitude of the maximum weight value was 2.75, which was the same neurotransmitter weight value used for the "ON" input of the dual control memory neuron in Figure 6.4. Thus, the actual weight matrix was:

$$T = \begin{bmatrix} -0.92 & 0.00 & 2.75 & -1.83 & 1.83 & -1.83 & 0.92 & 0.00 & 0.92 \\ 0.00 & -0.92 & 0.00 & 0.92 & 1.83 & 0.92 & -1.83 & 2.75 & -1.83 \\ 2.75 & 0.00 & -0.92 & -1.83 & 1.83 & -1.83 & 0.92 & 0.00 & 0.92 \\ -1.83 & 0.92 & -1.83 & -1.83 & 0.00 & 1.83 & -0.92 & 0.92 & -0.92 \\ 1.83 & 1.83 & 1.83 & 0.00 & 0.00 & 0.00 & 0.00 & 1.83 & 0.00 \\ -1.83 & 0.92 & -1.83 & 1.83 & 0.00 & -1.83 & -0.92 & 0.92 & -0.92 \\ 0.92 & -1.83 & 0.92 & -0.92 & 0.00 & -0.92 & -1.83 & -1.83 & 1.83 \\ 0.00 & 2.75 & 0.00 & 0.92 & 1.83 & 0.92 & -1.83 & -0.92 & -1.83 \\ 0.92 & -1.83 & 0.92 & -0.92 & 0.00 & -0.92 & 1.83 & -1.83 & -1.83 \end{bmatrix} \tag{7.27}$$

Figure 7.28 shows the results for the Hopfield network with all three patterns stored. The reset spikes cause the network to go to the off state, and the random input spikes cause the network to eventually converge to one of the three stored memories and remain there until another reset spike occurs. Since the weights were chosen to eliminate all spurious states, no other oscillatory patterns are possible. Notice that because the synaptic clusters saturate when the network converges to a stable state, all of the oscillating neurons have nearly the same frequency. However, they are not necessarily firing in phase. If the network was designed so that each input had a separate synapse, then the output spikes between the neurons would tend to be phase locked; however, the firing frequencies could vary between recalled memories.

Since the SNM can also incorporate delays within its synaptic connections, a

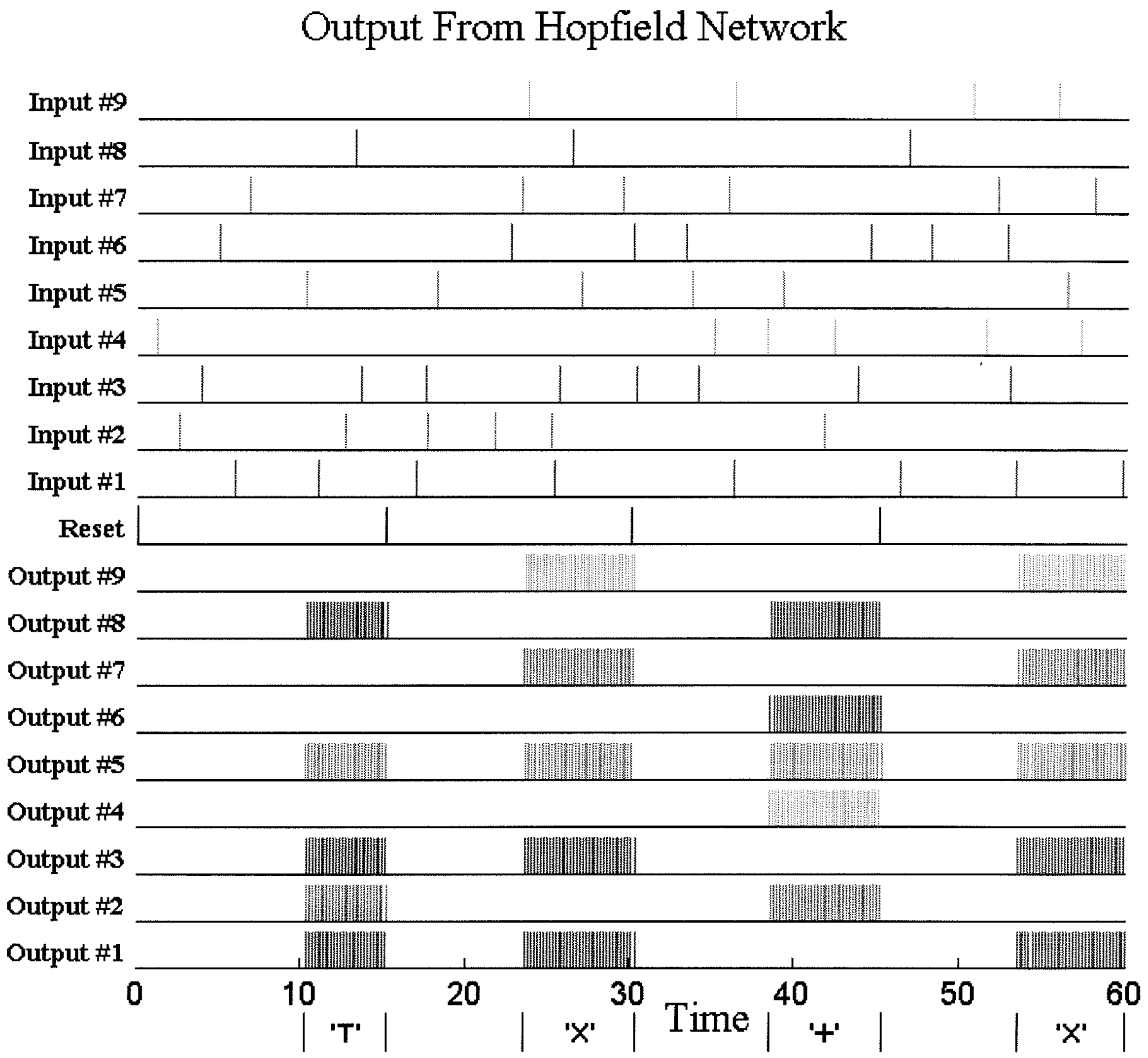


Figure 7.28: **Sample Output for Hopfield Network.** This plot shows a set of random input spikes, and the resulting output spikes from the Hopfield network shown in Figure 7.26. The network has only four stable states: 'T', 'X', '+', and all off. (The letters refer to the shapes formed when the appropriate neurons in Figure 7.26 are oscillating.) The reset spikes, which place the network in the off state, arrive at 0, 15, 30, and 45. After each reset, there is a "dead time" of approximately 8, during which the neurons are insensitive to input spikes. As the neurons again become responsive to the input signals, the network converges to a stable attractor pattern, in which the "ON" neurons oscillate. (With no input spikes, the network remains in the off state.) Once the network is in one of the stable patterns, it remains there until reset. The parameters are such that the oscillating neurons do not necessarily fire in phase, but all have nearly identical frequencies.

Hopfield network constructed of these neurons can be used to store oscillating patterns or limit cycles. To test this claim, the nine neuron Hopfield network was used to store the same memories shown in Figure 7.27, but now the network was designed to shift between the memories. After spending approximately 3 time units in each of the stored memories, the network was to shift to the next memory; i.e., 'T' $\Rightarrow$ 'X' $\Rightarrow$ '+' $\Rightarrow$ 'T' $\Rightarrow$ .... To accomplish this task, the network was constructed with two complete sets of neuron interconnections. The first set incorporated no delays and used the weight values calculated with Equation (7.25). The second set of connections had a delay of 3 and connected to the same synaptic cluster.

For the network to work as desired, the weights for the second set must be chosen so that they produce no stable states, but the neurons are forced to shift between the stable states created with the first set. The weights for the second set were chosen according to:

$$s_{ij}^p = 3\xi_i^n \cdot \xi_j^p - \xi_i^n - \xi_j^p = \begin{cases} +1 & \text{if } \xi_i^n = \xi_j^p = 1 \\ 0 & \text{if } \xi_i^n = \xi_j^p = 0 \\ -1 & \text{if } \xi_i^n \neq \xi_j^p \end{cases} \tag{7.28}$$

$$\text{where:} \quad n = \begin{cases} p+1 & \text{if } p < P \\ 1 & \text{if } p = P \end{cases} \tag{7.29}$$

As before, constants can be added to weight matrix to eliminate spurious states and make the shift between states complete.[10] The total weight matrix used for shifting is given by:

$$S_{ij} = \hat{K} + \sum_{p=1}^{3} s_{ij}^p \tag{7.30}$$

$$\text{where:} \quad \hat{K} = 0.4 \tag{7.31}$$

[10]Without the added constant, the shift is incomplete in the sense that not all of the appropriate neurons in the next pattern are stimulated to fire; however, even without the added constant, the majority of the neurons match the stored memory, and the remaining neurons eventually converge to the correct values due to the $W_{ij}$ weights. By using an added constant with the patterns used to test the network, all spurious states can be eliminated, and the shifts be made complete. When different patterns are used, this is not always possible.

Now, the $W_{ij}$ weights create the memory patterns, and the $S_{ij}$ weights causes the state of the network to shift between the memories. However, if the effects from the $W_{ij}$ weights are stronger than the effects from the $S_{ij}$ weights, it is possible for the network to stay in one or more of the memories without shifting to the next. Thus, it is necessary that the superposition of the $W$ and $S$ matrices have no stable states. Consequently, the $W$ and $S$ matrices must be weighted so that the total effective matrix, $X$, produces *no* stable patterns. Since the $S$ matrix should not have any stable patterns anyway, it usually must contribute more than the $W$ matrix. For the patterns tested, the $X$ matrix was given by:

$$X_{ij} = W_{ij} + R \cdot S_{ij} \tag{7.32}$$

$$\text{where:} \quad R = 5 \tag{7.33}$$

Notice that even though the total effective matrix has no stable states, each of the stored memories is stable for the length of the delay associated with the $S_{ij}$ connections. This is because the $W_{ij}$ connections with no delays are contributing to the current patterns, as well as the $S_{ij}$ connections from the *previous* memory pattern.

All of the neuron parameters used for pattern shifting were the same as those used to store the stable patterns; however, the $T_{i3j}$ values were normalized so that the magnitude of the maximum weight value of the $X$ matrix was 2.75. Thus, for the connections without delay, the $T$ matrix was:

$$T = \begin{bmatrix} -0.16 & 0.00 & 0.49 & -0.32 & 0.32 & -0.32 & 0.16 & 0.00 & 0.16 \\ 0.00 & -0.16 & 0.00 & 0.16 & 0.32 & 0.16 & -0.32 & 0.49 & -0.32 \\ 0.49 & 0.00 & -0.16 & -0.32 & 0.32 & -0.32 & 0.16 & 0.00 & 0.16 \\ -0.32 & 0.16 & -0.32 & -0.32 & 0.00 & 0.32 & -0.16 & 0.16 & -0.16 \\ 0.32 & 0.32 & 0.32 & 0.00 & 0.00 & 0.00 & 0.00 & 0.32 & 0.00 \\ -0.32 & 0.16 & -0.32 & 0.32 & 0.00 & -0.32 & -0.16 & 0.16 & -0.16 \\ 0.16 & -0.32 & 0.16 & -0.16 & 0.00 & -0.16 & -0.32 & -0.32 & 0.32 \\ 0.00 & 0.49 & 0.00 & 0.16 & 0.32 & 0.16 & -0.32 & -0.16 & -0.32 \\ 0.16 & -0.32 & 0.16 & -0.16 & 0.00 & -0.16 & 0.32 & -0.32 & -0.32 \end{bmatrix} \tag{7.34}$$

And for the connections with the delay of 3, the $T$ matrix was:

$$T = \begin{bmatrix} -0.49 & 1.94 & -0.49 & 0.32 & 1.13 & 0.32 & -2.10 & 1.94 & -2.10 \\ -0.49 & -0.49 & -0.49 & 0.32 & 1.13 & 0.32 & 0.32 & -0.49 & 0.32 \\ -0.49 & 1.94 & -0.49 & 0.32 & 1.13 & 0.32 & -2.10 & 1.94 & -2.10 \\ 0.32 & -2.10 & 0.32 & -1.29 & -0.49 & -1.29 & 1.13 & -2.10 & 1.13 \\ 1.13 & 1.13 & 1.13 & -0.49 & 2.75 & -0.49 & -0.49 & 1.13 & -0.49 \\ 0.32 & -2.10 & 0.32 & -1.29 & -0.49 & -1.29 & 1.13 & -2.10 & 1.13 \\ 0.32 & 0.32 & 0.32 & -1.29 & -0.49 & -1.29 & -1.29 & 0.32 & -1.29 \\ -0.49 & -0.49 & -0.49 & 0.32 & 1.13 & 0.32 & 0.32 & -0.49 & 0.32 \\ 0.32 & 0.32 & 0.32 & -1.29 & -0.49 & -1.29 & -1.29 & 0.32 & -1.29 \end{bmatrix} \tag{7.35}$$

Figure 7.29 shows the results from the Hopfield network with pattern shifting between the three stored memories. The reset spikes cause the network to go to the off state, and the random input spikes cause the network to eventually converge to one of the three stored memories. The network only remains in the stored memory for about 3 time units when the delayed connections push the network into the next pattern. The cycle continues until another reset spike is received. Since the weights were chosen to eliminate all spurious states, no other patterns are possible. As before, the synaptic clusters saturate when the network converges to a stable state, so all of the oscillating neurons have nearly the same frequency.

Finally, the previously designed Hopfield networks were constructed so that each neuron encoded a binary number, which depended upon whether or not it was oscillating. But it is also possible to design the network so that the output frequency of each neuron encodes an *analog* number, with higher output frequencies representing larger numbers. The main drawback of such an analog encoding scheme is that it complicates the design procedure, and makes the choice of the network parameters difficult, particularly when more than one pattern is to be stored. Also, the precision of the analog numbers is limited by the total number of neurons and patterns.

## 7.10 Summary of Network Computations

This chapter presented several simple networks that perform relatively complex computational tasks. The chapter started by discussing the stimulus detector network (Section 7.2), which could be used to phase lock (synchronize) different input spike trains. Since some of the other networks presented required synchronized input signals, this network acted as a preprocessor for them. The network could be configured so that the phase locking only occurs if the total activity on all of the input lines is sufficiently strong, and when the total activity is weak, no signals pass through. The number of neurons required for a stimulus detector is simply equal to the total number of input lines plus one, for the high gain pacing neuron.

Section 7.3 presented several different type of networks for pattern recognition. A distinction was made between temporal, spatial, and spatial-temporal patterns. A temporal pattern consisted of one input signal with a sequence of spikes. A spatial pattern used several inputs, but with only one target event on each input line. (Events could be either spikes or non-spikes.) A spatial-temporal pattern used several input signals with at least two target events on one or more of the input lines. The network configuration depended upon the type of pattern being recognized.

For the temporal pattern recognition task described in Section 7.3.2, each neuron

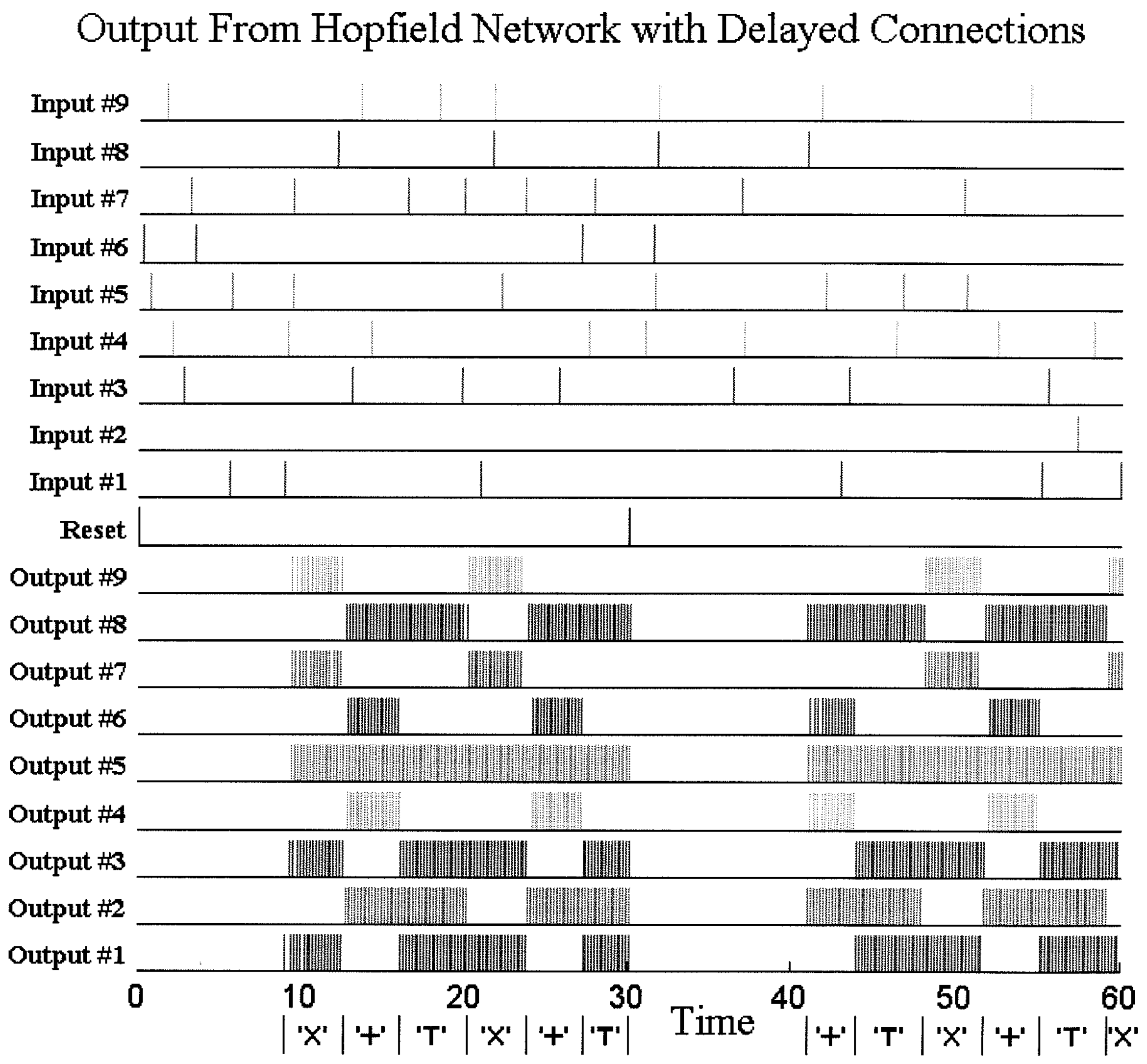


Figure 7.29: **Sample Output for Hopfield Network with Pattern Shifting.** This plot shows a set of random input spikes, and the resulting output spikes from a nine neuron Hopfield network. The network has two complete sets of interconnections between the neurons. The first set creates the four stable states: 'T', 'X', '+', and all off. The second set attaches to the same synapses, but all of the connections contain a delay of 3. The weights for the second set are chosen so that they produce no stable states, but instead force the neurons to shift between the stable states created with the first set. The reset spikes, which cause the network to go to the off state, arrive at 0 and 30. After each reset, there is a "dead time" of approximately 8, during which the neurons are insensitive to the input spikes. As the neurons again become responsive to the input signals, the network converges to a stable attractor pattern, in which the "ON" neurons oscillate. Once the network is in one of the stable patterns, it shifts to the next pattern after approximately 3 time units. The network continues shifting between patterns in the specified order until another reset spike arrives.

was designed to create a dividing hyperplane in the input space, where the dimension of the input space was defined by the number of events within the target pattern. While the neuron was designed for input events occurring at a specific frequency, the location of the dividing hyperplane determined how much the frequency could differ from this frequency and still have the neuron recognize the pattern. Figure 7.7 showed how these neurons could be used in the first layer of a three layered network for more complex pattern recognition tasks. A three layer network is capable of performing *any* classification problem. The number of neurons in the first layer equals the number of required hyperplanes; the number of neurons in the second layer equals the number of convex target regions in the input space (non-convex regions can be considered to be the union of convex regions); only one neuron is every required for the third layer, and it is only necessary if there is more than one convex region.

For the spatial recognition task, described in Section 7.3.3, a single neuron was designed to recognize a particular spiking pattern occurring on a set of input lines with one target event per line. If there was more than one target event on one or more of the input lines, the problem was a spatial-temporal recognition task, described in Section 7.3.4. Figure 7.9 showed that when a layer of temporal recognizing neurons was followed by a layer of spatial recognizing neurons, the network could recognize a particular spatial-temporal pattern. While the effect of this network was only to create a single dividing line in the spatial-temporal input space, several of these networks could be used with their outputs feeding into "AND" neurons whose outputs are then fed into an "OR" neuron, allowing such a network to perform any spatial temporal recognition task in much the same way as was described for the temporal recognition task.

The number of neurons within the second and third layers of a multilayered pattern recognizing network could be reduced to a single logic neuron, which was described in Section 7.4. A logic neuron can be used to evaluate any Boolean logic expression. The expression must first be reduced to conjunctive normal form (CNF), which is a product (AND) of sums (OR). The number of synaptic clusters required equals the number of clauses (products) in the CNF of the Boolean expression, and the number of inputs into each cluster equals the number of literals (variables) summed within its corresponding clause.

Section 7.5 showed how a network with feedback connections could be used to store entire segments of an input spike train. The spike pattern was preserved indefinitely, until it was replaced with a new memory. The total length of the memory was determined by the amount of delay on the feedback connections.

A counting network can be used to determine the number of spikes within a spike train. Section 7.6 described two different types of counters. The first was a linear counter, in which the number of neurons oscillating in the network represented the number of input spikes. The number of neurons in the network sets the limit as to how high it can count. The second was a sequential counter, in which only one neuron at a time oscillated, but each neuron in the network represented a specific number. The sequential counters could be cascaded together to create networks which encode different digits in a number. The number of cascaded networks determines the number of "digits" encoded in the network, and the number of neurons in each counting network sets the base for the encoding scheme.

Section 7.7 presented a multiplexer network, which selects one of many input signals to be transferred to a single output. This network can be particularly useful for routing data within a complex nervous system. The network required two neurons for each input signal, plus one for the summing neuron. The section also discussed how a demultiplexer could be constructed from a multiplexer.

While the pattern recognizing networks were useful for determining if an input spike train contained a specific spike sequence, it is sometimes necessary to determine if two or more spike trains are equal to each other. Section 7.8 described a comparator network, which would output spikes if the input spike trains were equivalent, regardless of the actual input patterns. While the network shown in Figure 7.22 only compared two input signals, more signals can be compared by including one synchronizing identity neuron and two difference detecting neurons for each additional input. The number of synapses on the pattern comparator neuron sets the length of the patterns being compared.

And finally, Section 7.9 described how a Hopfield-like associative memory network could be constructed from spiking neurons. The basic problem was to store a set of patterns in such a way that when the network was presented with a new pattern, it responded by producing whichever one of the stored patterns most closely resembled the input pattern. The stored patterns could be taken to be either a 0 or 1 value at each neuron in the network, where 1 indicated that the neuron was oscillating, while 0 indicated that it was not.

Each of the networks in this chapter took one or more input spike trains and produced one or more output spike trains. Chapter 8 discuss how information may actually be encoded in such spike trains. It is important to remember that these networks were not meant to be precise computational devices. They usually perform their intended tasks well, but under some operating conditions their outputs might not be exactly as desired; e.g., the pattern recognizing networks of Section 7.3 were designed for a specific input frequency, and faster or slower input signals could cause errors (Section 7.3.5).

# Chapter 8 How is Information Encoded?

## 8.1 Introduction

The networks presented in the previous chapter demonstrated some of computational abilities for networks composed of the SNM, but one fundamental question remains: how are spikes used to encode information? Neurobiologists have spent a great deal of time and effort trying to decipher the coding scheme used by real neurons. In general, most sensory neurons simply fire when stimulated, with the output frequency depending primarily upon the strength of the input stimulus. (Frequency also appears to vary with time, with neurons adapting to large input stimuli and reducing their firing rate. This could be due to either fatigue, or a temporary modification of the neuron parameters, or a feedback signal from other neurons, or some combination of these. Section 4.3.5 discusses how adaptation can be modeled with the SNM.)

Of course, due the similarity between spikes within a nervous system, it is evident that their meaning can only be inferred from the network locations in which they occur. In Kuffler's book on neuroscience [61, p. 6], he writes:

> In the nervous system, however, the frequency or pattern of discharges will not do on its own as a code, for the following reason: Even though impulses and frequencies are the same in different cells responding to light, touch, or sound, the content of information is quite different. The quality or meaning of a signal depends upon the origins and destinations of the nerve fibers, that is, on their connections. Various types of sensory modalities (light, sound, touch) are linked to different parts of the brain; even within each modality and in each area of the cortex, specific stimuli (such as lines or rectangles in the visual system) act selectively on specific

> populations of neurons. This organization is brought about by strictly determined connections. Frequency coding is used by the nervous system to convey information about the intensity of a stimulus rather than about its quality.

The networks in the previous chapter were developed with the tacit assumption that the nervous system probably uses some common network configurations to perform general computations, independent of the type of signal being received [48]. Of course, how an organism responds to a particular sensory input is very dependent upon the type of stimulus; e.g., the spikes signaling pain from a pin prick will probably evoke some type of response, while the auditory spikes stimulated by a gentle breeze blowing through trees probably will not.

Traditionally, the mean firing rate of a neuron is assumed to represent an analog variable. But if information is only encoded within the average spike frequency, then individual spikes do not carry significant information. Consequently, such an encoding scheme is very inefficient. However, there is a growing body of experimental evidence that suggests many biological neural systems use the timing of action potentials to encode information [22, 90]. Some experiments have even demonstrated that biological neurons are able to fire in vitro with high timing precision [67, see references listed therein for experimental results].

While there are numerous ways to encode information, this chapter focuses on methods which are compatible with the networks already presented in Chapter 7. Section 8.2 summarizes the encoding formats for the previous networks' input/output spikes. Section 8.3 discusses how the strength of an input stimulus can be retained within the phase differences between near-synchronous spikes, while Section 8.4 describes how the strength of an input stimulus can be logarithmically encoded in the amount of neurotransmitter released at a synapse. Finally, Section 8.5 summarizes the information encoding schemes presented in this chapter.

## 8.2 Format for Input/Output Spikes

### 8.2.1 Introduction

This section examines the encoding format used by the input/output spikes of the networks described in Chapter 7. These networks use an opportunistic approach to information encoding, where the meaning of each spike depends upon the task being performed. Thus, for some networks, the average spike rate *does* encode the information, but for others, each spike has significance.

The stimulus detector presented in Section 7.2 is perhaps the most important network for encoding information, since it is used as a preprocessor for those networks which require their input spikes to have a fixed underlying frequency. Consequently, its encoding scheme is discussed in greater detail than the other networks. Section 8.2.2 is devoted entirely to the stimulus detector network, while all of the other networks are briefly summarized in Section 8.2.3.

### 8.2.2 Stimulus Detector's Spikes

By using the stimulus detector of Section 7.2 to preprocess inputs from sensory neurons, information can be encoded within individual spikes. Sensory neurons can still encoded information within their average output spike frequency, but the stimulus detector converts this into a pattern of spikes and non-spikes which occur at a fixed frequency. These individual bits can then be used in one of the networks which require synchronized inputs (pattern recognizers, logic neurons, memory networks, and comparators).

While several variations were discussed, the usual stimulus detector requires a burst of input spikes to stimulate a high gain neuron, which then produces spikes at a nearly constant frequency. The output neurons which correspond to the spiking inputs, produce spikes at all of the events occurring within the duration of the stimulus. The frequency of output events is often lower than that of the input spike burst. In general, the frequency of events can be considered the sampling frequency, with each output spike indicating the presence of an input stimulus. Notice that the synchronized outputs from a stimulus detector may pass through another stimulus detector without any loss of spikes, provided that the pace neuron of the second detector is producing spikes at a frequency greater than or equal to the first.

A large network may contain different stimulus detectors, with each having its own high gain pace neuron. Furthermore, the output spikes from the pace neurons may be connected to each other at inhibitory synapses, whose parameters are chosen so that the pace neurons implement a winner-take-all behavior; i.e., only one stimulus detector produces output spikes, while the others are inhibited. A large input stimulus is required to overcome the inhibitory input and cause one of the "off" stimulus detectors to switch "on" and become the "winner." In this sense, the stimulus detector network can be considered a primitive model for consciousness.[1] Strong input stimuli draw the system's *attention*, resulting in active information processing. Weak stimuli still enter the system, but do not lead to active processing. The system's "awareness" can switch between different types of stimuli, depending upon their input strengths.

### 8.2.3 Other Networks' Spikes

There were three different types of pattern recognizing networks (temporal, spatial, and spatial-temporal) presented in Section 7.3. All of these networks require the input signals to have a fixed frequency of events. However, this constraint can be relaxed for the spatial recognizer when it is only necessary to determine which inputs contain spikes and which do not. Nevertheless, each input spike is significant and helps determine when the input matches the target pattern. Only when a match occurs are any output spikes produced. The temporal recognizer produces output spikes with the same frequency of events as its input line, but the spatial and spatial-temporal recognizers only produce spikes with a fixed underlying event frequency if all of the input lines have the same frequency. Thus, while the events on each input

[1]Some neurobiologists have suggested that neural oscillations may form the basis of consciousness, and as a person becomes cognizant of different stimuli, the neurons involved in the oscillations change [57].

line must have an underlying frequency, there does not need to be synchronization between the different input lines.

The inputs into the logic neuron (Section 7.4) were assumed to have been first synchronized. Furthermore, at every event the synchronization procedure should produce either an output spike for each variable or its inverse (see Figure 7.12). Only when the neuron's Boolean expression is satisfied, are any output spikes produced. Of course, the output events have the same underlying frequency as the input events, but the input frequency can vary below a maximum input rate for which the neuron was designed; i.e., there is no constraint on the minimum input frequency.

Both the memory and comparator networks shown in Figures 7.14 and 7.24 take unsynchronized inputs and synchronize them using their first layer of neurons. This layer acts as a stimulus detector, but since the signal from the pace neuron is also used in subsequent layers, the synchronizing function can not be performed by an independent network; i.e., the network and the stimulus detector require the same pace signal. Each input spike carries information, and ideally there is a one-to-one correlation between the unsynchronized input spikes and the output spikes from the first layer. Such a correlation is possible if the pace neuron is spiking at a higher frequency than the maximum allowable frequency for the input spikes. This is in contradistinction to the usual stimulus detector, which uses a lower sampling frequency. Finally, notice that the memory network also uses a control signal. A single control spike indicates the start of a new memory formation, and when there has not been a recent control spike, any input spikes have no effect on the network's output.

The counter networks of Section 7.6 do not require the input spikes to be synchronized, but since each spike alters the state of the network, all spikes carry significant information. Both of the counters use a reset input signal to put the network back into the "zero" state before counting again. Each input spike is counted provide that it does not follow the reset spike or previous input spike too closely. The output from the counters is encoded via the states of the output neurons, which are either oscillating or silent. Thus, while each input spike can be thought of as representing one bit of information, the output spike *rate* of each neuron represents one bit of information.

In this sense, counters transform information from one format into another.

For the multiplexer network of Section 7.7, the inputs can have a fixed underlying frequency, but it is not required. Similar to the memory network, there are control spikes to determine which of the input signals are actually passed through to the output. Except when the spikes on an input line are very closely spaced or when there are several "ON" inputs with near coincident spikes, there is a one-to-one correlation between the input and output spikes. Therefore, each individual spike is significant. Although the passing neurons and the summing neuron introduce delays into the system, the relative timing between spikes is fairly well preserved. Notice that the final summing neuron can be replaced by a synchronized identity neuron receiving spikes from a pace neuron. In this case, the output spikes have a fixed underlying frequency, thus allowing a multiplexer and stimulus detector to be combined into a single network.

The Hopfield network of Section 7.9 is similar to the counter networks, in that the output signal is encoded in the states of the neurons – not the individual spikes. Each neuron has an input and reset signal, and can be assigned a binary value, depending upon whether or not it is oscillating. (It is also possible to use an analog encoding scheme – see last paragraph of Section 7.9.) A single reset spike stops all network oscillations. But individual input spikes are not considered significant since it is the input frequency that determines the strength of the stimulus. And it is the net effect from all of the input signals that moves the network into the basin of attraction of a stable state.

## 8.3 Phase Encoding

### 8.3.1 Introduction

Previously, the neurotransmitter contributions from the pace neuron into the synchronized identity neurons of a stimulus detector were assumed to have very short time constants. Thus, when a stimulus was present, the corresponding identity neuron produced spikes very close to the times of the pseudospikes (times of maximum neurotransmitter contribution from the pace neuron). While the input spikes can be from a sensory neuron, which encodes the strength of a stimulus in its spike rate, the output spikes from the identity neuron merely indicate the presence of an input signal, and do not encode intensity (assuming that the frequency of events – sampling frequency – is less than the input spike frequency). While for some applications it may only be necessary to know when and if a specific input stimulus is present, for others the strength of the stimulus is also required.

This section describes how the intensity of a stimulus may be encoded within the *phase* of the output spikes. The basic idea is to still use a stimulus detector network to sample the inputs and produce near-synchronous output spikes, but the neurotransmitter contribution from the pace neuron is broadened.[2] Thus, all of the output events from the stimulus detector still have a fundamental frequency; however, the individual spikes may slightly lead the pseudospike events, with the amount of lead time dependent upon the intensity. In this way, the network is able to encode information concerning the presence and strength of an input signal in a manner that allows it to be processed by other neurons, whose synaptic connections have fixed delays associated with them.

As a simple example of an application where stimulus strength is necessary, consider the problem of recognizing a particular odor. The olfactory system consists of numerous chemical receptor neurons, which produce output spikes at a frequency that depends upon the strength of the input stimulus. Different combinations of stimuli represent different scents. If the sensory neurons' output spikes first go through a stimulus detector network, then the output spikes from the stimulus detector can be used as input spikes into pattern detector networks, which recognize specific odors. The information concerning the strength of the stimuli is not lost, but encoded in the phase differences between the actual output spikes from the synchronized identity

[2]To simplify the analysis, the pace signal is assumed to come from a single neuron; however, for robustness, real biological systems would be more likely to use the net input signal from a group of oscillating neurons. The encoding schemes presented in this section do not depend upon how the pace signal is actually generated.

neurons and the pseudospikes generated from the pace neuron. If one stimulus is disproportionally stronger than the others, it leads the pseudospikes by significantly more than the others; consequently, the times of the input spikes into the spatial pattern recognizer do not correspond, and if the difference is significant enough, the pattern goes unrecognized. This feature allow the system to correctly distinguish between similar but distinct odors. (As a hypothetical example, assume that both fresh milk and sour milk stimulate the same sensor neurons, but in different proportions. A system which retains the intensity information is able to distinguish between them.)

Section 8.3.2 discusses a neuron which always outputs a spike when it receives one from the pace neuron. The input signal causes the output spikes to either lead (excitatory input) or lag (inhibitory input) the times of the spikes produced with no input signal. Section 8.3.3 generalizes this phase encoding to those neurons which receive pace spikes but do not always output spikes, e.g., the synchronized identity and inverse neurons. Section 8.3.4 shows how phase encoding neurons can be arranged in layered networks.

## 8.3.2 Phase Encoding within a Spiking Neuron

To understand how the intensity of an input signal can be encoded in the phase of an output spike, first consider a neuron with only two input synapses: one receives the input signal, and one receives an excitatory pace signal from a beating neuron. The actual input signal may be from multiple neurons (connecting at a synaptic cluster) or an external input, but the total effective neurotransmitter can be grouped together and called $U^{\text{in}}$. Therefore, the neuron's potential is:

$$\begin{aligned} V_i(t) \;=\; & \tanh^2\left(\frac{t-t_i^K}{\tau_i}\right)\cdot\left\{R_{i0} + R_{ij}\tanh\left(\frac{T_{ij}}{R_{ij}}\cdot U^{\text{in}}(t)\right)\right. \\ & \left. + R_{ip}\tanh\left[\frac{T_{ip}}{R_{ip}}\cdot\left(\frac{1}{\kappa\left(\frac{\Delta_p}{\tau_{ip}}\right)}\right)\cdot f\left(\frac{t-t_p^K-d_{ip}}{\tau_{ip}}+\psi\left(\frac{\Delta_p}{\tau_{ip}}\right)\right)\right]\right\} \end{aligned} \tag{8.1}$$

Here, the "$ij$" parameters correspond to the input signal and the "$ip$" parameters correspond to the pace signal, whose neurotransmitter is given by Equation (4.23).

If the neuron's time constant, $\tau_i$, is sufficiently small, then the refractory coefficient is approximately 1, provided that the neuron only fires once per input spike from the pace neuron; i.e., if $\Delta_p$ represents the time between pace spikes, then:

$$\tanh^2\left(\frac{t_i^{K+1} - t_i^K}{\tau_i}\right) \approx \tanh^2\left(\frac{\Delta_p}{\tau_i}\right) \approx 1 \tag{8.2}$$

Notice that if $t_i^{K+1} - t_i^K \approx \Delta_p$, then the pace neuron is actually determining the fundamental output spike frequency, and the input signal, $U^{\text{in}}(t)$, only creates a small perturbation in the phase.

To facilitate the analysis, the number of parameters can be reduced (as in Section 6.4), with the firing condition being:

$$\bar{\Theta} = \tanh\left[h^{\text{in}}(t)\right] + \left(\frac{1}{r}\right)\cdot\tanh\left[\left(\frac{1}{s}\right)\cdot f\left(\frac{\bar{t}}{\tau_{ip}}\right)\right] \tag{8.3}$$

where:

$$\bar{\Theta} \equiv \frac{\Theta_i}{R_{ij}} - \frac{R_{i0}}{R_{ij}} \tag{8.4}$$

$$s \equiv \left(\frac{R_{ip}}{T_{ip}}\right)\cdot\kappa\left(\frac{\Delta_p}{\tau_{ip}}\right) > 0 \tag{8.5}$$

$$r \equiv \frac{R_{ij}}{R_{ip}} > 0 \tag{8.6}$$

$$h(t) \equiv \frac{T_{ij}}{R_{ij}}\cdot U(t) \tag{8.7}$$

$$\frac{\bar{t}}{\tau_{ip}} \equiv \frac{t - t_p^K - d_{ip}}{\tau_{ip}} + \psi\left(\frac{\Delta_p}{\tau_{ip}}\right) \tag{8.8}$$

Notice that $\bar{t}$ represents the time measured relative to the arrival time of the pace spike, plus the contribution from the beating behavior, $\tau_{ip}\cdot\psi\left(\frac{\Delta_p}{\tau_{ip}}\right)$.

Now, this section only considers neurons which satisfy the following two constraints:

1) Regardless of the input signal, $h^{\text{in}}$, an input spike from the pace neuron is required before any output spikes can be produced, and each input spike from the pace neuron must produce an output spike. Thus, the neuron fires with a fixed

underlying frequency, and the input signal only perturbs the phase of the output spikes. Notice that this neuron is different from the identity and inverse neurons discussed in Section 6.5, which do not always produce an output spike with each pace spike. Due to the tanh (−) function, the effect on the neuron's potential from $h^{\text{in}}$ is bounded between -1 and +1, so these constraints become:

$$1 \le \bar{\Theta} \le \left(\frac{1}{r}\right) \cdot \tanh\left(\frac{\mathrm{e}^{-1}}{s}\right) - 1 \tag{8.9}$$

For there to exist a value of $\bar{\Theta}$ which solves this equation, the relationship between $s$ and $r$ is constrained by:

$$0 < r \le 0.5 \cdot \tanh\left(\frac{\mathrm{e}^{-1}}{s}\right) \tag{8.10}$$

This equation is plotted in Figure 8.1.

2) The input signal changes on a slower time scale than the neurotransmitter contribution from the pace spikes, so that $h^{\text{in}}$ can be considered a constant through all possible output spike times. Thus, it is possible to solve for the time of an output spike as a function of the input signal; i.e.:

$$\frac{\bar{t}\left(h^{\text{in}}\right)}{\tau_{ip}} = F^{-1}\left(s \cdot \text{Atanh}\left[r \cdot \left(\bar{\Theta} - \tanh\left(h^{\text{in}}\right)\right)\right]\right) \tag{8.11}$$

Figure 8.2 shows how $h^{\text{in}}$ can affect the neuron's potential and modify the time at which it reaches threshold, $\bar{\Theta}$. Notice that when there is no input signal, the time of the output spike is given by:

$$\frac{\bar{t}\left(h^{\text{in}} = 0\right)}{\tau_{ip}} = F^{-1}\left(s \cdot \text{Atanh}\left(r \cdot \bar{\Theta}\right)\right) \tag{8.12}$$

And when $h^{\text{in}}$ is non-zero, the times of the output spikes are perturbed from this value, with excitatory inputs causing the spikes to lead and inhibitory inputs causing the spikes to lag. Of course, the total phase difference is bounded by the earliest and

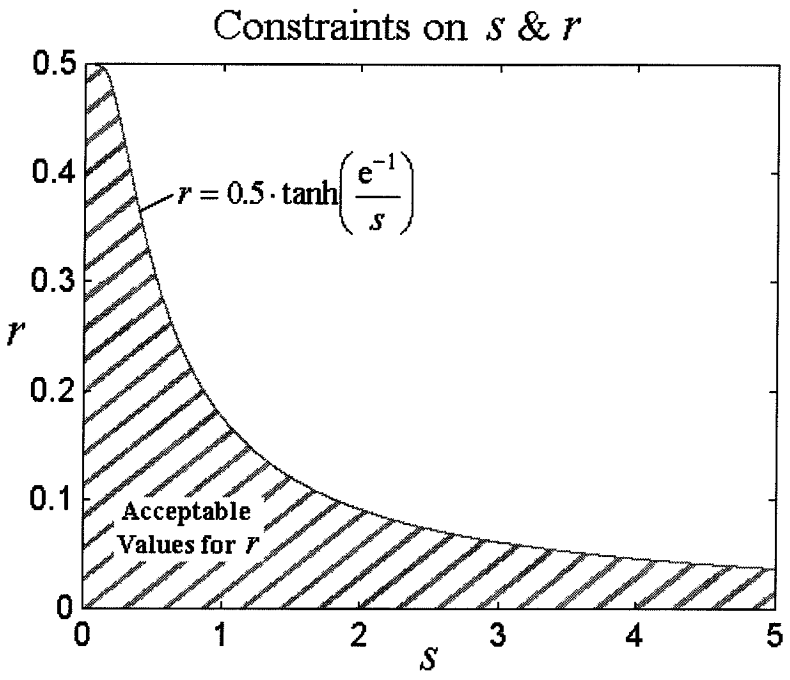


Figure 8.1: **Constraints on $s$ & $r$.** This plot shows the range of acceptable values for $r$ as a function of $s$. Only when $r$ is below the curve can a value for $\bar{\Theta}$ be found such that: (1) the input signal can not create an output spike without there first being a pace spike, $1 \leq \bar{\Theta}$; and (2) there is always an output spike after each pace spike, regardless of the input signal, $\bar{\Theta} \leq \left(\frac{1}{r}\right) \cdot \tanh\left(\frac{e^{-1}}{s}\right) - 1$. For values of $r$ exactly on the curve, the only valid choice for $\bar{\Theta}$ is 1. Notice that $r$ is bounded by 0.5 as $s$ approaches 0.

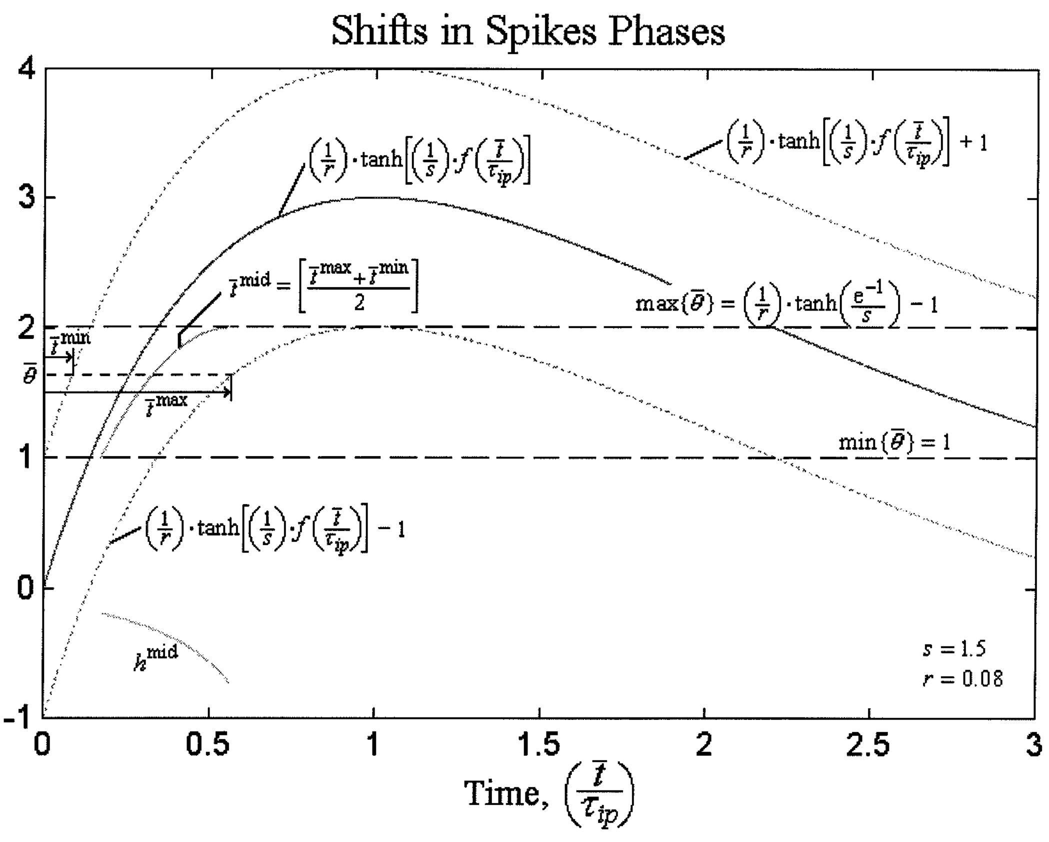


Figure 8.2: **Shifts in Output Spike Phase.** This plot shows the neuron's potential due to a spike from the pace neuron (solid line). Depending upon the sign of the input signal, the total potential will be either raised or lowered by $\tanh\left(h^{\text{in}}\right)$. The dotted lines indicate the maximum and minimum possible potentials. If the neuron is not to fire unless a pace spike has occured, regardless of the input signal, then the threshold, $\bar{\Theta}$, must be at or above 1. And if the neuron is always to fire after a pace spike, regardless of the input signal, then the threshold, $\bar{\Theta}$, must be at or below $\left(\frac{1}{r}\right)\cdot\tanh\left(\frac{e^{-1}}{s}\right)-1$. The dashed lines mark the maximum and minimum allowable levels for $\bar{\Theta}$. For each acceptable value of $\bar{\Theta}$, the actual time of an output spike is bounded between $\bar{t}^{\text{min}}$ and $\bar{t}^{\text{max}}$, with the mid-range value given by $\bar{t}^{\text{mid}} = \left(\frac{\bar{t}^{\text{max}}+\bar{t}^{\text{min}}}{2}\right)$ and $h^{\text{mid}}$ defined as the value of $h^{\text{in}}$ which produces a spike at $\bar{t}^{\text{mid}}$. Notice that $\bar{t}^{\text{max}}$, $\bar{t}^{\text{min}}$, and $h^{\text{mid}}$ depend upon $\bar{\Theta}$, therefore, the curves for $\bar{t}^{\text{mid}}$ and $h^{\text{mid}}$ are not actually functions of time as the plot seems to indicate. Instead, the curve of $\bar{t}^{\text{mid}}$ is drawn as a function of the y-axis, which represents $\bar{\Theta}$. And each value of $\bar{t}^{\text{mid}}$ has a corresponding point on the $h^{\text{mid}}$ curve drawn below it. For this figure, the values of $s$ and $r$ are 1.5 and 0.08, respectively.

latest possible times for the output spikes, which are given by:

$$\frac{\bar{t}^{\min}}{\tau_{ip}} \equiv \frac{\bar{t}\left(h^{\text{in}} = +\infty\right)}{\tau_{ip}} = F^{-1}\left(s \cdot \text{Atanh}\left[r \cdot \left(\bar{\Theta} - 1\right)\right]\right) \geq 0 \tag{8.13}$$

$$\frac{\bar{t}^{\max}}{\tau_{ip}} \equiv \frac{\bar{t}\left(h^{\text{in}} = -\infty\right)}{\tau_{ip}} = F^{-1}\left(s \cdot \text{Atanh}\left[r \cdot \left(\bar{\Theta} + 1\right)\right]\right) \leq 1 \tag{8.14}$$

Notice that $\left[\bar{t}^{\max} - \bar{t}\left(h^{\text{in}} = 0\right)\right] \neq \left[\bar{t}\left(h^{\text{in}} = 0\right) - \bar{t}^{\min}\right]$. Thus, the effects from an excitatory versus an inhibitory stimulus are asymmetric. It is useful to determine the value for $h^{\text{in}}$ which produces the mid-range phase shift value; i.e. let:

$$\begin{aligned} \bar{t}^{\text{mid}} &\equiv \frac{\bar{t}^{\max} + \bar{t}^{\min}}{2} \\ &= \left(\frac{\tau_{ip}}{2}\right) \cdot \left\{F^{-1}\left(s \cdot \text{Atanh}\left[r \cdot \left(\bar{\Theta} - 1\right)\right]\right) + F^{-1}\left(s \cdot \text{Atanh}\left[r \cdot \left(\bar{\Theta} + 1\right)\right]\right)\right\} \end{aligned} \tag{8.15}$$

Now, if $h^{\text{mid}}$ represents the value of $h^{\text{in}}$ which causes the neuron to produce an output spike at time $\bar{t}^{\text{mid}}$, $\left(\bar{t}\left(h^{\text{in}} = h^{\text{mid}}\right) = \bar{t}^{\text{mid}}\right)$, then:

$$h^{\text{mid}} = \text{Atanh}\left(\bar{\Theta} - \left(\frac{1}{r}\right) \cdot \tanh\left[\left(\frac{1}{s}\right) \cdot f\left(\frac{\bar{t}^{\text{mid}}}{\tau_{ip}}\right)\right]\right) \tag{8.16}$$

Here, $h^{\text{mid}}$ can be thought of as representing the offset signal needed to symmetrically encode the strength of the input signal within the phase of an output spike.

The actual phase shift of an output spike is given by:

$$\begin{aligned} z(h^{\text{in}}) &\equiv \frac{\bar{t}\left(h^{\text{in}} = 0\right) - \bar{t}\left(h^{\text{in}}\right)}{\tau_{ip}} \\ &= F^{-1}\left(s \cdot \text{Atanh}\left(r \cdot \bar{\Theta}\right)\right) - F^{-1}\left(s \cdot \text{Atanh}\left[r \cdot \left(\bar{\Theta} - \tanh\left(h^{\text{in}}\right)\right)\right]\right) \end{aligned} \tag{8.17}$$

Notice that $z$ only depends upon the frequency of the pace spikes, $\frac{1}{\Delta_p}$, through $s$. Thus, as along as $\kappa\left(\frac{\Delta_p}{\tau_{ip}}\right) \approx 1$, $(\Delta_p \gg \tau_{ip})$, the input frequency from the pace neuron may change without affecting the phase encoding (provided the approximation in Equation (8.2) still holds).

The asymptotic limits on $z$ are:

$$z(\pm\infty) = F^{-1}\left(s\cdot\text{Atanh}\left(r\cdot\bar{\Theta}\right)\right) - F^{-1}\left[s\cdot\text{Atanh}\left(r\cdot\left(\bar{\Theta} \mp 1\right)\right)\right] \tag{8.18}$$

Also, using the relationship of:

$$\text{if:} \quad y = F^{-1}(q(x)) \tag{8.19}$$

$$\text{then:} \quad \frac{dy}{dx} = \left(\frac{e^{y}}{1-y}\right)\cdot\frac{dq}{dx} \tag{8.20}$$

the derivative of $z$ with respect to $h^{\text{in}}$ is given by:

$$\frac{dz}{dh^{\text{in}}} = s\cdot r\cdot\left[\frac{e^{\left(\frac{\bar{t}(h^{\text{in}})}{\tau_{ip}}\right)}}{1-\frac{\bar{t}(h^{\text{in}})}{\tau_{ip}}}\right]\cdot\left[\frac{\text{sech}^2\left(h^{\text{in}}\right)}{\text{sech}^2\left[\left(\frac{1}{s}\right)\cdot f\left(\frac{\bar{t}(h^{\text{in}})}{\tau_{ip}}\right)\right]}\right] \tag{8.21}$$

For all possible value of $\bar{t}\left(h^{\text{in}}\right)$, between $\bar{t}^{\text{min}}$ ($\geq 0$) and $\bar{t}^{\text{max}}$ ($\leq \tau_{ip}$), the derivative of $z$ is positive. Since $z$ is a bounded differential function with a positive derivative everywhere, it is a sigmoid. In fact, by plotting $z$ as a function of $h^{\text{in}}$ for several different sets of parameters, it becomes evident that $z$ can be well approximated by:

$$z(h^{\text{in}}) \approx C\cdot\tanh\left[\gamma\cdot\left(h^{\text{in}} - h^{\text{mid}}\right)\right] - \left[\bar{t}^{\text{mid}} - \bar{t}\left(h^{\text{in}} = 0\right)\right] \tag{8.22}$$

where:

$$\begin{aligned} C &= \frac{z(+\infty) - z(-\infty)}{2} \\ &= \left(\frac{1}{2}\right)\cdot\left\{F^{-1}\left[s\cdot\text{Atanh}\left(r\cdot\left(\bar{\Theta}+1\right)\right)\right] - F^{-1}\left[s\cdot\text{Atanh}\left(r\cdot\left(\bar{\Theta}-1\right)\right)\right]\right\} \end{aligned} \tag{8.23}$$

$$\gamma = \left(\frac{1}{C}\right)\cdot\left.\frac{dz}{dh^{\text{in}}}\right|_{h^{\text{in}}=h^{\text{mid}}} = \left(\frac{s\cdot r}{C}\right)\left[\frac{e^{\left(\frac{\bar{t}^{\text{mid}}}{\tau_{ip}}\right)}}{1-\frac{\bar{t}^{\text{mid}}}{\tau_{ip}}}\right]\left[\frac{\text{sech}^2\left(h^{\text{mid}}\right)}{\text{sech}^2\left[\left(\frac{1}{s}\right)\cdot f\left(\frac{\bar{t}^{\text{mid}}}{\tau_{ip}}\right)\right]}\right] \tag{8.24}$$

*Thus, the phase of an output spike encodes the* $\tanh(-)$ *of an input signal's intensity.* Figure 8.3 shows $z(h^{\text{in}})$ and the approximate $\tanh(-)$ functions for four different values of $\bar{\Theta}$ with $s$ and $r$ fixed. Different choices for $s$ and $r$ yield similar results.

When there is more than one synaptic input, each additional input alters the effective threshold voltage, $\bar{\Theta}$, by $R_{ij}\cdot\tanh\left(h_j^{\text{in}}\right)$; i.e., to determine how the output phase depends upon one of the input signals *while the others are held constant*, Equation (8.3) can be replaced by:

$$\bar{\Theta}^{\text{effective}} = \bar{R}_{ik}\cdot\tanh\left(h_k^{\text{in}}\right) + \left(\frac{1}{r}\right)\cdot\tanh\left[\left(\frac{1}{s}\right)\cdot f\left(\frac{\bar{t}}{\tau_{ip}}\right)\right] \tag{8.25}$$

where:

$$\bar{\Theta}^{\text{effective}} \equiv \frac{\Theta_i}{\sum_j R_{ij}} - \frac{R_{i0}}{\sum_j R_{ij}} - \frac{\sum_{j\neq k} R_{ij}\cdot\tanh\left(h_j^{\text{in}}\right)}{\sum_j R_{ij}} \tag{8.26}$$

$$\bar{R}_{ik} \equiv \frac{R_{ik}}{\sum_j R_{ij}} \tag{8.27}$$

$$r \equiv \frac{\sum_j R_{ij}}{R_{ip}} > 0 \tag{8.28}$$

and all other variables are as previously defined. Thus, by using $\bar{\Theta}^{\text{effective}}$ instead of $\bar{\Theta}$, the output phase can be calculated as a function of *one* of the input signals. It can be shown that as long as constraint Equation (8.9) continues to be satisfied (by $\bar{\Theta}$ not $\bar{\Theta}^{\text{effective}}$, but normalized by $\sum_j R_{ij}$), the composite transfer function can be approximated by:

$$z(h_1^{\text{in}},\ldots,h_N^{\text{in}}) \approx C\cdot\tanh\left(\gamma\cdot\text{Atanh}\left[\sum_j R_{ij}\cdot\tanh\left(h_j^{\text{in}}\right)\right] - O\right) + A \tag{8.29}$$

where:

$$C = \frac{z(+\infty) - z(-\infty)}{2} \tag{8.30}$$

$$A = \frac{z(+\infty) + z(-\infty)}{2} \tag{8.31}$$

$$O = \text{Atanh}\left(\frac{A}{C}\right) \tag{8.32}$$

Here, $z$ is measured relative to $\bar{t}\left(h_1^{\text{in}} = 0,\ldots,h_N^{\text{in}} = 0\right) = F^{-1}\left[s\cdot\text{Atanh}\left(r\cdot\bar{\Theta}\right)\right]$, and $z(\pm\infty)$ is still given by Equation (8.18). ($z(\pm\infty)$ implies that *all* of the $h_k^{\text{in}}$ inputs

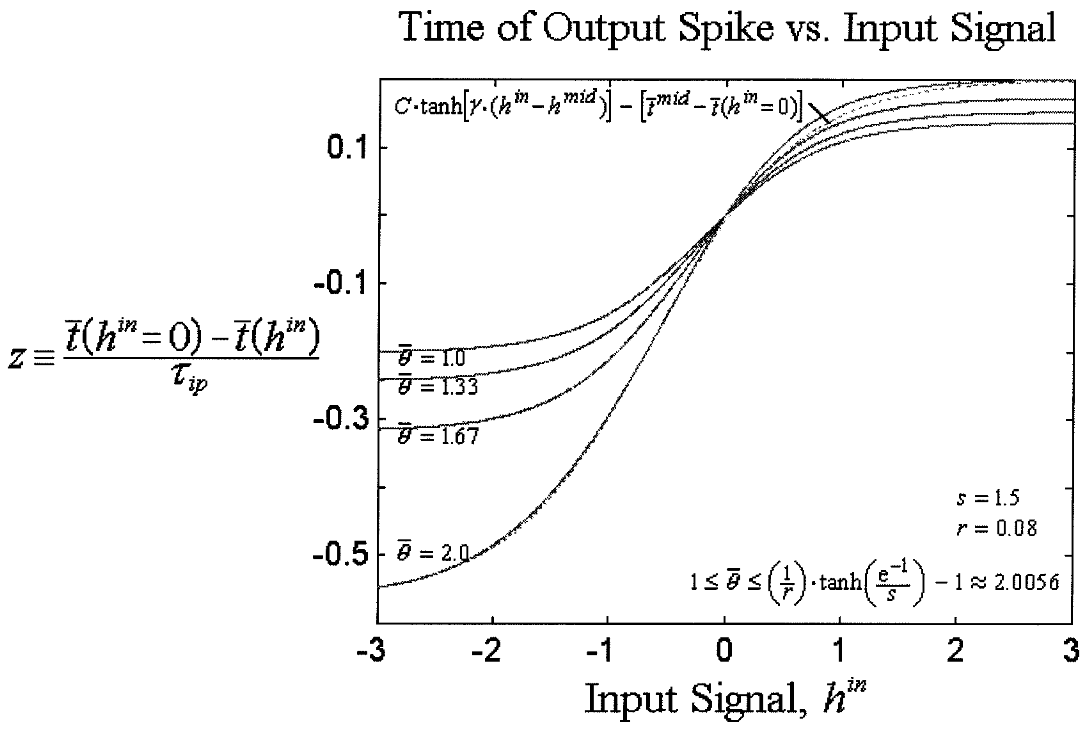


Figure 8.3: **Time of Output Spike vs. Input Signal.** The plot shows how the time of an output spike depends upon the strength of the input signal, for four different values of $\bar{\Theta}$ (1.0, 1.33, 1.67, and 2.0) with $s = 1.5$ and $r = 0.08$. For each of the four $\bar{\Theta}$ values, the corresponding $C{\cdot}\tanh\left[\gamma\cdot\left(h^{\text{in}} - h^{\text{mid}}\right)\right] - \left[\bar{t}^{\text{mid}} - \bar{t}\left(h^{\text{in}} = 0\right)\right]$ function is drawn (dotted lines), but it is nearly indistinguishable from the actual curve except when $\bar{\Theta} = 2.0$. (The maximum error, $\max\left\{\left|z(h^{\text{in}}) - \left[C{\cdot}\tanh\left[\gamma\cdot\left(h^{\text{in}} - h^{\text{mid}}\right)\right] - \left[\bar{t}^{\text{mid}} - \bar{t}\left(h^{\text{in}} = 0\right)\right]\right]\right|\right\}$, for each curve is: 0.00043, 0.00078, 0.0019, and 0.0186.) The values for $C$ are: 0.1701, 0.1991, 0.2468, and 0.4004; and the values for $\gamma$ are: 0.9795, 0.9694, 0.9453, and 0.7696. The values for $h^{\text{mid}}$ are: -0.1945, -0.2400, -0.3211, and -0.6642, which represent the input signals required to produce output spikes at $\bar{t}^{\text{mid}}$ (0.1701, 0.2408, 0.3342, and 0.5384 – relative to the arrival time of the pace spike), which is the average of $\bar{t}^{\text{min}}$ and $\bar{t}^{\text{max}}$ (see Figure 8.2).

are equal to either $+\infty$ or $-\infty$.) The parameter of $\gamma$ can be calculated based upon the mid-range slope of the transfer function given in Equation (8.24), but to do so, it is necessary to know $\bar{t}^{\text{mid}}$ and $h_k^{\text{mid}}$ for each input synapse. ($\bar{t}^{\text{mid}}$ represents the time when all of the inputs are equal to $h_k^{\text{mid}}$. Each $h_k^{\text{mid}}$ represents the mid-range value for the $k^{\text{th}}$ input when all of the other inputs are also equal to their mid-range $h_j^{\text{mid}}$ values.) To determine these values, the initial values for all $h_k^{\text{mid}}$ can be set to one. Then the values for $\bar{t}_k^{\text{min}}$, $\bar{t}_k^{\text{max}}$, and $\bar{t}_k^{\text{mid}}$ with respect to the $k^{\text{th}}$ input can be calculated using:

$$\frac{\bar{t}_k^{\text{min}}}{\tau_{ip}} = F^{-1}\left(s\cdot\text{Atanh}\left[r\cdot\left(\bar{\Theta} - \bar{R}_{ik} - \frac{\sum_{j\neq k} R_{ij}\cdot\tanh\left(h_j^{\text{in}}\right)}{\sum_j R_{ij}}\right)\right]\right) \tag{8.33}$$

$$\frac{\bar{t}_k^{\text{max}}}{\tau_{ip}} = F^{-1}\left(s\cdot\text{Atanh}\left[r\cdot\left(\bar{\Theta} + \bar{R}_{ik} - \frac{\sum_{j\neq k} R_{ij}\cdot\tanh\left(h_j^{\text{in}}\right)}{\sum_j R_{ij}}\right)\right]\right) \tag{8.34}$$

$$\bar{t}_k^{\text{mid}} = \frac{\bar{t}_j^{\text{max}} + \bar{t}_j^{\text{min}}}{2} \tag{8.35}$$

In the equations for $\bar{t}_k^{\text{min}}$ and $\bar{t}_k^{\text{max}}$, all of the inputs are equal to $h_j^{\text{mid}}$ except for the $k^{\text{th}}$ input being considered, whose value is $\pm\infty$. A new estimate for $h_k^{\text{mid}}$ can be calculated from Equation (8.16):

$$h_k^{\text{mid}} = \text{Atanh}\left[\left(\frac{1}{\bar{R}_{ik}}\right)\cdot\left(\bar{\Theta} - \left(\frac{1}{r}\right)\cdot\tanh\left[\left(\frac{1}{s}\right)\cdot f\left(\frac{\bar{t}^{\text{mid}}}{\tau_{ip}}\right)\right] - \frac{\sum_{j\neq k} R_{ij}\cdot\tanh\left(h_j^{\text{in}}\right)}{\sum_j R_{ij}}\right)\right] \tag{8.36}$$

After new estimates are found for all of the $h_k^{\text{mid}}$, the process can be repeated with these values substituted back into Equations (8.33)-(8.34). After a couple iterations, stable values for the $h_j^{\text{mid}}$ emerge, which can then be used to calculate:

$$\frac{\bar{t}^{\text{mid}}}{\tau_{ip}} = F^{-1}\left(s\cdot\text{Atanh}\left[r\cdot\left(\bar{\Theta} - \frac{\sum_j R_{ij}\cdot\tanh\left(h_j^{\text{in}}\right)}{\sum_j R_{ij}}\right)\right]\right) \tag{8.37}$$

Here, $\bar{t}^{\text{mid}}$ is time necessary for evaluating $\gamma$ in Equation (8.24) with $\text{sech}^2\left(h^{\text{mid}}\right) = 1$.

## 8.3.3 Phase Encoding within an Identity Neuron

The previous results were based upon the assumption that with every pace spike, the neuron produces an output spike, regardless of the input signal. (See assumption (1) on page 222.) However, for the synchronized identity and inverse neurons used in stimulus detectors, this is not the case. Both neurons have their thresholds set close to the maximum potential level due to the input pace spikes, and only produce output spikes for excitatory inputs.[3] (In Figure 8.2, $\bar{\Theta}$ is approximately equal to the maximum value of the solid curve.) For this type of neuron, constraint Equation (8.9) becomes:

$$\left(\frac{1}{r}\right)\cdot\tanh\left(\frac{\mathrm{e}^{-1}}{s}\right) - 1 < \bar{\Theta} < \left(\frac{1}{r}\right)\cdot\tanh\left(\frac{\mathrm{e}^{-1}}{s}\right) + 1 \qquad (8.38)$$

When $\bar{\Theta}$ exceeds the upper bound, the neuron's potential can never reach threshold and produce spikes, and when $\bar{\Theta}$ is below the lower bound, the neuron produces an output spike for each pace spike, regardless of the input signal.

Now, assuming that the maximum contribution from the pace signal is higher than that from all possible input signals; i.e.:

$$\tanh\left(h^{\mathrm{in}}\right) < 1 \leq \left(\frac{1}{r}\right)\cdot\tanh\left(\frac{\mathrm{e}^{-1}}{s}\right) \qquad (8.39)$$

then the new relationship between $s$ and $r$ is constrained by:

$$r \leq \tanh\left(\frac{\mathrm{e}^{-1}}{s}\right) \qquad (8.40)$$

Notice that $r$ can be twice as large as shown in Figure 8.1.

The times of the output spikes can still be determined from Equation (8.11). But notice in Figure 8.2 that as $\bar{\Theta}$ increases above $\left(\frac{1}{r}\right)\cdot\tanh\left(\frac{\mathrm{e}^{-1}}{s}\right) - 1$, there is no longer

[3]For the synchronized identity neuron, $\Theta_i$ is slightly above the maximum level, so no output spike is produced without an excitatory input. And for the synchronized inverse neuron, $\Theta_i$ is slightly below the maximum level, so an output spike is produced without any input signal. The general response of both the identity and inverse neurons to input signals is the same: an excitatory input results in an output spike leading the pseudospike and an inhibitory input prevents any output spikes from being produced. The only significant difference is the output response with no input signal.

a value for $\bar{t}^{\max}$. Instead, there is a finite value for $h^{\text{in}}$ at which the maximum of the neuron's potential exactly reaches threshold. For values of $h^{\text{in}}$ below this level, no spikes are produced. Thus, the phase of the output spikes is only defined when:

$$h_o \equiv \text{Atanh}\left[\bar{\Theta} - \left(\frac{1}{r}\right)\cdot\tanh\left(\frac{\text{e}^{-1}}{s}\right)\right] < h^{\text{in}} \tag{8.41}$$

Here, $h_o$ is the value for $h^{\text{in}}$ which produces an output spike at $\frac{\bar{t}}{\tau_{ip}} = 1$.[4]

Because the phase is no longer defined for all possible inputs, it can no longer be approximated by the tanh $(-)$ function in Equation (8.22). In fact, when $h^{\text{in}}$ is at its minimum allowed value, the derivative of the phase is infinite. But, as long as $\bar{\Theta}$ is below the upper limit given in Equation (8.38), there is still a value for $\bar{t}^{\min}$; i.e., as $h^{\text{in}}$ approaches infinity, the phase asymptotically approaches $\bar{t}^{\min}$, which is given by Equation (8.13).

Since when $\bar{\Theta}$ is above $\left(\frac{1}{r}\right)\cdot\tanh\left(\frac{\text{e}^{-1}}{s}\right)$, there are no output spikes with $h^{\text{in}} = 0$, it no longer makes sense to define the phase as in Equation (8.17). Instead, the phase can be defined as the time shift relative to 1, which is the output time when $h^{\text{in}} = h_o$, i.e.:

$$\hat{z}(h^{\text{in}}) \equiv 1 - \frac{\bar{t}\left(h^{\text{in}}\right)}{\tau_{ip}} = 1 - F^{-1}\left(s\cdot\text{Atanh}\left[r\cdot\left(\bar{\Theta} - \tanh\left(h^{\text{in}}\right)\right)\right]\right) \tag{8.42}$$

Figure 8.4 shows how this function changes as $\bar{\Theta}$ increases above the critical value of $\left(\frac{1}{r}\right)\cdot\tanh\left(\frac{\text{e}^{-1}}{s}\right) - 1$. If $q$ is defined as the ratio of $\bar{\Theta}$ to this critical value, then as $q$ rises above $\approx 1.15$, $\hat{z}$ can be well approximated by:

$$\hat{z}(h^{\text{in}}) \approx C\cdot\sqrt{\tanh\left[\gamma\cdot(h^{\text{in}} - h_o)\right]} \tag{8.43}$$

where: $$C = \hat{z}(+\infty) = 1 - \frac{\bar{t}^{\min}}{\tau_{ip}} = 1 - F^{-1}\left[s\cdot\text{Atanh}\left(r\cdot\left(\bar{\Theta} - 1\right)\right)\right] \tag{8.44}$$

[4]While the refractory coefficient was approximated by 1 in Equation (8.2), in actuality, it must be below 1. Consequently, $h^{\text{in}}$ will need to be slightly larger than $h_o$ to produce spikes.

$$\gamma = \frac{\text{Atanh}\,(p^2)}{h_p - h_o} \tag{8.45}$$

$$h_p \equiv \text{Atanh}\left(\bar{\Theta} - \left(\frac{1}{r}\right)\cdot\tanh\left[\left(\frac{1}{s}\right)\cdot f\,(1 - p\cdot C)\right]\right) \tag{8.46}$$

Here, rather than setting $\gamma$ so that the slope of the approximation function and derivative of $\hat{z}$ are equal at some point[5], it is chosen so that the approximating function equals $\hat{z}$ at $p\cdot C$, where $C$ is its asymptotic maximum and $p$ can be arbitrarily chosen between 0 and 1. Of course, some values of $p$ produce better approximations than others. For the approximating functions shown in Figure 8.4, $p$ was chosen to be 0.9.

Previously, the end of Section 8.3.2 described the composite effect from multiple input synapses when constraint Equation (8.9) is satisfied. While this constraint is not satisfied for the neurons considered in this section, it would seem that Equation (8.43) could be transformed to accommodate multiple input synapses, in a similar manner to the way Equation (8.22) was transformed into Equation (8.29); however, this is not the case. The problem is that while the effective threshold, $\bar{\Theta}^{\text{effective}}$, given in Equation (8.25) always remains in the $\tanh(-)$ approximating region when Equation (8.9) is satisfied, in general $\bar{\Theta}^{\text{effective}}$ does not remain in the $\sqrt{\tanh(-)}$ approximating region, when constraint Equation (8.38) is satisfied. In fact, it can be shown that the transfer function remains exclusively in the $\sqrt{\tanh(-)}$ region for all possible inputs, only if there are exactly two inputs with equal weights, $R_{i1} = R_{i2}$. When this is not the case, no composite transfer function is possible. Instead, to determine how the output phase depends upon *one* of the input signals, the values for the other inputs must first be assigned. Then $\bar{\Theta}^{\text{effective}}$ can be calculated with respect to the variable input, and depending upon whether it is smaller or larger than $\left(\frac{1}{r}\right)\cdot\tanh\left(\frac{e-1}{s}\right) - 1$, either Equation (8.22) or Equation (8.43) can be used to approximate the output. With different choices for the input values, the approximating region can change between the $\tanh(-)$ and $\sqrt{\tanh(-)}$ regions.

[5]Both $\hat{z}$ and the approximating function already have an equal derivative at $h_o$, namely infinity.

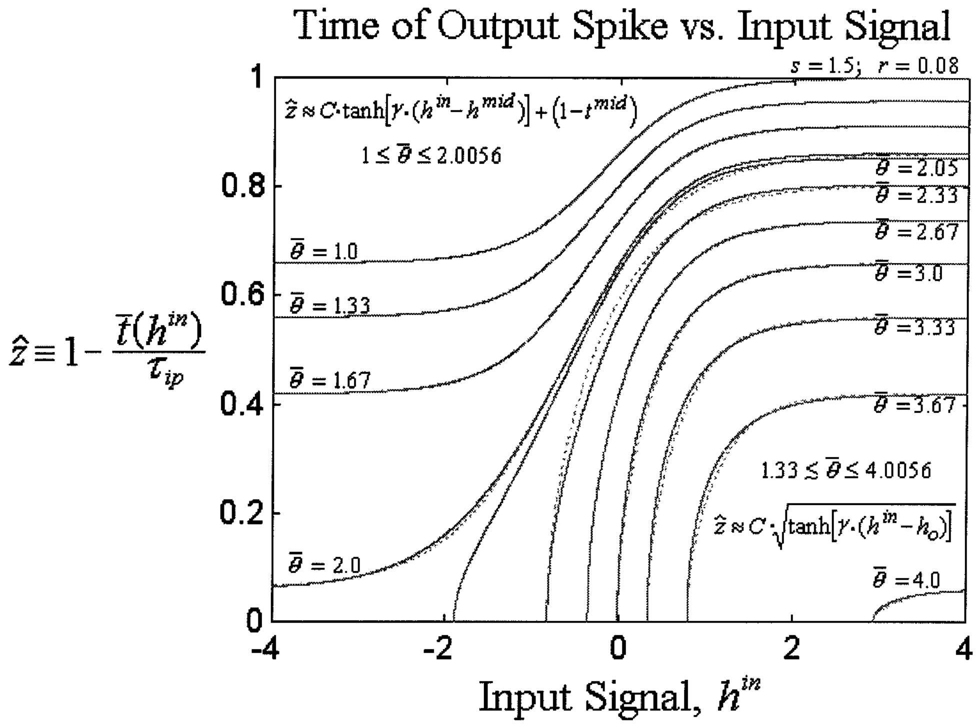


Figure 8.4: **Time of Output Spike vs. Input Signal for Increasing $\bar{\Theta}$.** The plot shows how the time of an output spike depends upon the input signal, for eleven different values of $\bar{\Theta}$ (1.0, 1.33, 1.67, 2.0, 2.05, 2.33, 2.67, 3.0, 3.33, 3.67, and 4.0) with $s = 1.5$ and $r = 0.08$. The first four transfer functions are the same as shown in Figure 8.3, but with the output phase, $\hat{z}$, measured relative to 1, instead of when $h^{\text{in}} = 0$. Thus, all phases are measured relative to the same absolute time. As $\bar{\Theta}$ increases above 2.0056, the shape of the curves abruptly change, with there being a minimum value of $h^{\text{in}}$ for producing an output spike. As $q > 1.15$ the transfer function can be well approximated with the $\sqrt{\tanh(-)}$ function. For the six curves fitted with this function ($\bar{\Theta} = 2.33$ to 4.0), the values for $C$ are: 0.8048, 0.7387, 0.6598, 0.5603, 0.4188 and 0.0612; the values for $\gamma$ are: 0.7466, 0.9447, 1.1003, 1.2299, 1.3338, and 1.3697; and the values for $h_o$ are: -0.8155, -0.3526, -0.0056, 0.3399, 0.7952, and 2.9341. The maximum error, $\max\left\{\left|z(h^{\text{in}}) - C\cdot\sqrt{\tanh\left[\gamma\cdot(h^{\text{in}} - h_o)\right]}\right|\right\}$, for each curve is: 0.0484, 0.0063, 0.0240, 0.0309, 0.0277, and 0.0039.

## 8.3.4 Layered Networks of Phase Encoding Neurons

The previous two sections described how the strength of a constant input stimulus can be encoded within the output spike phase. This section describes how such a phase encoding scheme can be used within a layered network. While the neurons in the first layer may be receiving a constant or slowly varying input signal (see assumption (2) on page 223), the neurons in subsequent layers are receiving the phase encoded output spikes from the previous layer's neurons.

Like the neurons in the first layer, the neurons in the next layers are assumed to receive an input signal from the *same* pace neuron. Since the first layer neurons encode the strengths of their input stimuli within the timing of their output spikes, the times of the output spikes from a second layer neuron must somehow depend upon the times that the input spikes arrive from the first layer neurons. Initially, it is helpful to consider a neuron with a single input signal from one of the neurons discussed in Section 8.3.2, which produces an output spike with every pace spike. Figure 8.5 shows how the neuron's potential is affected by the input spike.

When there is only one input signal, the firing condition becomes:

$$\bar{\Theta} = \tanh\left[W \cdot f\left(\frac{\hat{t}}{\tau_{ij}}\right)\right] + \left(\frac{1}{r}\right) \cdot \tanh\left[\left(\frac{1}{s}\right) \cdot f\left(\frac{\bar{t}}{\tau_{ip}}\right)\right] \tag{8.47}$$

where:

$$W \equiv \frac{T_{ij}}{R_{ij}} \cdot \left(\frac{1}{\kappa\left(\frac{\Delta_p}{\tau_{ij}}\right)}\right) \tag{8.48}$$

$$\frac{\hat{t}}{\tau_{ij}} \equiv \frac{t - t_m^K - d_{ij}}{\tau_{ij}} + \psi\left(\frac{\Delta_p}{\tau_{ij}}\right) \tag{8.49}$$

Notice that $\hat{t}$ represents the time measured relative to the arrival time of the input spike, plus the contribution for the beating behavior, $\tau_{ij} \cdot \psi\left(\frac{\Delta_p}{\tau_{ij}}\right)$.[6] As in Equation

[6]While the pace neuron fires at constant periodic rate, the times of the input spikes from the previous layer may change slightly between cycles due to changes in the input signal; however, since the phase dependence upon the input signal is assumed to be only a minor perturbation of the regularly occurring output spikes, the input can still be well approximated by a $P^1$ signal; i.e., $\Delta_j \approx \Delta_p$. (See Equation (8.8).)

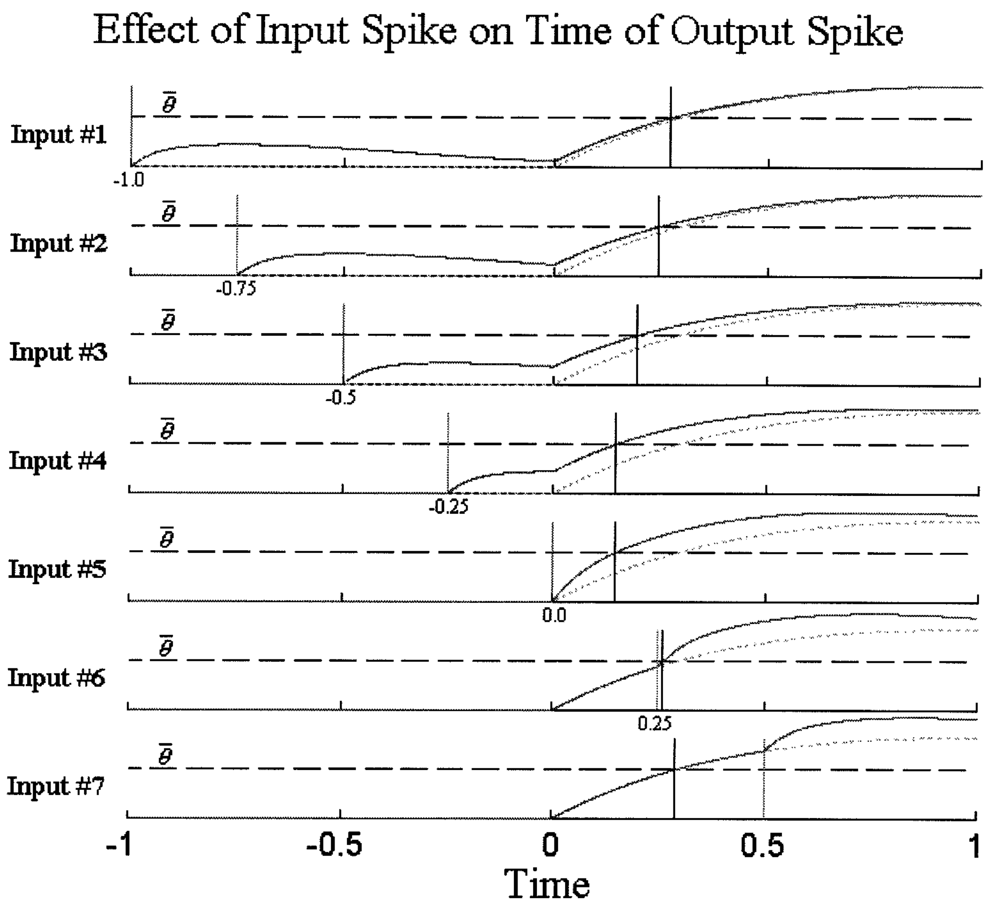


Figure 8.5: **Effect of Input Spike on Time of Output Spike.** The plot shows a neuron's potential as a function of time for seven different input spike arrival times, ranging from -1.0 to 0.5 (relative to the arrival of the pace spike) in steps of 0.25. The parameters are: $s = 1.5$, $r = 0.08$, $\tau_{ip} = 1.0$, $\tau_{ij} = 0.25$, $\bar{\Theta} = 1.8034$, and $W = 3$. The dashed lines represent the threshold level. When the neuron's potential reaches threshold, an output spike is produced. The dotted curves represent the neuron's potential with only the pace spike and no input spike, while the solid curves represent the actual potential. With no input spikes, the neuron fires at 0.2918 after the pace spike arrives. (The resetting effect of the refractory coefficient, which occurs after an output spike is produced, is not included.) Notice that when an input spike arrives much earlier than the pace spike, it has little effect on the time of the output spike. But as the time difference between the input and pace spikes decreases, the output spike occurs earlier. When the input spike arrives just before the neuron is about to fire, it does not significantly contribute to the neuron's potential before the output spike is produced, and the last input spike, at $t = 0.5$, occurs after the output spike.

(8.2), the refractory coefficient is still approximated by 1. The other parameters are defined in Equations (8.4)-(8.8).

While the neurons of the first layer are able to encode both positive and negative signals within their output phase, the input synapse can only be *either* excitatory or inhibitory, depending upon the sign of $W$. If like the first layer neurons, this neuron is to always produce an output spike with each pace spike, regardless of the *time* of the input spike, and can not produce an output spike unless a pace spike is received, then the neuron's parameters must be chosen so:

$$\text{if} \quad W > 0: \qquad \tanh\left(W \cdot \mathrm{e}^{-1}\right) < \bar{\Theta} \leq \left(\frac{1}{r}\right) \cdot \tanh\left(\frac{\mathrm{e}^{-1}}{s}\right) \tag{8.50}$$

$$\text{if} \quad W < 0: \qquad 0 < \bar{\Theta} \leq \left(\frac{1}{r}\right) \cdot \tanh\left(\frac{\mathrm{e}^{-1}}{s}\right) + \tanh\left(W \cdot \mathrm{e}^{-1}\right) \tag{8.51}$$

If there are no input spikes at all (except the pace spikes), the neuron fires at:

$$\left.\frac{\bar{t}}{\tau_{ip}}\right|_{\text{No Spikes}} = F^{-1}\left(s \cdot \text{Atanh}\left(r \cdot \bar{\Theta}\right)\right) \tag{8.52}$$

But if there are input spikes which are arriving just after the neuron fires, i.e., the first layer neuron is firing at a regular frequency but its spikes are arriving slightly after the second layer neuron reaches threshold so the most recent spike contribution to the neuron's potential at the time it fires occured approximately $\Delta_p$ time units ago, then the firing time is given by:

$$\left.\frac{\bar{t}}{\tau_{ip}}\right|_{\text{Late Spikes}} = F^{-1}\left(s \cdot \text{Atanh}\left[r \cdot \left(\bar{\Theta} - \tanh\left[W \cdot f\left(\psi\left(\frac{-\Delta_p}{\tau_{ij}}\right)\right)\right]\right)\right]\right) \tag{8.53}$$

Here, Equation (C.7) was used to simplify the argument of the $f(-)$ function corresponding to the input spikes. Of course, if $\Delta_p$ is large compared to $\tau_{ij}$, then Equation (8.53) can be well approximated by Equation (8.52).

Now, Equation (8.53) represents the critical time before which an input spike must arrive to influence the output time of the spike being produced. Spikes arriving before this time provide a recent contribution to the neuron's potential and tend to make the neuron fire earlier or later, depending upon the sign of $W$. Unfortunately, the exact time of an output spike can only be found be solving the transcendental equation of (8.47) for $t$. (Both $\hat{t}$ and $\bar{t}$ are functions of $t$.) Figure 8.6 shows the time of an output spike as a function of the input spike time for a variety of $W$ values. In general, for positive values of $W$, the output time has a response curve which resembles a radial basis function,[7] with some particular input time producing the earliest possible output spike. Input spikes arriving before or after this maximum response time produce later spikes.

If as in Figure 8.6, all times are measured relative to the arrival time of the pace spike, $t_p^K + d_{ip}$, then the times of the input spike, $t^{\text{in}}$, and output spike, $t^{\text{out}}$, are given by:

$$\begin{aligned} t^{\text{in}} &\equiv \left(t_m^K + d_{ij}\right) - \left(t_p^K + d_{ip}\right) && (8.54) \\ t^{\text{out}} &\equiv t - \left(t_p^K + d_{ip}\right) && (8.55) \end{aligned}$$

Notice that using these definitions and Equation (8.52), the time of an output spike with no input spikes is given by:

$$t_o \equiv \tau_{ip} \cdot \left[ F^{-1}\left(s \cdot \text{Atanh}\left(r \cdot \bar{\Theta}\right)\right) - \psi\left(\frac{\Delta_p}{\tau_{ip}}\right) \right] \tag{8.56}$$

To determine the time of maximum response, it is possible to use Equation (8.47) to solve for the derivative of the output spike time with respect to the arrival time of the input spike, which is:

$$\frac{dt^{\text{out}}}{dt^{\text{in}}} = \frac{\left(\frac{W}{\tau_{ij}}\right) \cdot \text{sech}^2\left(W \cdot f\left(\frac{\hat{t}}{\tau_{ij}}\right)\right) \cdot f\left(\frac{\hat{t}}{\tau_{ij}} - 1\right)}{\left(\frac{W}{\tau_{ij}}\right) \cdot \text{sech}^2\left(W \cdot f\left(\frac{\hat{t}}{\tau_{ij}}\right)\right) \cdot f\left(\frac{\hat{t}}{\tau_{ij}} - 1\right) + \left(\frac{1}{\tau_{ip} r s}\right) \cdot \text{sech}^2\left(\frac{1}{s} \cdot f\left(\frac{\bar{t}}{\tau_{ip}}\right)\right) \cdot f\left(\frac{\bar{t}}{\tau_{ip}} - 1\right)} \tag{8.57}$$

[7]Radial basis functions have a localized bump-like response. Traditionally, when a radial basis function is used in an artifical neural network it is a Gaussian function. The advantage of radial basis functions is that only one layer of hidden units is needed to represent any reasonable function [42, pp. 143, 248-249].

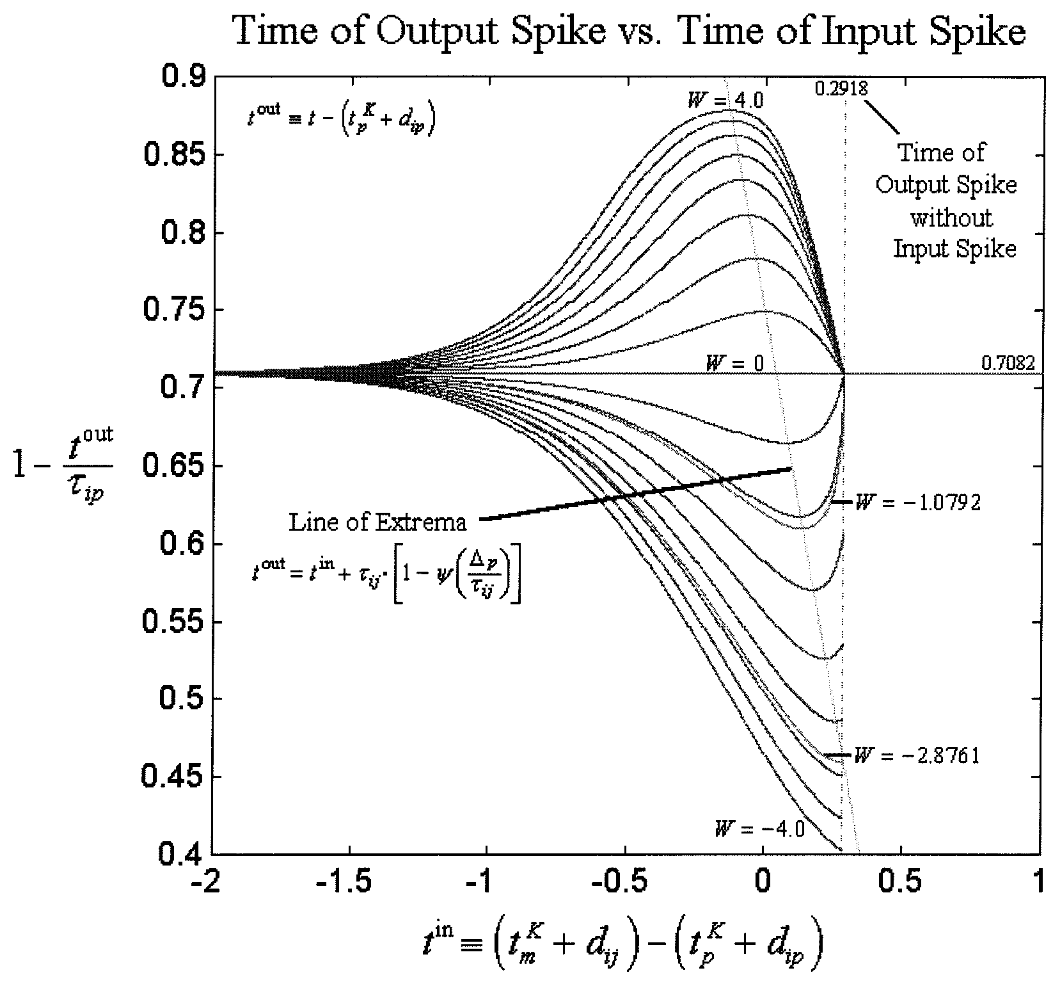


Figure 8.6: **Time of Output Spike vs. Time of Input Spike.** The plot shows how the time of an output spike depends upon the time of an input spike for values of $W$ ranging from -4 to 4 in steps of 0.5. All times are referenced to $\left(t_p^K + d_{ip}\right)$, the arrival time of the pace spike, but as in Figure 8.4, the output phase is measured relative to 1. The output times were determined by numerically solving the transcendental equation of (8.47) for $t$, with $s = 1.5$, $r = 0.08$, $\tau_{ip} = 1.0$, $\tau_{ij} = 0.25$, and $\bar{\Theta} = 1.8034$. The value for $\Delta_p$ was assumed to be large enough so that both $\psi\left(\frac{\Delta_p}{\tau_{ij}}\right)$ and $\psi\left(\frac{\Delta_p}{\tau_{ip}}\right)$ could be approximated by zero. With these parameter values, the time of an output spike with no input spike is 0.2918 (see Equation 8.56). Input spikes arriving after this time, have no influence on the time of an output spike, and the output phase is given by $0.7082 (= 1 - 0.2918)$. For most values of $W$, the extrema in the output phase lies along the line determined by Equation (8.58), but when $W < -2.8761$ (see Equation (8.61)), the extrema occur when the input spikes arrive just as the output spike is about to be produced at $t = 0.2981$. Also, the time of an output spike changes abruptly at $t = 0.2981$ if $W < -1.0792$ (see Equation (8.62)).

This equation implies that there is an extrema when the argument of $f(-)$ is 0; i.e.:

$$t^{\text{out}} = t^{\text{in}} + \tau_{ij} \cdot \left[1 - \psi\left(\frac{\Delta_p}{\tau_{ij}}\right)\right] \tag{8.58}$$

with the extrema for each weight value given by:

$$t^{\text{out}}\Big|_{\text{min,max}} = \tau_{ip} \cdot F^{-1}\left(s \cdot \text{Atanh}\left[r \cdot \left(\bar{\Theta} - \tanh\left[W \cdot f\left(\text{e}^{-1}\right)\right]\right)\right]\right) - \tau_{ip} \cdot \psi\left(\frac{\Delta_p}{\tau_{ip}}\right) \tag{8.59}$$

$$\text{at:} \quad t^{\text{in}} = t^{\text{out}}\Big|_{\text{min,max}} - \tau_{ij} \cdot \left[1 - \psi\left(\frac{\Delta_p}{\tau_{ij}}\right)\right] \tag{8.60}$$

However, for large negative weight values, the extreme value for $t^{\text{out}}$ in Equation (8.59) may occur for input spikes which arrive after the neuron has already fired (see Line of Extrema in Figure 8.6). Thus, there is a minimum value for $W$ for which Equation (8.59) is a solution. If $\Delta_p$ is large compared to $\tau_{ij}$ (Equation (8.53) can be well approximated by Equation (8.52)), then it is possible to solve for the critical value of $W$, which is given by:

$$W^{\text{critical}} = \text{e} \cdot \text{Atanh}\left[\bar{\Theta} - \left(\frac{1}{r}\right) \cdot \tanh\left(\left(\frac{1}{s}\right) \cdot f\left[F^{-1}\left(s \cdot \text{Atanh}\left(r \cdot \bar{\Theta}\right)\right) + \frac{\tau_{ij}}{\tau_{ip}}\right]\right)\right] \tag{8.61}$$

For values of $W$ greater than $W^{\text{critical}}$, the extreme value for $t^{\text{out}}$ is given by Equations (8.59)-(8.60), but for values of $W$ below $W^{\text{critical}}$, the extreme value of $t^{\text{out}}$ occurs when $t^{\text{in}} = t_o$. (See Figure 8.6.)

While Equation (8.61) separates the values of $W$, depending upon where the extrema for $t^{\text{out}}$ lies, there is also a value for $W$ which determines whether or not there is a discontinuity in the transfer function at $t^{\text{in}} = t_o$. This value is given by:

$$W^{\text{discontinuous}} = -\left(\frac{\tau_{ij}}{\tau_{ip} \cdot r \cdot s}\right) \cdot \left[1 - \left(r \cdot \bar{\Theta}\right)^2\right] \cdot f\left[F^{-1}\left(s \cdot \text{Atanh}\left(r \cdot \bar{\Theta}\right)\right) - 1\right] \tag{8.62}$$

When $W$ is above $W^{\text{discontinuous}}$, the times of the output spikes smoothly approach $t_o$ as $t^{\text{in}}$ approaches $t_o$. But for values of $W$ below $W^{\text{discontinuous}}$, the times of the output spikes change abruptly as $t^{\text{in}}$ passes through $t_o$. (See Figure 8.6.)

In general, the phase of an output spike due to the time of an input spike resembles that of a radial basis function. There is a particular time at which the input spike has a maximum effect on the production of an output spike. The value of this maximally effective time depends upon all of the neuron's parameters. However, for most parameters, when the input and pace spikes are closely correlated in time, the neuron tends to fire early. If the input spike comes from a neuron in the first layer which encodes the strength of the input stimulus as $\tanh(-)$, then the synaptic delay between the first and second layer neurons can be used to select the input strength which produces a maximum phase change in the second layer neuron. E.g., while an input stimulus of $h^{\text{in}} = 1.0$ may cause the first neuron to produce an output spike early by 0.2 time units, the delay into the second neuron can be chosen so that this particular lead time in the input spike produces an output spike at the earliest possible time. Thus, the second layer neurons can be tuned to respond maximally to a particular input stimulus level into the first layer neurons.

When a neuron has multiple input synapse, each input spike has a similar radial basis transfer function associated with it; however, the maximally effective time for one input spike depends upon the arrival times for the other input spikes; i.e., the transfer functions are not independent of each other. But as long as the neuron still can not fire without a pace spike, each of the inputs can only be a perturbation of the neuron's potential which is primarily determined by the contribution from the pace spike. Thus, the transfer functions only weakly depend upon each other, and can still be well approximated by the results derived for a single input.

## 8.4 Logarithmic Encoding

The previous section described a method for encoding the strength of a stimulus in the phase of a neuron's output spikes, where the neuron's transfer function is well approximated by either the $\tanh(-)$ or $\sqrt{\tanh(-)}$ functions. However, these functions are not ideally suited as a preprocessing stage for networks which rely on the timing between spikes on different input lines, such as the spatial pattern detector. As an example, consider the problem of recognizing an odor in the olfactory system, where the odor is composed of two distinct chemical receptors with a ratio of 5:1. Assume that this ratio is required for the particular odor being considered, and different ratios correspond to different odors. Ideally, the odor could be recognized by the spatial pattern detector discussed in Section 7.3.3, with two *synchronized* input lines. The detector would respond only if the time delay between the input spikes on the two lines corresponded to the targeted ratio. But when only a phase encoding scheme is used, the odor can only be correctly recognized over a narrow range of intensities; i.e., if $h_1^{\text{in}}$ and $h_2^{\text{in}}$ are functions of the intensities of the two odor components, then the spatial pattern detector requires:

$$\begin{aligned} &C \cdot \tanh\left(\gamma \cdot h_1^{\text{low}}\right) - C \cdot \tanh\left(\gamma \cdot h_2^{\text{low}}\right) \\ &\quad \approx C \cdot \tanh\left(\gamma \cdot h_1^{\text{in}}\right) - C \cdot \tanh\left(\gamma \cdot h_2^{\text{in}}\right) \\ &\quad \approx C \cdot \tanh\left(\gamma \cdot h_1^{\text{high}}\right) - C \cdot \tanh\left(\gamma \cdot h_2^{\text{high}}\right) \end{aligned} \tag{8.63}$$

where $h_1^{\text{low}} \leq h_1^{\text{in}} \leq h_1^{\text{high}}$ and $h_2^{\text{low}} \leq h_2^{\text{in}} \leq h_2^{\text{high}}$ represents the range of intensities for the components over which the odor should be correctly recognized. Obviously for the detector to distinguish between this odor and others with different ratios, the $\tanh(-)$ functions must be operating in the near-linear region, or:

$$h_1^{\text{low}} - h_2^{\text{low}} \approx h_1^{\text{in}} - h_2^{\text{in}} \approx h_1^{\text{high}} - h_2^{\text{high}} \tag{8.64}$$

The problem is that this equation can only be satisfied over a wide range of input intensities if $h^{\text{in}}$ *logarithmically encodes* the strength; i.e.:

$$\begin{aligned} \text{if:} \quad & h^{\text{low}} = \ln(I_o) \\ \text{and:} \quad & h^{\text{in}} = \ln\left(I^{\text{in}} = A \cdot I_o\right) = \ln(A) + \ln(I_o) = \ln(A) + h^{\text{low}} \\ \text{then:} \quad & h_1^{\text{in}} - h_2^{\text{in}} = \left[\ln(A) + h_1^{\text{low}}\right] - \left[\ln(A) + h_2^{\text{low}}\right] = h_1^{\text{low}} - h_2^{\text{low}} \end{aligned} \tag{8.65}$$

Thus, it is necessary to logarithmically encode the strength of an input signal to preserve the relative timing between spikes, which is necessary for recognition in a pattern detector network.

Hopfield [47] was the first to suggested such a logarithmic encoding scheme using the phase of an output spike, relative to the network oscillations (analogous to the pace spikes used in the stimulus detector). He pointed out that such an encoding can be used by neurons which respond to coincident input spikes, with the time delays associated with the input synapses determining the *relative* strengths necessary for recognition of a pattern (analogous to the spatial recognition neuron). However, he did not present any neuron model or suggest how such a logarithmic encoding could actually be accomplished.

This section shows how with a minor modification to the SNM, it is able to approximate a logarithmic encoding of the input signal in its *output spike frequency.* Since the average input neurotransmitter at a synapse is directly proportional to the input spike frequency, the logarithmically encoded frequency can then be used as the input signal into one of the phase encoding neurons described in Section 8.3. If this neuron is operating within the near-linear region of its transfer function, then its output phase represents the logarithmically compressed input signal. Thus, Hopfield's encoding scheme can be accomplished with two successive neurons.

The required modification to the SNM is to include a true absolute refractory period. Section 4.3.3 described how the SNM presented in Equations (3.1)-(3.4) could be considered to have an absolute refractory period which depended upon the excitatory synaptic weights. But the refractory coefficient, $\tanh^2\left(\frac{t-t_i^K}{\tau_i}\right)$, allowed the neuron to immediately start recovering from an output spike. This was convenient because the neuron dynamics did not have multiple operating regions. But in real neurons, it is likely that there is a true "dead time" after the initiation of an output spike in which the neuron is incapable of producing a spike. This is consistent with experiments in which current is injected directly into a neuron and the neuron is found to produce spikes at a maximum output frequency that does not depend upon the number or strengths of its input connections. For the new SNM, the potential is given by:

$$V_i(t) = \mathcal{R}(t)\cdot\left\{R_{i0} + \sum_{j=1}^{N_i} R_{ij}\cdot\tanh\left(\frac{T_{ij}}{R_{ij}}\cdot U_{ij}(t)\right) + \sum_{x=1}^{P_i} R_{ix}\cdot\tanh\left(T_{ix}\cdot I_{ix}(t)\right)\right\} \tag{8.66}$$

$$\text{where:}\quad \mathcal{R}(t) = \begin{cases} 0 & \text{if } t - t_i^K < D_i \\ \tanh^2\left(\frac{t-t_i^K-D_i}{\tau_i}\right) & \text{if } t - t_i^K \geq D_i \end{cases} \tag{8.67}$$

Thus, $D_i$ is the time after an output spike before the neuron begins to recover.

Now, assume that the neuron is receiving multiple input signals of equal influence, but the number of inputs represents the strength of the odor; e.g., the neuron has a large number of chemical receptors, all of the same type, and each receptor has the same effect on the neuron when the target chemical is received, but the number of active receptors depends upon the odor concentration. (Alternatively, the neuron can be considered a second layer neuron which receives spikes from a large number of sensory neurons, whose output frequency is nearly constant when stimulated.) This assumption allows the neuron's potential to depend linearly upon the strength of the input signal rather than on the tanh (−) of a single input. Setting $R_{i0} = 0$ and normalizing by $\Theta_i$, the firing condition becomes:

$$1 = \mathcal{R}(t)\cdot I^{\text{in}} \tag{8.68}$$

where $I^{\text{in}}$ represents the input signal, which is assumed to be changing on a slower time scale than the time between output spikes. Solving for the time between output spikes gives:

$$\Delta_i = t_i^{K+1} - t_i^K = \tau_i\cdot\text{Atanh}\left(\frac{1}{\sqrt{I^{\text{in}}}}\right) + D_i \tag{8.69}$$

If the output spikes from this neuron, $n_i$, go into a phase encoding neuron, $n_j$, then the average neurotransmitter in the connecting synapse is given by Equation

(4.27), or:

$$\langle U_{ji} \rangle = \frac{\tau_{ji}}{\Delta_i} = \frac{\tau_{ji}}{\tau_i \cdot \mathrm{Atanh}\left(\frac{1}{\sqrt{I^{\mathrm{in}}}}\right) + D_i} \approx \ln\left(I^{\mathrm{in}}\right) \tag{8.70}$$

When the parameters for this equation are chosen appropriately, the average neurotransmitter can be well approximated by $\ln\left(I^{\mathrm{in}}\right)$ over a wide range of input values. This approximation is demonstrated in Figures 8.7 and 8.8. Notice that $h^{\mathrm{in}}$ in the phase encoding approximation of Equation (8.22) equals the input neurotransmitter, $U$, multiplied by the synaptic weight values, (see Equation (8.7)). Thus, the strength of an input signal can first be logarithmically encoded in the amount of neurotransmitter released at a synapse, before being converted into an appropriate phase delay, suitable for the pattern recognition networks.

Finally, for some problems it is useful to encode the duration of a stimulus [48]. If there are two logarithmic encoding neurons, but with slightly different values for $D_i$,[8] then the duration of a stimulus may be encoded in the instantaneous phase delay between their output spike trains; i.e., assume that the times between output spikes from neurons $n_i$ and $n_j$ are given by:

$$\Delta_i = \tau \cdot \mathrm{Atanh}\left(\frac{1}{\sqrt{I^{\mathrm{in}}}}\right) + D_i \tag{8.71}$$

$$\Delta_j = \tau \cdot \mathrm{Atanh}\left(\frac{1}{\sqrt{I^{\mathrm{in}}}}\right) + D_j \tag{8.72}$$

Now, if the input signal starts suddenly, but remains constant, then both neurons will output an initial spike at approximately the same time (assuming neither neuron fired recently, and their refractory coefficients are approximately 1), but the difference in the times of the $N^{\mathrm{th}}$ spikes is:

$$d^N = t_i^N - t_j^N = (N-1)\cdot(\Delta_i - \Delta_j) = (N-1)\cdot(D_i - D_j) \tag{8.73}$$

[8]The choice for $D_i$ determines how well the neuron is able to approximate the logarithmic transfer function. Therefore, both neurons can not accurately perform the logarithmic encoding of stimulus strength.

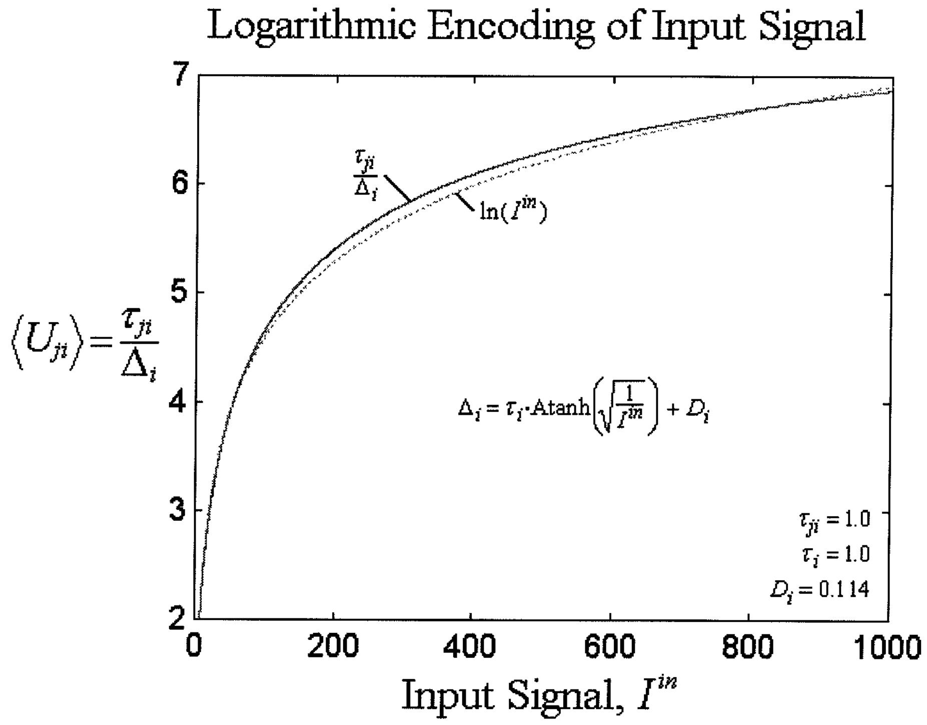


Figure 8.7: **Logarithmic Encoding of Input Signal.** The plot shows how the average neurotransmitter (solid line) at a neuron's output synapse can be made to closely approximate a logarithmic encoding of the neuron's input signal (dotted line). Both the neuron's time constant, $\tau_i$, and the output synapse's time constant, $\tau_{ji}$, were set to unity. The neuron's absolute refractory time, $D_i = 0.114$, was chosen to minimize the error in the logarithmic approximation for values of $I^{\text{in}}$ between 10 and 1000, (maximum error = 2.37% at $I^{\text{in}} = 15.95$.). Different values may be used to improve the approximation over different input ranges. However, a reasonable approximation can only be achieved over 2-3 orders of magnitude, and for values of $I^{\text{in}}$ less than 5, the percentage error tends to become significant, regardless of the parameters used. However, at $I^{\text{in}} = 1$, both the logarithm and neurotransmitter equal zero. For $I^{\text{in}}$ between 10 and 1000, the time between spikes, $\Delta_i$ varied from 0.1456 to 0.4415. Since $\Delta_i$ is relatively small, the actual neurotransmitter level has little variation for constant input signals (see Figure 4.2), and the constraint of a constant input signal into the phase encoding neuron is well satisfied. (The maximum variation between max$\{U_{ji}\}$ and min$\{U_{ji}\}$ over the targeted range was only 2.42% of $\langle U_{ji} \rangle$.)

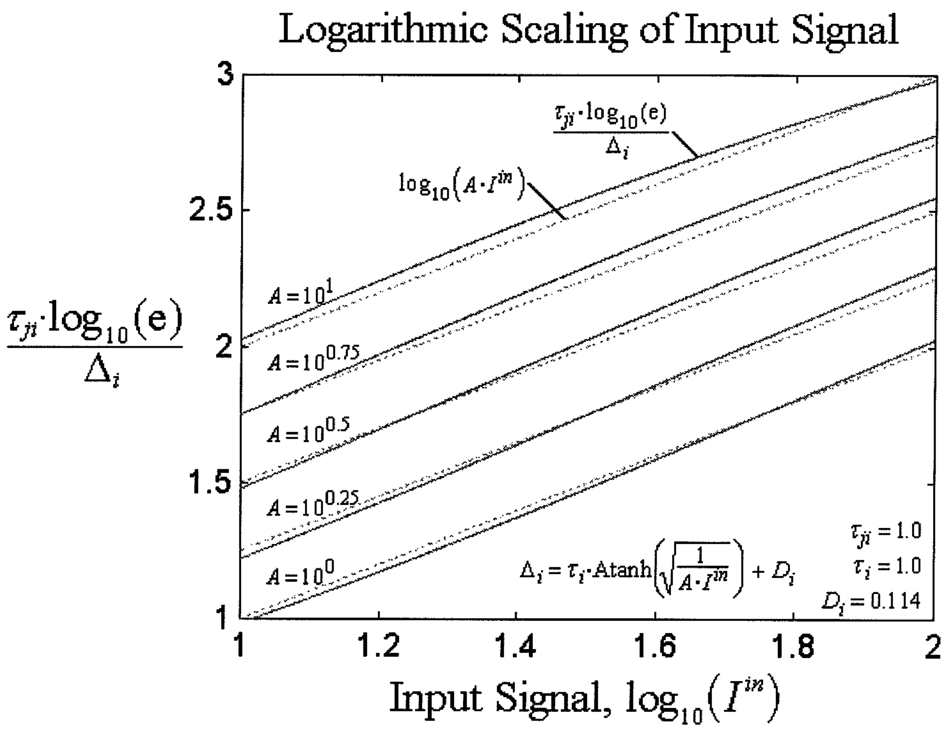


Figure 8.8: **Logarithmic Scaling of Input Signal.** This plot is the same as Figure 8.7, but drawn on a log-log plot with five different scaling coefficients. The input signal ranged from 10 to 100 (1 to 2 on a $\log_{10}$ scale), and the output signal was scaled so that the slope was 1. The purpose of a logarithmic encoding scheme is to convert multiplicative factors of input intensity into addition terms, (see Equation (refeq: log diff for odor)). This plot demonstrates how the curves are approximately increased by a constant amount for each of the different values of $A$. The values for $A$ were chosen to be equally spaced on a $\log_{10}$ scale, with the maximum effective input, $A \cdot I^{\text{in}}$, limited to 1000.

Here, $D_i$ is assumed to be larger than $D_j$. Therefore, with each pair of output spikes, the time difference between the output spikes grows by a fixed amount, namely $D_i - D_j$, independent of the input intensity. This makes it is possible to use the time delay between the output spikes to calculate $N$ and estimate the time since the stimulus began. Notice that this estimate can be made without needing to count the output spikes. Of course, the estimate for $N$ will only be accurate if the phase difference is less than one period (of the fastest oscillator), or:

$$N < 1 + \frac{\Delta_j}{D_i - D_j} \tag{8.74}$$

When $N$ exceeds this limit, it is necessary to count the output spikes from each neuron to know which pairs of spikes correspond. Of course, the smaller the difference between $D_i$ and $D_j$, the larger $N$ can be before this becomes a problem. But if $(D_i - D_j)$ is small, short duration times become difficult to accurately determine; however, it is easy to imagine using several pairs of neurons with different values for $(D_i - D_j)$. Pairs with larger differences can be used to measure short duration times, but as $N$ exceeds the limit in Equation (8.74), the pairs with smaller differences can be used. Figure 8.9 shows how the phase difference increases during the length of the stimulus.

## 8.5 Summary of Information Encoding Schemes

This chapter discussed how information can be encoded within spikes when using the SNM. The chapter started by discussing the format used for the input/output spikes of the networks presented in Chapter 7. Next, Section 8.3 showed how information can be encoded in the phase of a output spike from a neuron which is receiving an oscillatory pace signal. If the contribution from the pace spike is sufficiently strong to guarantee that an output spike is produced regardless of the input signal, then the input signal's strength is encoded within the output phase as tanh $(-)$ (Section 8.3.2). But if the contribution from the pace spike is not sufficiently strong, as with the

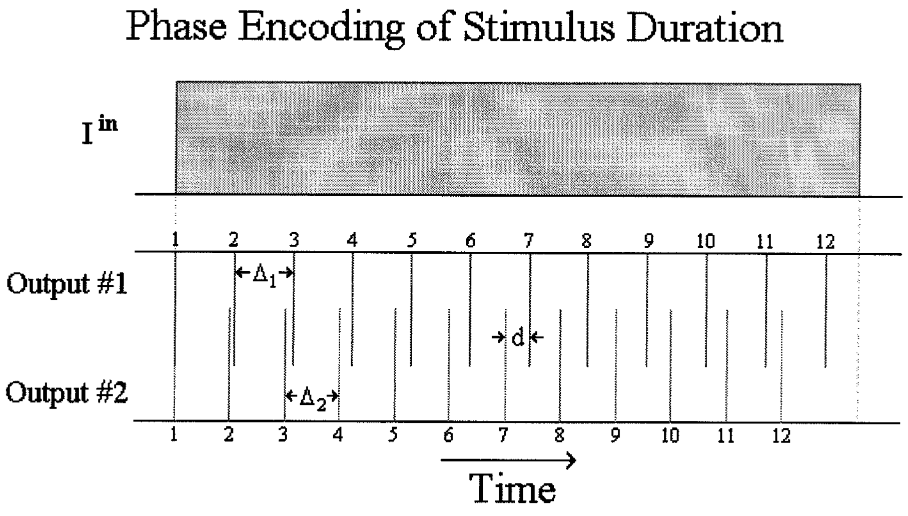


Figure 8.9: **Encoding of Stimulus Duration.** The plot shows how the duration of a stimulus can be encoding in the phase difference between two neurons with slightly different values for $D_1$ and $D_2$. Both neurons are assumed to produce an initial spike when the stimulus first begins, and then with each subsequent pair of output spikes, the time difference, $d$, increases by a fixed amount, $D_1 - D_2$. For this plot, $D_1 = 0.125$ and $D_2 = 0.11$, while $\tau = 1$ for both neurons.

synchronized inverse and identity neurons, then the output phase varies with the strength of the input signal as $\sqrt{\tanh(-)}$ (Section 8.3.3). Neurons which receive a phase encoded spike from a previous neuron, have a transfer function that resembles a radial basis function, where the time of the input spike determines the time of the output spike with a localized bump-like response, (Section 8.3.4). Thus, layered networks can be constructed from neurons which transmit information in the output spike phase. The neurons in the first layer use a sigmoidal transfer function, while the neurons in the subsequent layers use a radial basis transfer function.

Section 8.4 described how the strength of an input stimulus can be logarithmically encoded in the amount of neurotransmitter released at a synapse. Such neurons can be used for sensory inputs, and their output spikes can connect to phase encoding neurons, which convert the firing rates into phase differences. As long as the phase neurons are operating in the near-linear region of their transfer functions, the phase difference remains nearly constant for inputs with a constant strength ratio, regardless of the signals' actual magnitudes. (The magnitude is still logarithmically retained in the phase difference between the output spikes and the pace spike, but the ratio between inputs is encoded in the phase difference between output spikes.) These spikes can then be input into one of the spatial pattern recognizing networks of Section 7.3.3 to respond to inputs signals with a particular ratio. Thus, the network is able to recognize patterns based upon their relative input strengths, over a wide range of strength magnitudes. Also, if there are two sensory neurons receiving the same input signal, but they have slightly mismatched parameters, then the time duration of the signal is linearly encoded within the phase difference between their output spikes, while the strength of the input signal is still logarithmically encoded in their firing rates.

# Chapter 9 Cognitive Learning Methods

## 9.1 Types of Learning

Much of the popularity for neural networks lies within their touted ability to "learn." In a general sense, learning simply refers to modifications made in a network to optimize performance. However, in biological organisms, much of the network structure does *not* directly depend upon its interaction with the environment. Although not necessarily mutually exclusive, the following definitions for *non-interactive developmental learning*, *interactive developmental learning*, and *cognitive learning* attempt to distinguish between the different mechanisms believed to drive neural modifications in biological organisms. These learning paradigms are presented in increasing order of their dependence upon the environment.

**Non-Interactive Developmental Learning** – the process by which a nervous systems alters itself, based predominantly on genetics, *largely independent of the organism's external environment.*[1] For example, the pacemaker neurons, which are used to regulate the heartbeat, form very early in development.[2] Obviously, since the brain and sensory neurons have not yet fully developed, the formation of the pacemaker network must be predominantly controlled through genetics. How specific synaptic connections are genetically controlled is a subject of active research and is beyond the scope of this thesis. (See [28, pp. 628-645] for an overview of some of the more popular theories concerning neural pattern formation.)

[1]For developmental learning, the organism is assumed to be within an environment which allows it to flourish. Obviously, factors such as sever nutritional deprivation can affect the proper development of the nervous system.

[2]A heartbeat can be detected in human a fetus only 22 days after conception. However, these contractions are of myogenic origin and the autonomic nervous system, which regulates heartbeat, does not start to develop until the fifth week [76, pp. 294, 398]. Still, the autonomic nervous system is well developed by birth.

**Interactive Developmental Learning** – the processes by which a nervous systems alters itself, based on genetics, *but aided by sensory feedback from the organism's external environment.* This type of learning refers to those behaviors that can not develop fully in utero, primarily because sensory feedback is required to "fine tune" the parameters. For example, walking is a genetically determined behavior, but without any sensory feedback (touch and vision), many organisms are unable to learn to perform this task. Notice that human infants are unable to walk until they are about a year old, while many other mammals, e.g., horses, are able to walk within hours after birth. Humans are unable to walk because their muscular and nervous systems are not sufficiently developed at birth, and the development of these systems is genetically controlled [20, pp. 331-334]. Horses have more developed muscular and nervous systems at birth, but it takes several weeks before they are able to "learn" to walk proficiently without stumbling. To some extent, the difference between the non-interactive and interactive developmental learning is a matter of semantics. While only interactive developmental learning relies on direct sensory feedback from the *environment*, non-interactive developmental learning does not occur in a vacuum. In fact, it relies on feedback from the other systems forming within the developing organism.

**Cognitive Learning** – the processes by which a nervous systems alters its adjustable parameters, based on the organism's interaction with the environment, *largely independent of genetics.* Genetics determines the general location (type of input signals) and structure of the network components, which limits the range of functions that the network is capable of performing, but cognitive learning determines what and how the functions are performed. This type of learning is probably the predominant type of learning that takes place when acquiring advanced mental skills, such as speech. While genetics determines that humans have a predisposition for speech, it is the environment that determines how and if the skill is obtained. Thus, children learn the language of their parents, and children raised with little adult interaction learn few if any communication skills [99, pp. 232-235].

When attempting to build computational devices based on neurobiology, it is necessary to recognize the profound importance of the developmental learning mechanisms, which are the least understood learning paradigms. Genetically controlled development determines the various structures within the brain, and their functions. The brain can not be considered a large, fully-connected network that organizes itself via cognitive learning. Only through billions of years of evolution has biology developed the intricate structure observed in human brains. Thus, neural computationalists must put a great deal of forethought into a network's architecture before using any learning method. Through genetic algorithms it is possible to attempt to develop optimal architectures for accomplishing particular tasks; however, since the overall performance of a network depends upon both the architecture and cognitive learning, it can be very time consuming to compare different network architectures for even very simple tasks [42, p. 157]. The goal in designing the networks presented in Chapters 6 and 7 was to develop networks structures that could be expected to solve specific desired tasks. Some of the networks presented are the result after several iterations of trial and error, and more optimal solutions may yet exist.

This chapter does not deal with the difficult problem of determining the best network architecture to use, but instead only deals with cognitive learning, which modifies the network parameters to perform a predetermined task for which the network architecture has been chosen. However, many of the networks developed in Chapters 6 and 7 probably can not have their parameters determined solely through learning, even when the network architecture is known. For example, this chapter does not consider how a single neuron with an arbitrary feedback connection can learn the correct parameter values to become one of the memory oscillators discussed in Section 6.3. Networks of this type, which are to perform a specific function, are assumed to undergo developmental learning, and the parameters must be chosen a priori by the network designer. In contrast, the pattern recognition and logic evaluating networks of Sections 7.3 and 7.4 have a general function for which they are structured, but the specific patterns or logic expressions to which they respond can be set through cognitive learning methods.

Obviously, the topic of learning is a very broad subject and could become a thesis in itself. Therefore, this chapter does not attempt to discuss this topic in depth, but only reviews some of the more popular learning paradigms and outlines some ideas as to how learning may be implemented within a network of spiking neurons. Section 9.2 presents an overview of learning methods, before Sections 9.3 and 9.4 introduce perceptron learning and backpropagation learning in networks of artificial neurons. Section 9.5 then develops a novel network architecture and learning paradigm for binary threshold neurons, which can be easily extended to networks of the SNM. And finally, Section 9.6 summarizes the learning methods described in this chapter.

## 9.2 An Overview of Learning Methods

Often learning algorithms are categorized as being either supervised or unsupervised. (Some networks, such as counterpropagation networks, use both methods, but for different sets of parameters – see [38].) Supervised learning uses information concerning the task to be accomplished, while unsupervised learning does not. Unsupervised methods rely only on local information, e.g., the correlation in activity between two connected neurons, to adjust the parameters, while supervised methods require some form of global information concerning the desired output and the network's current performance.

Although unsupervised methods are considered by most neuroscientists to be more biologically plausible, their use in computational applications has remained limited. With unsupervised methods, the network organizes itself based on the emergent properties of the training set, but the results are unpredictable in terms of outputs from specific neurons. After being trained, input patterns may be classified according to their degree of similarity, with like patterns activating the same output neurons [109, pp. 211-212]. In biology, unsupervised learning is probably used primarily for "low-level" processing of sensory information; e.g., through competitive learning, neurons in V1, which receive inputs from neurons in the retina, can organize themselves to respond to different orientations of lines [35, p. 109]. Notice that while genetics determines that these neurons should respond to patterns in the visual field, learning can be used to determine the specific target patterns associated with each neuron. And as long as the neurons in the next layers receive inputs from many of the pattern recognizing neurons, it does not matter which particular neurons respond to which patterns.

Supervised learning may be required for "high-level" cognitive functions in which the organism is interacting with the environment and receiving sensory feedback indicating its performance on an attempted task. Supervised learning is usually associated with backpropagation, which uses gradient descent to adjust the parameters in a direction that reduces the difference between the desired output and the actual output. Section 9.4 discussed backpropagation in more detail; however, it is often criticized as being biologically implausible.

The actual parameter modifications discussed in Section 9.5 for the SNM are based on a Hebbian learning paradigm. The term "Hebbian learning" refers to the learning method first suggested by Hebb [37] to account for the adaptability demonstrated in biological nervous systems. Basically, the idea is that if the activities of two connected neurons are correlated over time, then the strengths of their connecting synapses should increase. Conversely, if the neural activities are uncorrelated, then their corresponding synaptic strengths should decrease. Since this formulation is somewhat vague, there are various ways to implement Hebbian learning, and essentially any learning rule that relies on the activity of the pre- and post-synaptic neurons to adjust the synaptic weights is often referred to as a Hebbian learning algorithm [41]. Figure 9.1 shows how Hebbian learning can account for classical conditioning in psychology.

In support of Hebbian learning, synaptic facilitation, which is an increase in the response of a postsynaptic neuron to stimulation, and synaptic depression, which is a decrease in the response of a postsynaptic neuron to stimulation, have been observed, and are believed to depend upon the timing between presynaptic spikes and postsynaptic spikes [55, 10, 11, 56]. While synaptic strength can be modified in many ways, long-term potentiation (LTP), which increases the strength of a synapse,

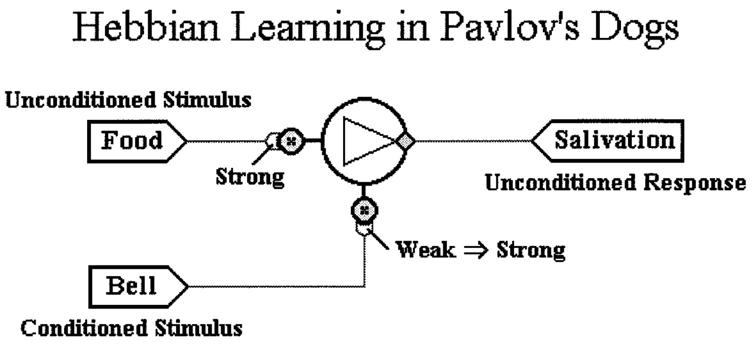


Figure 9.1: **Classical Conditioning through Hebbian Learning.** This figure illustrates how Hebbian learning can be used to explain classical conditioning. In Pavlov's famous dog experiments, he paired a stimulus (food), which would elicit an unconditioned response (salivation), with a stimulus (bell), which did not elicit any response at the beginning of the experiment. After the bell and food had been paired for some time, the dogs would salivate in anticipation of food, when only the bell was rung. The neuron in the figure represents the network that controls salivation, and it is stimulated by the sensory input of food. While the sensory input from the bell also connects to the control neuron, its initial coupling is so weak as not to elicit any response. But when the bell ringing occurs during salivation (due to the food stimulus), the coupling from the bell stimulus increases through Hebbian learning. Eventually the coupling becomes sufficiently strong so that the bell ringing is able to activate salivation without any input from the food stimulus [35, pp. 40-41].

and long-term depression (LTP), which decreases the strength of a synapse, have attracted the most attention since their effects can last for days or weeks [106]. If the presynaptic input spike precedes the postsynaptic depolarization, then the synapse undergoes LTP; but if the timing is reversed, then the synapse can undergo LTD [70]. Of course, for weight changes of this type, it is necessary for each synapse to receive precise information concerning the time that an output spike is produced. There is some evidence that when a neuron reaches threshold, there is not only an action potential along the axon, but also a dendritic pulse that spreads over the dendritic tree to the neuron's input synapses. (See [113, 27] for more details on dendritic pulses and their possible role in learning.)

But while neurobiologists have only recently found evidence in support of Hebbian learning, neural computationalists have developed a variety of learning rules based on the activity of connected neurons. The next couple of sections discuss two of the popular variations: perceptron learning and backpropagation. Section 9.5 then presents a new variant of these for layered networks of threshold units. This new learning paradigm can easily be extended to networks of spiking neurons.

## 9.3 Perceptron Learning

### 9.3.1 Introduction

In 1957, Rosenblatt [94] developed a neural network model, called a perceptron, that stimulated intense interest. In its original form, it was a single layer feedforward network, that was severely limited in its capabilities; however, it has since become the basis for most of today's more sophisticated supervised learning algorithms [109]. In supervised learning, the network has its output compared with known correct answers, and receives feedback concerning its error.

Notice that while some networks may use special feedback connections for learning, these networks should not be confused with fully-recurrent neural networks. Many biological networks have a well defined layered structure, and although feedback connections exist, information is assumed to have a flow direction associated with it, where each successive layer can be described as implementing a higher level task than the one preceding it. An example of a layered network is shown in Figure 7.7, which is used for pattern recognition. Each of the neurons in the first layer has a decision boundary associated with it, and decides which side the input pattern is on. Each neuron in the second layer decides if the input pattern is within its convex target region. The neuron in the third layer determines if the input pattern is within any of the target regions. Thus, information flows from one layer to the next, with each layer accomplishing a more complex task which requires the results from the previous layer.[3]

Figure 9.2 shows a simple perceptron network, based on the artifical neuron shown in Figure 1.1. There is only one layer of neurons, with each of these output neurons receiving connections from all of the network's inputs. The neurons are of the McCulloch-Pitts type, which only produce a non-zero output signal when the weighted sum of the inputs exceeds their threshold level; i.e.:

$$O_i = \begin{cases} 0 & \text{if } \sum_{j=1} W_{ij} I_j < \Theta_i \\ 1 & \text{if } \sum_{j=1} W_{ij} I_j \geq \Theta_i \end{cases} \tag{9.1}$$

Here, $O_i$ is the output of the $i^{\text{th}}$ neuron, and $\Theta_i$ is its threshold value. Often the threshold for all of the neurons in the network is set to zero, and a bias input is used, as shown in Figure 9.2. (A bias is a constant external input of -1 with the adjustable weights from it to the neurons setting their effective threshold levels.) Notice that each output neuron acts independently of the others; consequently, it is only necessary to analyze the behavior of one neuron.

The input weights into a neuron and the input patterns can be written as vectors, e.g., $\vec{W}_i = (W_{i1}, W_{i2}, \ldots, W_{iN})$ and $\vec{I} = (I_1, I_2, \ldots, I_N)$. Thus, the output of the

[3]An apparent example of an unlayered network would be the Hopfield network shown in Figure 7.26, where all of the neurons are fully connected, and there is no well defined flow of information within the network. However, all of the neurons can actually be considered part of the same layer, and the input signals each neuron receives belong to the previous layer.

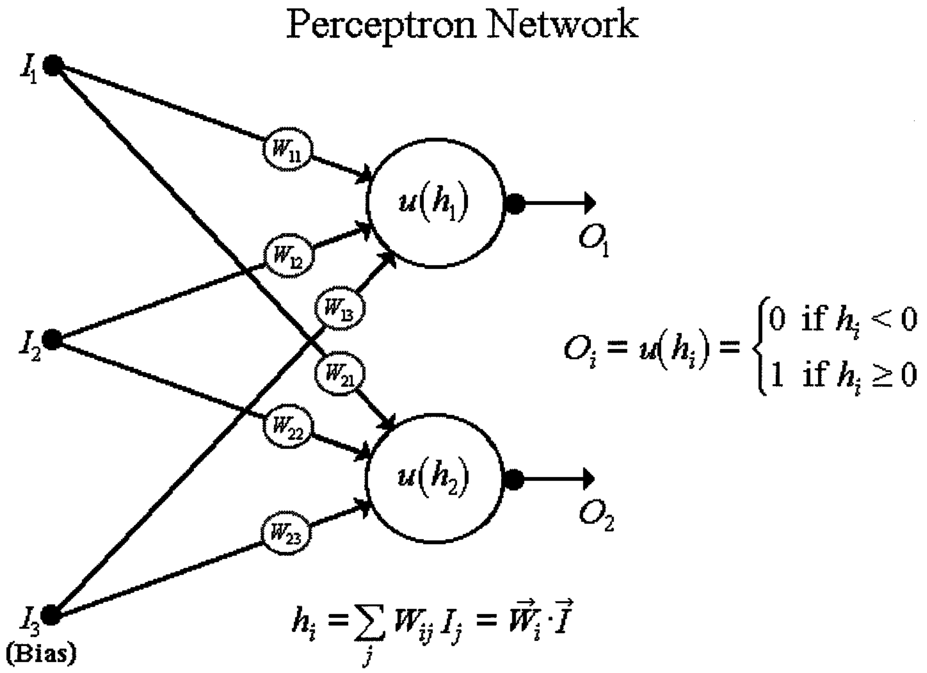

Perceptron Network
$I_1$
$I_2$
$I_3$
(Bias)
$W_{11}$
$W_{12}$
$W_{13}$
$W_{21}$
$W_{22}$
$W_{23}$
$u(h_1)$
$u(h_2)$
$O_1$
$O_2$
$O_i = u(h_i) = \begin{cases} 0 \text{ if } h_i < 0 \\ 1 \text{ if } h_i \geq 0 \end{cases}$
$h_i = \sum_j W_{ij} I_j = \vec{W}_i \cdot \vec{I}$

Figure 9.2: **Perceptron Network.** This figure shows an example of a perceptron network with two output neurons and three external inputs. If the weighted sum of the inputs into a neuron is positive, then the neuron's output is 1, otherwise its output is 0. The third input is the bias unit. It has a constant value of -1, and the weights emanating from it effectively determine the threshold values of the neurons. The interconnection strength from the $j^{\text{th}}$ input to the $i^{\text{th}}$ neuron is labeled $W_{ij}$.

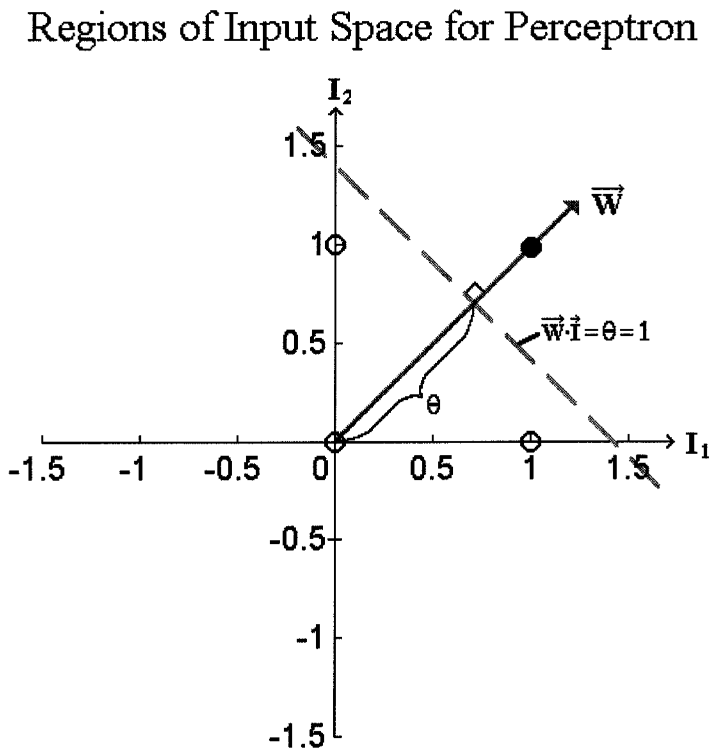

Regions of Input Space for Perceptron
$I_2$
$I_1$
1.5
1
0.5
-0.5
-1
-1.5
-1.5
-1
-0.5
0
0.5
1
1.5
$\vec{W}$
$\vec{W}\cdot\vec{I}=\theta=1$
$\theta$

Figure 9.3: **Regions of Input Space for Perceptron.** This figure shows the input space for a perceptron output neuron. The input space is divided into two regions depending upon whether the projection onto the weight vector, $\vec{W}_i$, is greater than or less than the neuron's threshold, $\Theta_i$, which equals 1 in this figure. Thus, there is a linear boundary determined by the plane of $\vec{W}_i \cdot \vec{I} = \Theta_i$. Also shown are four sample inputs, indicated by circles at $(0,0)$, $(1,0)$, $(0,1)$, and $(1,1)$. Here, the perceptron has been trained to perform the Boolean AND function, with only the projection from the $(1,1)$ input being greater than $\Theta_i$.

neuron given in Equation (9.1) can be rewritten as:

$$O_i = u\left(\vec{W}_i \cdot \vec{I} - \Theta_i\right) \tag{9.2}$$

where $u(-)$ is the unit step function defined in Equation 4.4. (Here, the threshold term has been explicitly included, rather than grouped with the other inputs as the bias signal.) Thus, the weight vector associated with each neuron linearly divides the input space into two regions: (1) those who projection onto $\vec{W}_i$ is greater than $\Theta_i$; and (2) those whose projection onto $\vec{W}_i$ is less than $\Theta_i$. This is shown in Figure 9.3.

Learning is accomplished by training the network on an example set of correct input-output pairs. When one of the training input sets is applied to the network, the correct outputs are compared with network's outputs and the weights are incrementally adjusted to minimize the difference, with only small changes made in response to each training pair. Section 9.3.2 discusses how learning is actually implemented in a perceptron network, but the main limitation of a perceptron network is that the learning task can not be completed if the mapping is not linearly separable. A linearly separable mapping is one in which a hyperplane can be found in the $\vec{I}$ space which divides the inputs with a desired output of 1 from those with a desired output of 0. A perceptron is able to learn the Boolean AND function shown in Figure 9.3; however, it is unable to learn the Boolean XOR function in which the desired output is 1 for inputs of $(1,0)$ and $(0,1)$ but 0 for outputs of $(0,0)$ and $(1,1)$. This problem requires two separating lines, and can only be accomplished with a multilayered network, as discussed in Section 9.4.

## 9.3.2 Perceptron Learning

If a perceptron network is used on a linearly separable training set, the problem becomes one of determining how to update the weights to converge to an appropriate dividing plane. A simple procedure is to present the input patterns one at a time and determine whether each of the neurons is producing the desired output of 0 or 1. If a neuron is producing the desired function, then the connections into it remain unchanged; however, if not, then in the spirit of Hebbian learning, each connection is changed by an amount proportional to the input signal; i.e.:

$$W_{ij}^{\text{new}} = W_{ij}^{\text{old}} + \eta \left(D_i - O_i\right) I_j \tag{9.3}$$

$\vec{W}^{\text{old}}$ refers to the vector of weight values before the adjustment and $\vec{W}^{\text{new}}$ refers to the vector of weight values after the adjustment. The constant $\eta$ is referred to as the "learning rate." It is usually less than 1, and it determines the magnitude of all weight changes. Notice that $(D_i - O_i)$ is either equal to -1, 0, or +1, and a weight is only changed if its corresponding input is active, e.g., $I_j = 1$.

During training, each member of the training set is presented to the network and the weights are readjusted according to Equation (9.3), which is known as the perceptron learning rule [95]. It can be proven that after a finite number of learning iterations the weights will converge to the correct solution, provided that the problem is linearly separable. (See [42, pp. 100-101] for proof.)

It is interesting to notice that while this rule originated as an empirical Hebb assumption, it can be derived through gradient descent with *linear* output units; i.e., let:

$$O_i = \sum_{j=1} W_{ij} I_j \tag{9.4}$$

rather that the threshold output of Equation (9.1). Now, learning can be described as adjusting the network parameters to minimizing the error function of:

$$E = \frac{1}{2} \sum_{i=1} (D_i - O_i)^2 \tag{9.5}$$

Each weight value is changed by an amount proportional to the negative gradient of $E$ at the *present location*[4]; e.g.:

$$\begin{aligned} W_{ij}^{\text{new}} &= W_{ij}^{\text{old}} - \eta \frac{\partial E}{\partial W_{ij}} \\ &= W_{ij}^{\text{old}} + \eta \, (D_i - O_i) \cdot I_j \end{aligned} \tag{9.6}$$

This is the same as Equation (9.3), but for linear units. It is known as the delta rule [96], the adaline or Widrow-Hoff rule [111], and the least mean square rule [42]. The advantage of using the gradient descent approach is that it can be generalized to more than one layer of neurons, provided that they have differentiable transfer functions. With additional layers of *non-linear* neurons, a network is not restricted to problems that are linearly separable. (Multilayered networks of *linear* neurons are

[4]The adjustments can be made after each training pair is presented or after all training pairs have been presented in a process known as batch learning. Since batch learning uses the total from the gradient calculations associated with each training pair, it requires additional storage for each connection. While batch learning reduces the *total* error for the entire training set, the incremental approach seems superior in most cases, requiring less time and getting trapped in local minima less often [42, p. 119].

still not able to solve problems that are not linearly separable.[5]) The backpropagation learning algorithm discussed in the next section applies to multilayered networks of non-linear neurons.

## 9.4 Backpropagation Learning

### 9.4.1 Introduction

Backpropagation learning in multilayered feedforward networks of non-linear neurons is central to much of the current work in artificial neural networks [42, p. 115]. The learning method has been independently developed several times by Bryson and Ho [12], Werbos [110], and Rumelhart et al. [96].

Figure 9.4 shows a typical two layer feedforward network. The adjustable parameters are the interconnections strengths, $W_{ij}^{p}$. Often, $\tanh(-)$ is used as the neuron transfer function, $f(-)$, and any bias values are considered inputs from a constant external input. The output from each neuron is determined by the weighted sum of the inputs from the previous layer; i.e.:

$$V_j^p = \tanh\left(\sum_{k=1} W_{jk}^{p} V_k^{p-1}\right) \tag{9.7}$$

$$\text{where:} \qquad V_k^0 = I_k \tag{9.8}$$

Notice that there are no restrictions on the range of values for the weights, but with a $\tanh(-)$ transfer function, the output of each neuron is bound between -1 and +1.

As with the single layer perceptron network, learning is accomplished by training the network on a set of correct input-output pairs. The correct outputs are compared with network's outputs and the connection strengths, $W_{ij}$, are incrementally adjusted to minimize the error. The direction of the adjustment is determined by calculating the gradient of an error function which measures the difference between the correct

[5]A linear transformation of a linear transformation is just a linear transformation.

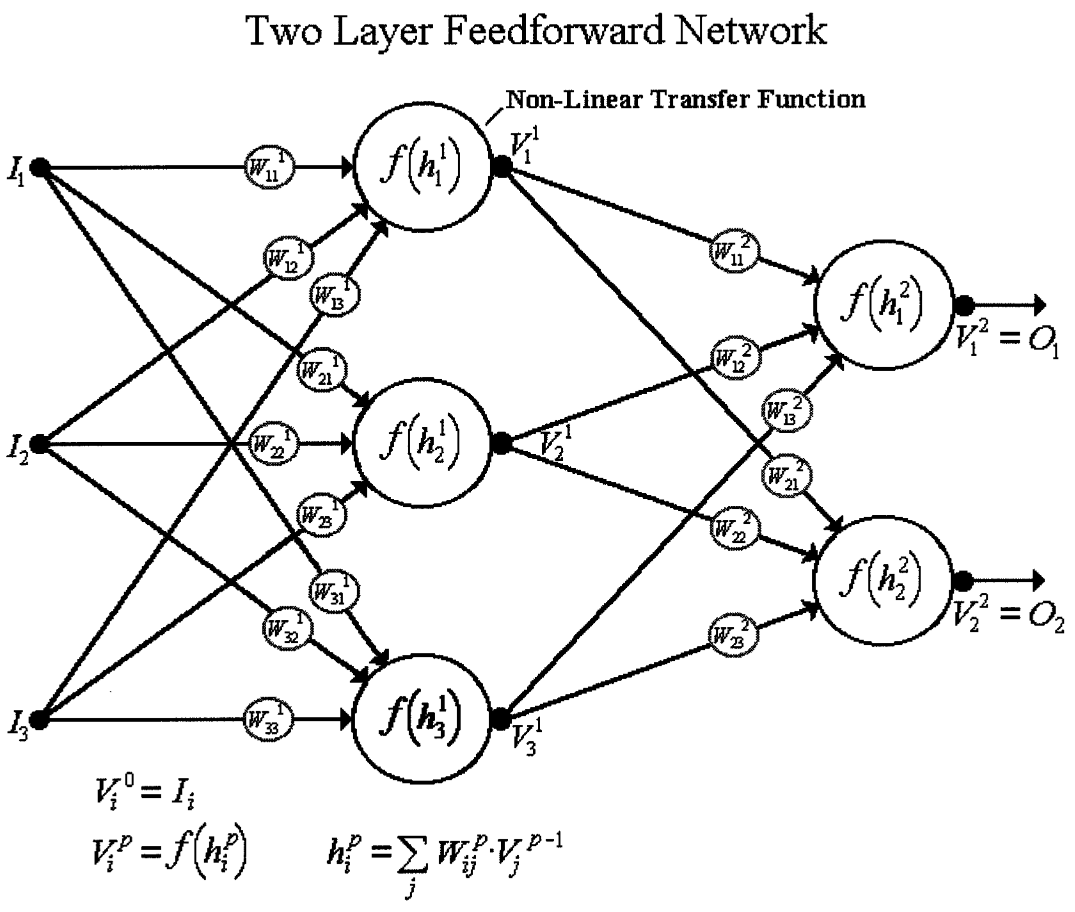


Figure 9.4: **Two Layer Feedforward Network.** This figure shows an example of a two layer feedforward network. The external inputs, $I_k$, shown as solid circles, perform no computation and are not included in the count of layers. Each of the neurons in the network performs a non-linear transformation on the weighted sum of its inputs. The interconnection strength from the $j^{\text{th}}$ input to the $i^{\text{th}}$ neuron of the $p^{\text{th}}$ layer is labeled $W_{ij}^p$. The first layer consists of three neurons which receive inputs from each of three external inputs. Since the outputs from these neurons, $V_j^1$, are only used by other neurons in the network, they are referred to as hidden neurons. The second layer consists of two neurons which only receive inputs from the previous layer. The outputs from these neurons, $V_i^2$, are considered the output signals of the network, $O_i$.

and actual outputs. Ideally, the parameters converge to a solution in which the training set is "known" with high fidelity, and when given inputs not in the training set, the network generalizes to produce reasonable outputs. Section 9.4.2 discusses how learning is actually implemented in a multilayer feedforward network and the problems involved with implementing it in a SNM network.

### 9.4.2 Backpropagation in Multilayered Networks

Backpropagation, like many other learning methods, can be described as minimizing some error function, $E$, which measures the distance between the desired and actual outputs. At each time step during the learning process, the network parameters are incrementally changed in the direction which reduces the error, i.e., by an amount proportional to the negative gradient of $E$:

$$\vec{W}^{\text{new}} = \vec{W}^{\text{old}} + \Delta\vec{W} \tag{9.9}$$

$$\text{where:} \qquad \Delta\vec{W} = -\eta\frac{\partial E}{\partial \vec{W}} \tag{9.10}$$

Here, $\vec{W}$ is used to refer to the interconnection weights, which are assumed to be the only adjustable parameters in the network, however, this may not by the case, and other parameters, such as the neurons' gains, can be adjusted in the same manner by utilizing the gradient of the error function with respect to the parameters being adjusted. Assuming a tanh $(-)$ neural transfer function and a quadratic cost function as given in Equation (9.5), the weight adjustments from the hidden to the output neurons are:

$$\begin{aligned} \Delta W_{ij}^2 &= \eta\left(D_i - O_i\right)\operatorname{sech}^2\left(\sum_{k=1} W_{ik}^2 V_k^1\right) V_j^1 \\ &= \eta\delta_i^2 V_j^1 \end{aligned} \tag{9.11}$$

$$\text{where:} \quad \delta_i^2 = \operatorname{sech}^2\left(\sum_{k=1} W_{ik}^2 V_k^1\right)(D_i - O_i) = \left(1 - (O_i)^2\right)(D_i - O_i) \tag{9.12}$$

(The "2" superscript on $W_{ij}$ and $\delta_i$ refers to the "second" layer parameters, while the "2" superscript on $\text{sech}^2(-)$ indicates that the function is squared. In general, all superscripts on variables in this chapter refer to the neuron layer. When a variable is actually squared, it will be placed within parentheses, as was $O_i$ in Equation (9.12).) The weigth adjustments from the external inputs to the hidden neurons are:

$$\begin{aligned}\Delta W^1_{jk} &= \eta \sum_{i=1} (D_i - O_i)\,\text{sech}^2\left(\sum_{m=1} W^2_{im} V^1_m\right) W^2_{ij}\text{sech}^2\left(\sum_{n=1} W^1_{jn} V^0_n\right) V^0_k \\ &= \eta \delta^1_j V^0_k \qquad (9.13)\end{aligned}$$

where:

$$\delta^1_j = \text{sech}^2\left(\sum_{n=1} W^1_{jn} \cdot V^0_n\right)\left[\sum_{i=1} W^2_{ij}\delta^2_i\right] = \left(1 - \left(V^1_j\right)^2\right)\sum_{i=1} W^2_{ij}\delta^2_i \qquad (9.14)$$

$$V^0_k = I_k \qquad (9.15)$$

Notice that Equation (9.13) has the same form as Equation (9.11), but with a different definition for $\delta$. For a network with a total of $L$ layers, the weight updates are:

$$\Delta W^p_{rs} = \eta \delta^p_r V^{p-1}_s \qquad (9.16)$$

where:

$$\delta^p_r = \text{sech}^2\left(\sum_{j=1} W^p_{rj} \cdot V^{p-1}_j\right)\left[\sum_{i=1} W^{p+1}_{ir}\delta^{p+1}_i\right] = \left(1 - (V^p_r)^2\right)\left[\sum_{i=1} W^{p+1}_{ir}\delta^{p+1}_i\right] \qquad (9.17)$$

$$\delta^L_i = \text{sech}^2\left(\sum_{k=1} W^L_{ik} V^{L-1}_k\right)(D_i - O_i) = \left(1 - (O_i)^2\right)(D_i - O_i) \qquad (9.18)$$

Notice that in Equations (9.14) and (9.17) the $\delta$ of a neuron is determined by the $\delta$'s from the neurons it feeds, with the weight coefficients being the same as the usual "forward" $W_{ij}$'s. Therefore, the same network which computes the output signals can be used to propagate the errors ($\delta$'s) backward, hence the name backpropagation. Figure 9.5 shows the backpropagation connections for the same network shown in Figure 9.4, with the $\delta$'s being the input signals for the "backwards" computation.

(Notice that the summations are over the second index of the weights for the forward computations of Equation (9.7) but over the first index of the weights for the backward computations of Equations (9.14) and (9.17).)

One important consequence of the backpropagation learning rule is that the parameter modifications only rely on *local* information; i.e., after backpropagation of the $\delta$'s, each weight update calculation only needs information which is available at the two ends of the connection. This makes backpropagation appropriate for parallel computation [42, p. 119]. But while locality is necessary for a biological implementation [15], it is not sufficient. As described here, the backpropagation algorithm requires bidirectional, bifunctional connections, which are not biologically reasonable. Hecht-Nielsen [40] has developed a neuron/network model which avoids this problem. A separate type of unit is associated with the weight multiplications; however, the transfer functions associated with these unusual multiplication units (called planets, with the traditional neuron called a sun) are biologically improbable [42, p. 119]. (Essentially, the equations are the same as described above, but distinct units are associated with the different calculations required.)

Aside from the problems with backpropagating the error signal, the main obstacle in attempting to implement backpropagation learning within a SNM network is that the neural transfer functions are not continuous, which prevents a calculation of the error gradients. Of course, the output spike frequencies may be considered the neural output signals, and for neurons operating above threshold, the frequency transfer function is continuous and resembles a sigmoid function (see Section 4.4). Also, if a neuron is already firing and information is being encoded within the times of its output spikes, as suggested in Section 8.3, then the error gradient may be calculated and used to adjust the synaptic parameters which control the relative timing between an input pseudospike from a pace neuron and the neuron's actual output spikes. Thus, for those applications where the neurons' output functions can be continuously defined or at least approximated by a continuous function, backpropagation can be directly applied. However, for some of the applications suggested in Chapter 7, the occurrence or non-occurrence of a single output spike carries information. Section 9.5

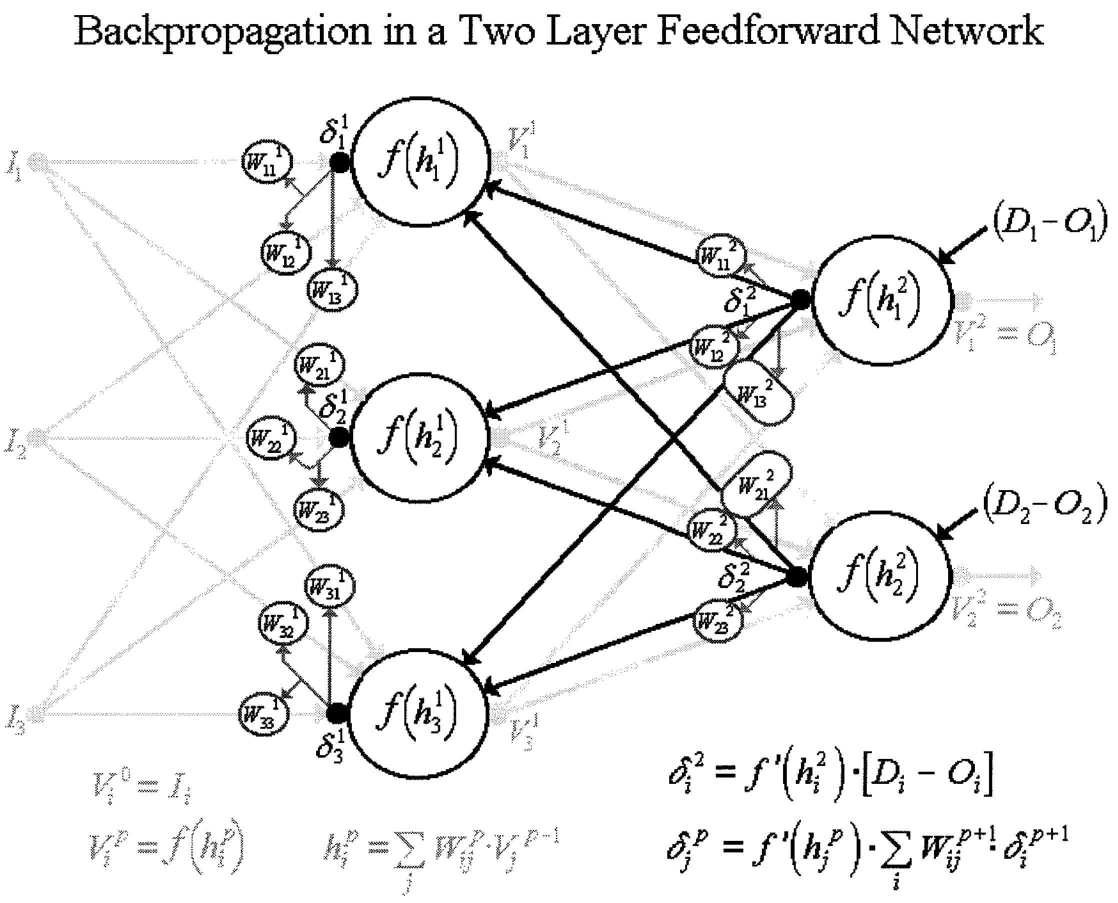


Figure 9.5: **Backpropagation in a Two Layer Feedforward Network.** This figure shows how the same connections used to calculate the output of each neuron can be used to propagate the gradients of the error function back through the network to update the parameters. Each output neuron receives the difference between the desired and actual outputs $(O_i - D_i)$. With this information, the network is able to calculate the appropriate updates for the weights from the first layer (Equation (9.11)), and the $\delta$'s required by the previous layer (Equation (9.14)). Each of these $\delta$'s are multiplied by the same connection weights which were used to propagate the neurons' outputs forward. From the weighted sums of the input $\delta$'s, the neurons in the first layer are able to calculate the appropriate updates for the weights from the external inputs (Equation (9.13)). If the network contains additional layers, the $\delta$'s can continue to be calculated and propagated backward in the same manner (see Equations (9.16)-(9.18)).

discusses how learning may be implemented in networks of this type.

## 9.5 Learning within SNM Networks

### 9.5.1 Introduction

In traditional feedforward networks, each of the neurons in one layer connects to all of the neurons in the next layer, and does not make any connections to neurons in subsequent layers. In adjusting a weight value associated with a connection from an input signal to a hidden neuron, it is necessary to know how that input signal affects the outputs of the network; i.e., does the input have an excitatory or inhibitory effect on each of the outputs? The answer depends upon the neuron transfer function and the sign of the weights associated with the connections from the hidden neuron to the output neurons. The problem with this is that these weight values are also changing during learning and their signs may change during the learning process. Thus, the values of the weights from the hidden neuron to the output neurons must be backpropagated to the synapse from the input to the hidden neuron in order to adjust the weight in a direction which reduces the error in the network's output signal. Also, the output weights from a hidden neuron to the different output neurons can have different signs, thus allowing a hidden neuron to have both an inhibitory and excitatory effect on different output neurons at the same time.

In biology, inhibitory and excitatory synapses differ both chemically and structurally, and it is considered biologically implausible for an inhibitory synapse to become excitatory, or vice versa. Furthermore, experimental evidence indicates that all the output synapses from a neuron are either excitatory or inhibitory (Dale's law – see page 19). Thus, the effect from a neuron can not vary with changes in its input nor between the neurons to which it connects. While in general the SNM allows neurons to form both excitatory and inhibitory synapses, when the neurons are restricted to being either excitatory or inhibitory the learning process can be simplified. Section 9.5.2 discusses the network architecture required to exploit the benefits of having the neurons be either excitatory or inhibitory. Section 9.5.3 then develops a learning algorithm for this particular network, and Section 9.5.4 compares this new learning paradigm with the perceptron and backpropagation learning methods previously discussed. Section 9.5.5 presents some results from when this type of learning was applied to some simple test problems.

## 9.5.2 Network Architecture

Consider the biologically inspired network shown in Figure 9.6. Here, rather than the input signals only connecting to the neurons in the hidden layer, they also connect to the output neuron. The connections from the input signals are only excitatory and the connections from the hidden neurons to the output neuron are only inhibitory. While this network can be used with the SNM, to simplify the analysis it is informative to use the McCulloch-Pitts neuron model, which uses a binary transfer function, i.e., the output of a neuron is either 0 or 1 (-1 for inhibitory neurons) based on the current input signals. Figure 9.7 shows how a network of McCulloch-Pitts neurons can be equated to a network of synchronized spiking neurons, where spikes only occur at discrete times (see Section 7.3).

Mathematically, the neural dynamics for this network can be expressed as:

$$H_k^- = \begin{cases} 0 & \text{if } \sum_{j=1} W_{kj}^1 I_j < \Theta_i \\ -1 & \text{if } \sum_{j=1} W_{kj}^1 I_j \geq \Theta_i \end{cases} \tag{9.19}$$

$$O_i = \begin{cases} 0 & \text{if } \sum_{j=1} W_{ij}^2 I_j + \sum_{k=1} W_{ik}^2 H_k^- < \Theta_i \\ 1 & \text{if } \sum_{j=1} W_{ij}^2 I_j + \sum_{k=1} W_{ik}^2 H_k^- \geq \Theta_i \end{cases} \tag{9.20}$$

where $H_k^-$ is the output from the inhibitory hidden neurons, and $O_i$ is the output of the network. All of the weights are greater than or equal to zero, with $W_{kj}^1$ referring to the input connections into the hidden neurons (first layer), $W_{ij}^2$ referring to the input connections from the inputs into the output neuron (second layer), and $W_{ik}^2$ referring to the input connections from the hidden neurons to the output neuron.

While there is some evidence that biology is able to make minor modifications to

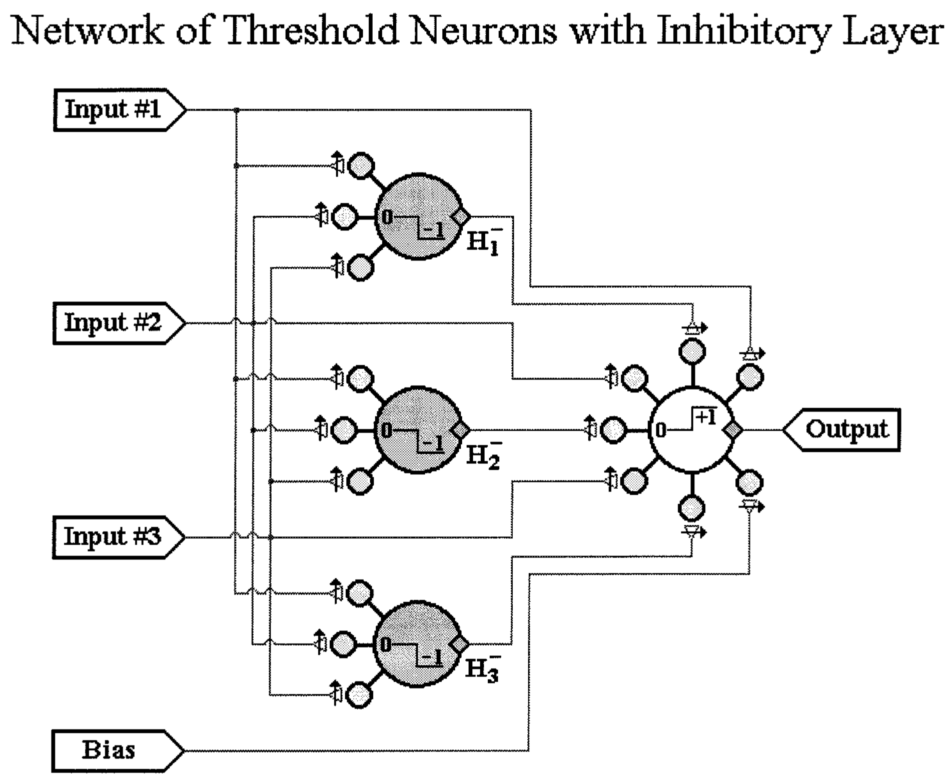


Figure 9.6: **Network of Threshold Neurons with Inhibitory Layer.** This figure shows a network with three inputs, three hidden neurons, and a single output neuron. All of the neurons are of the McCulloch-Pitts type, and only produce a non-zero output signal when the weighted sum of their inputs exceeds their threshold levels. The network's output neuron receives inputs from all of the neurons in both the input and hidden layer. Each of the inputs connects to all of the output and hidden neurons, and all of the connections from the inputs are excitatory. Conversely, all of the connections from the hidden neurons are inhibitory. Thus, each unit can be described as being either excitatory (inputs) or inhibitory (hidden neurons). Consequently, all of the network's interconnection weights can be considered positive, with the inhibitory and excitatory neurons producing outputs of -1 and +1 above threshold. (The colored neurons indicate an inhibitory output signal, and the arrows over the synapses indicate that their strengths vary over integer values during learning.) The output neuron also receives an input connection from a constant +1 source, which is referred to as the bias input.

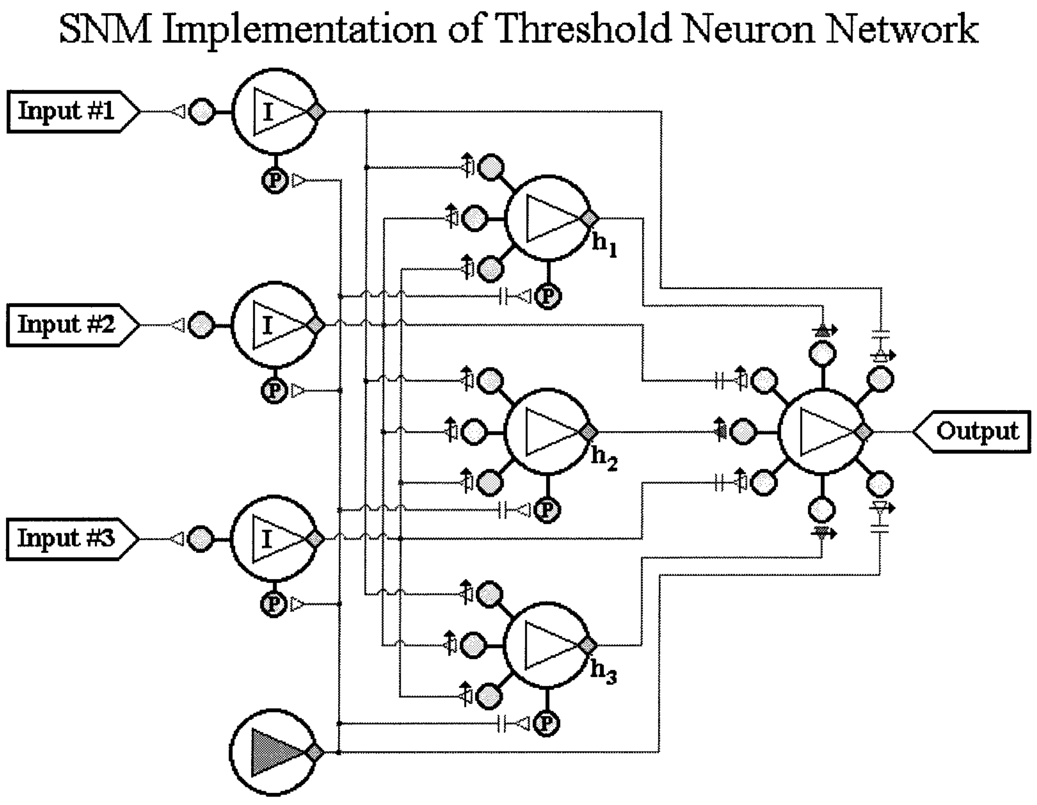


Figure 9.7: **SNM Implementation of Threshold Neuron Network.** This figure shows how the network of threshold neurons in Figure 9.6 can be implemented in a SNM network. The current output of the neuron is described as "1" if it fired during the most recent spike event interval, and "0" if it did not fire. Each of the input signals goes through a synchronized identity neuron. The neurons in the second layer receive synchronized inputs from the first layer, and an input signal from the pace neuron. These neurons act as multiple input identity neurons, and are only capable of producing output spikes at the times when pseudospikes are created from the pace neuron; i.e., the input from the pace neuron is the dominant input signal but it is insufficient to produce output spikes without other inputs. The variable weights from the input signals determine which of the synchronized input neurons must be firing to produce an output spike. Notice that the signals from the pace neuron have delay, which is chosen so that the pseudospikes coincide with the maximum neurotransmitter contributions due to the spikes arriving from the synchronized input neurons. The hidden neurons connect to the output neuron through inhibitory synapses, and the input neurons connect to the output neuron through excitatory synapses. The pace neuron also connects to the output neuron, but unlike its connections to the other neurons, the synaptic weight can vary through learning, which allows it to serve both as a pace neuron and a variable bias input. The delays on the connections from the inputs and pace neuron are set so that their maximum neurotransmitter contributions coincide with the maximum neurotransmitter due to the spikes arriving from the hidden inhibitory neurons.

the synaptic coupling between neurons through LTP and LTD, it appears that most learning occurs through the formation of new synapses and the pruning away of old synapses. Similarly, for this network model, the synaptic strengths will be restricted to positive integer values, e.g., 0, 1, 2, ..., where the weight value can be considered the number of distinct synapses connecting the neurons, and all connections are of equal strength, namely 1. Thus, through learning, new synapses are created and old synapses are destroyed, but each synaptic strength remains constant.

Before developing a learning rule, it is useful to first analyze the type of computations that such a network can perform. If there are $N$ input signals (not including the bias), then the total number of different input patterns is $2^N$. Each of these input patterns can be thought of as one of the vertices on an $N$-dimensional hypercube, i.e., $\vec{I} \in \{0,1\}^N$. Ideally, the output neuron can be trained to produce a spike when any of several target input patterns are presented. Thus, each of the $2^N$ input patterns has a corresponding desired output of either 0 or 1, and the network is to learn to reproduce one of $2^{2^N}$ possible mappings. Figure 9.8 displays the 16 $\left(= 2^{2^2}\right)$ possible mappings for a network with only 2 inputs.

When designing a network of this type, it is necessary to know how many hidden neurons should be used to accomplish the desired mapping. It has been experimentally verified that when there are 2 inputs (16 possible mappings), only one hidden neuron is necessary, and when there are 3 inputs (256 possible mappings), only two hidden neurons are necessary. But while these represent the minimum number of hidden neurons required, additional hidden units have proven useful for reducing the probability of becoming trapped in a local minimum, which prevents the network from learning the complete mapping. Therefore, when this type of network was tested, the number of hidden neurons was set equal to $N$, and while this was one more neuron than needed, the network often converged to a solution which made use of all of the hidden neurons. For problems with a large number of inputs, it is probable that adding even more hidden neurons will improve the learning rate and avoid the

Possible Target Patterns for Two Input Network

| Input Patterns | Inputs | | Possible Output Mappings | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | I1 | I2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 2 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 3 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

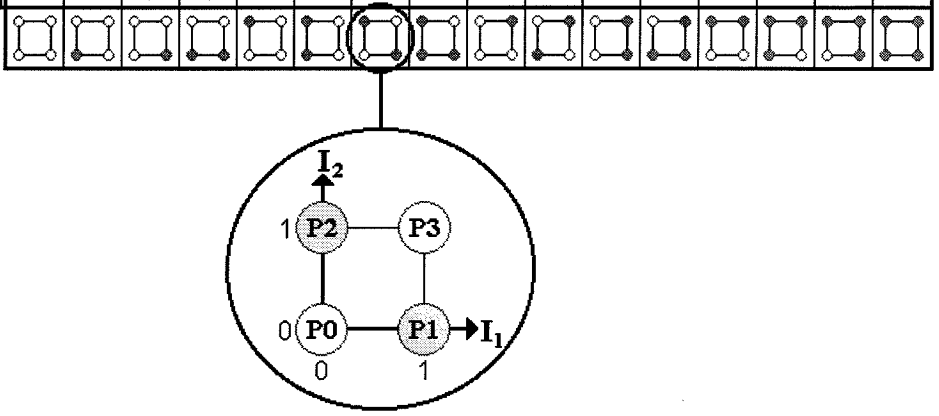


Figure 9.8: **Possible Target Patterns for Two Input Network.** This figure shows the 16 possible mappings that a network with only 2 inputs can learn. There are four possible input patterns: $(0,0)$, $(1,0)$, $(0,1)$, and $(1,1)$, which can be labeled $P0$, $P1$, $P2$, and $P3$. (The number associated with each input pattern corresponds to the decimal equivalent of the binary representation of $(I1, I2)$, with $I1$ being the least significant bit.) Each input pattern can cause the network to produce either a 0 or 1 output. Thus, there are 16 possible output mappings, numbered from 0 to 15. (The number corresponds to the decimal equivalent of the binary representation of $(P0, P1, P2, P3)$, with $P0$ being the least significant bit.) The last line of the table shows a graphical representation of each of the output mappings, with the filled circles corresponding to an output of 1 and the empty circles corresponding to an output of 0. The graphical representation for mapping #6, is enlarged to show the axes. (Mapping #6 is equivalent to the XOR function in Boolean logic.)

problem of becoming trapped in a local minimum.[6]

## 9.5.3 The Learning Rule

When a network with a single output neuron is to learn one of the $2^{2^N}$ mapping patterns, the different input patterns can be applied in random order, and the output of the network can be compared with the desired output. The learning rule developed for this type of network is:

1. If the desired output and actual output are the same (both 0 or both 1), then do not alter any of the connections into the output neuron or any of the neurons that connect to it.

2. If an input or hidden neuron is not firing, then the output connections it makes to other neurons can not change regardless of the actual or desired output state; i.e., if the presynaptic neuron is not firing, the synapse remains unchanged.

3. If a hidden neuron does not have any connections to the output neuron, then it may randomly change its own input connections. The connections from any firing input neurons into this hidden neuron may increase or decrease with equal probability. Notice that if the neuron does not have any connections to the output neuron, then it must be firing at times uncorrelated to the output function being learned. The formation of random connections allows it to change its spiking behavior with the possibility of becoming correlated with the desired output function.

4. If the desired output is 1 and the actual output is 0, then:

[6]It is possible that layered biological networks are able to vary the number of "hidden" neurons. When an output neuron is unable to correctly learn a task, additional neurons can be "recruited" to increase the degrees of freedom available for learning. For the network presented here, this scenario would be analogous to having multiple output neurons and hidden units, with the different layers not being fully connected; i.e., most hidden neurons only connect to a fraction of the output neurons and receive inputs from a fraction of the inputs. When one of the output neurons is unable to learn its desired task, additional connections are made from hidden neurons, which previously connected only to other output neurons.

A) Increase the connections from the excitatory neurons (inputs), which are firing, to the output neuron.

B) Decrease the connections from the inhibitory (hidden) neurons, which are firing, to the output neuron.

C) Decrease the weights from the excitatory neurons (inputs), which are firing, to the hidden neurons which are also firing and connected to the output neuron. (If an inhibitory neuron is not connected to the output neuron, randomly adjust its input weights according to Rule 3.)

D) Do not alter the weights from the excitatory neurons (inputs) to any hidden neurons which are not firing.

5. If the desired output is 0 and the actual output is 1, then:

A) Decrease the connections from the excitatory neurons (inputs), which are firing, to the output neuron.

B) Increase the connections from the inhibitory (hidden) neurons, which are firing, to the output neuron.

C) Increase the weights from the excitatory neurons (inputs), which are firing, to the hidden neurons which are not firing but connected to the output neuron. (If an inhibitory neuron is not connected to the output neuron, randomly adjust its input weights according to Rule 3.)

D) Do not alter the weights from the excitatory neurons (inputs) to any hidden neurons which are already firing.

Now, the above learning rules indicate which type of changes should occur at a synapse; however, all actual weight changes are assumed to be random occurrences. This can be mathematically summarized as:

$$\vec{W}^{\text{new}} = \vec{W}^{\text{old}} + \Delta\vec{W} \tag{9.21}$$

where:

$$\Delta W_{ij}^2 = [u(\epsilon_{ij} - \chi)] \cdot (D_i - O_i) \cdot I_j \tag{9.22}$$

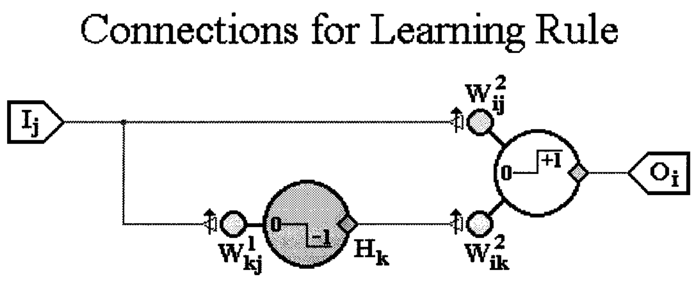
$$\Delta W_{ik}^2 = [u(\epsilon_{ik} - \chi)] \cdot (D_i - O_i) \cdot H_k^- \tag{9.23}$$

$$\Delta W_{kj}^1 = \begin{cases} [u(\epsilon_{kj} - \chi)] \cdot (D_i - O_i)^2 \cdot \left(1 + H_k^- - D_i\right) \cdot I_j & \text{if } W_{ik}^2 \neq 0 \\ [u(\epsilon_{kj1} - \chi)] \cdot \operatorname{sgn}(\epsilon_{kj2} - 0.5) \cdot I_j & \text{if } W_{ik}^2 = 0 \end{cases} \tag{9.24}$$

Here, $u(-)$ is the unit step function defined in Equation 4.4, and $\epsilon$ is a uniformly distributed random number between 0 and 1. The subscripts on $\epsilon$ indicate that a different random number is chosen for each weight on each learning iteration and the subscripts of "1" and "2" on $\epsilon_{kj}$, for the case when $W_{ik}^2 = 0$, distinguish the two different random numbers chosen for each of those weights, with the first determining if the weight will be changed and the second determining if the weight will be increased or decreased. (The sgn$(-)$ function is as defined in Figure 1.2.) The value of $\chi$ is usually a constant for the entire network, and it sets the probability of each weight change, with values close to 1 producing few if any changes during each learning iteration and values close to 0 producing many changes. Notice that the $\Delta W$'s can only assume the values of either +1, 0, or -1.

This learning rule is best understood by referring to Figure 9.9, which shows which type of weight changes are possible for the different combinations of input, hidden, output, and desired values. Notice that whenever a neuron's output is zero, none of its output connections may change.

When $O_i - D_i = 1$, the output neuron is firing when it should not be. Consequently, the connections from the firing inputs to the output neuron tend to decrease while the connections from the firing inhibitory hidden neurons to the output neuron tend to increase. Also the input connections from the firing inputs to the non-firing inhibitory hidden neurons are increased, making these inhibitory neurons more likely to fire. The input connections into the inhibitory neurons which are already firing are unaltered, provided that these non-firing inhibitory neurons are connected to the output neuron.



**Possible Weight Changes**

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Neuron Outputs** | $I_j$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| | $H_k$ | 0 | 0 | −1 | −1 | 0 | 0 | −1 | −1 | 0 | 0 | −1 | −1 | 0 | 0 | −1 | −1 |
| | $O_i$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| | $D_i$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **Weight Changes** | $\Delta W^2_{ij}$ | 0 | 0 | 0 | 0 | 0 | −1 | 0 | −1 | 0 | +1 | 0 | +1 | 0 | 0 | 0 | 0 |
| | $\Delta W^2_{ik}$ | 0 | 0 | 0 | 0 | 0 | 0 | +1 | +1 | 0 | 0 | −1 | −1 | 0 | 0 | 0 | 0 |
| | $\Delta W^1_{kj}$ | 0 | 0 | 0 | 0 | 0 | +1 | 0 | 0 | 0 | 0 | 0 | −1 | 0 | 0 | 0 | 0 |

Figure 9.9: **The Learning Rule.** This figure shows the different network connections and how they can change during the learning process. Only one input and one hidden neuron are shown although there are more in the complete network. While the table indicates the possible weight changes, the actual changes are probabilistic. Thus, those changes with a +1 value may increase or stay the same; those changes with a -1 value may decrease or stay the same; and those change with a 0 value can not change. Of course, none of the weights is allowed to become negative. Also, the table assumes that $W^2_{ik} \neq 0$. If $W^2_{ik} = 0$, then the weight from the input to the hidden neuron, $W^1_{kj}$, can randomly increase or decrease whenever $I_j = 1$.

Similarly, when $O_i - D_i = -1$, the output neuron is not firing when it should be. Consequently, the connections from the firing inputs to the output neuron tend to increase while the connections from the firing inhibitory hidden neurons to the output neuron tend to decrease. Also, the input connections from the firing inputs to the firing inhibitory hidden neurons are decreased, making these inhibitory neurons less likely to fire. The input connections into the inhibitory neurons which are not firing are unaltered, provided that these non-firing inhibitory neurons are connected to the output neuron.

Finally, when $O_i = D_i$ only the weights from the inputs to the hidden neurons, which are not connected to the output neuron, may change. (Figure 9.9 assumes that the hidden neuron, $H_k^-$, is connected to the output neuron.) These neurons may be described as independent units since they have no influence on the network's output.

When implementing this learning rule, it is important to observe that very little global information is required. To adjust the connections into the output neuron, it is only necessary to know the actual output, the desired output, and the outputs of the neurons forming the connections. Similarly, to adjust the connections from the inputs into the hidden neurons, it is only necessary to know the output of the hidden neuron, the outputs of the input neurons forming the connections, the desired network output, and whether or not the network is producing the desired output. Notice that because the output signal is only used within the $(D_i - O_i)^2$ term, it is unnecessary to know the actual output of the network. Instead, it is only necessary to know if the output is correct. Theoretically, such a rule could be implemented within a biological system, with a special neuron releasing a chemical if the output is incorrect, and another special neuron releasing a chemical when the desired output is 1 (or 0). These chemicals could diffuse through the network, with the first increasing the probabilities for synaptic changes and the second indicating which synapses should increase or decrease. Since none of the weight changes depends upon any weight values, this learning rule does not require any bidirectional connections as are required with backpropagation learning.

When a network has multiple outputs, the connections into the output neurons are independent of each other; however, these output neurons may share hidden neurons; i.e., some hidden neurons may connect to more than one output neuron. This causes a problem in determining how to adjust the input weights into these hidden neurons, since the adjustments depend upon the desired outputs. For multiple output networks, no adjustments should be made in any of the weights from the inputs to a hidden neuron unless all of the output neurons to which the hidden neuron connects indicate that a change should be made and agree on the direction of the change. Also, when a hidden neuron fires, the weights associated with the connections it forms on the output neurons will tend to either increase, decrease, or stay the same. But if any of these output neurons indicate a decrease in the weight, no weight increases to the other neurons are allowed. Since weight increases are only allowed when the output neurons are in agreement, while all weight decreases are permitted, eventually the number of output neurons that the hidden neuron connects to will be reduced until there is either only one output neuron or all of the remaining output neurons are in agreement as to the required weight changes. No weight changes are made from the inputs to a hidden neuron until the hidden neuron only connects to a set of output neurons with correlated requirements.

### 9.5.4 Comparison with Backpropagation

This learning rule is very similar to the perceptron and backpropagation methods previously discussed in Sections 9.3 and 9.4. For the connections into the output neuron, Equations (9.22) and (9.23) are essentially identical to the Perceptron learning rule in Equation (9.3) except that the weights can only assume positive integer values and all weight changes are probabilistic rather than deterministic. (Notice that the output from a hidden neuron is always 0 or -1 rather than 0 or +1. Thus, for the same $O_i$ and $D_i$ values, the connections from the inputs to the output and the connections from the hidden neurons to the output move in opposite directions.) However, the primary difference between the perceptron learn rule and the learning rule presented in this section is the ability to adjust the weights between the inputs and hidden neurons. It is this hidden layer of neurons that allows the network to solve problems that are linearly inseparable.

The motivation for the weight changes in Equations (9.22)-(9.24) can be understood by applying backpropagation to the network in Figure 9.9. Since the backpropagation method only works for neurons with a differentiable transfer function, the outputs from the neurons will be determined by:

$$H_k^- = \frac{-1}{1+\mathrm{e}^{-\gamma h_k^1}} \tag{9.25}$$

$$O_i = \frac{1}{1+\mathrm{e}^{-\gamma h_i^2}} \tag{9.26}$$

where:

$$h_k^1 = \sum_{j=1} W_{kj}^1 I_j \tag{9.27}$$

$$h_i^2 = \sum_{j=1} W_{ij}^2 I_j + \sum_{k=1} W_{ik}^2 H_k^- \tag{9.28}$$

While these neurons have continuous transfer functions, they have the same limits as the McCulloch-Pitts threshold neurons; i.e., the output of the network, $O_i$, is bounded between 0 and 1, and the outputs from the hidden neurons, $H_k^-$, are bounded between 0 and -1. Also notice that as $\gamma \to \infty$, these transfer functions reduce to threshold neurons.

Now, by calculating the gradient of the quadratic error function in Equation (9.5), the appropriate weight changes are found to be:

$$\Delta W_{ij}^2 = -\eta\frac{\partial E}{\partial W_{ij}^2} = \eta\left[\gamma O_i\left(1-O_i\right)\right]\cdot\left(D_i - O_i\right)\cdot I_j \tag{9.29}$$

$$\Delta W_{ik}^2 = -\eta\frac{\partial E}{\partial W_{ik}^2} = \eta\left[\gamma O_i\left(1-O_i\right)\right]\cdot\left(D_i - O_i\right)\cdot H_k^- \tag{9.30}$$

$$\Delta W_{kj}^1 = -\eta\frac{\partial E}{\partial W_{ik}^1} = \eta\left[\gamma O_i\left(1-O_i\right)\right]\cdot\left[\gamma H_k^-\left(1+H_k^-\right)\right]\cdot W_{ik}^2\cdot\left(D_i - O_i\right)\cdot I_j \tag{9.31}$$

If the output of any neuron is exactly 0 or $\pm1$, the coefficient terms of $\left[\gamma O_i\left(1-O_i\right)\right]$ or $\left[\gamma H_k^-\left(1+H_k^-\right)\right]$ are zero; however, with the neural transfer functions in used

Equations (9.25)-(9.28), this can never be the case for finite weights and inputs. In fact, $[\gamma O_i (1 - O_i)]$ must always be greater than zero, and $\left[\gamma H_k^- \left(1 + H_k^-\right)\right]$ must always be less than zero. Thus:

$$\Delta W_{ij}^2 = \eta_i (D_i - O_i) \cdot I_j \tag{9.32}$$

$$\Delta W_{ik}^2 = \eta_i (D_i - O_i) \cdot H_k^- \tag{9.33}$$

$$\Delta W_{kj}^1 = -\eta_{ik} (D_i - O_i) \cdot I_j \tag{9.34}$$

Here, the $\eta$'s represent the proportionality constants which are all positive. Thus, it is obvious that the weight changes for $W_{ij}^2$ and $W_{ik}^2$ in the learning rule of Equations (9.22)-(9.23) are in the same direction as the negative gradient of $E$.

To understand the relationship between Equation (9.34) and (9.24), it is necessary to consider the direction of the weight changes. When $D_i$ is greater than $O_i$, the $W_{kj}^1$ weight is always decreased by an amount proportional to $I_j$, and when $D_i$ is less than $O_i$, the $W_{kj}^1$ is always increased by an amount proportional to $I_j$. Therefore, when $I_j$ is very small, the actual weight changes are approximately zero. But these are the same directions for $W_{kj}^1$ as in Figure 9.9, with the exception that the learning rule of Equation (9.24) never increases $W_{kj}^1$ when $H_k^-$ is already firing and never decreases $W_{kj}^1$ when $H_k^-$ is not firing. For the McCulloch-Pitts threshold neuron, these weight changes would have no effect on the hidden neuron's output.

While the discrete nature of the learning rule's weight changes prevents it from following the exact gradient of the error function, *all of the weight changes have the same sign as the negative of the error gradient.* Thus, each of the parameter adjustments can be expected to reduce the error for the current pattern being presented, in much the same way as backpropagation learning. However, the weight adjustments are made without propagating the error signal backwards through bidirectional connections.

**Possible Mappings**

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Min.** | 2 | 15 | 2 | 9 | 2 | 4 | 2 | 5 | 2 | 60 | 2 | 6 | 2 | 6 | 2 | 2 |
| **Max.** | 30 | 214 | 139 | 225 | 191 | 189 | 532 | 242 | 221 | 1306 | 93 | 204 | 96 | 219 | 111 | 79 |
| **Mean** | 7 | 62 | 32 | 60 | 37 | 54 | 95 | 45 | 56 | 364 | 20 | 65 | 22 | 69 | 13 | 17 |
| **Med.** | 6 | 48 | 20 | 54 | 18 | 50 | 57 | 29 | 44 | 321 | 12 | 55 | 16 | 56 | 6 | 9 |

Table 9.1: **Results for Two Input Network with Two Hidden Inhibitory Neurons.** This table shows the minimum, maximum, mean (rounded to nearest integer), and median number of learning iterations required to correctly learn each of the 16 different mappings described in Figure 9.8 over 50 trials. Overall, for the 800 (= 16 $*$ 50) trials, the mean number of iterations was 63.6 and the median number of iterations was 30.

### 9.5.5 Test Results

To test this learning rule, the network was started from a random initial configuration, with some connection strengths set at 1 and the remaining connections set at 0. None of the connections had an initial strength greater than 1, and the randomization procedure was such that each of the connections had a 75% probability of being 1 and a 25% probability of being 0. (The value of each connection was chosen independent of the others.) The threshold value for all of the neurons in the network was set equal to $N - 0.5$, where $N$ is the number of network inputs (not counting the bias), and $\chi$ was set at 0.8. Neither of these varied during the learning process. When this learning rule was tested on a network with two inputs and two hidden inhibitory neurons, the results were as shown in Table 9.1.

The network was also tested on a network with three inputs and three hidden inhibitory neurons. As with the two input network, the learning rule eventually converged to a solution for all 50 trials of each of the 256 $\left(= 2^{2^3}\right)$ mapping patterns on which it was trained; however, some mappings were found to be much more difficult than others. Overall, for the 12800 (= 256 $*$ 50) trials, the mean number of learning iterations was 898.81 and the median was 240. However, the network had the most difficulty learning the mapping for pattern #105, which is depicted in Figure 9.10. For the 50 trials with this mapping pattern, the number of learning iterations ranged

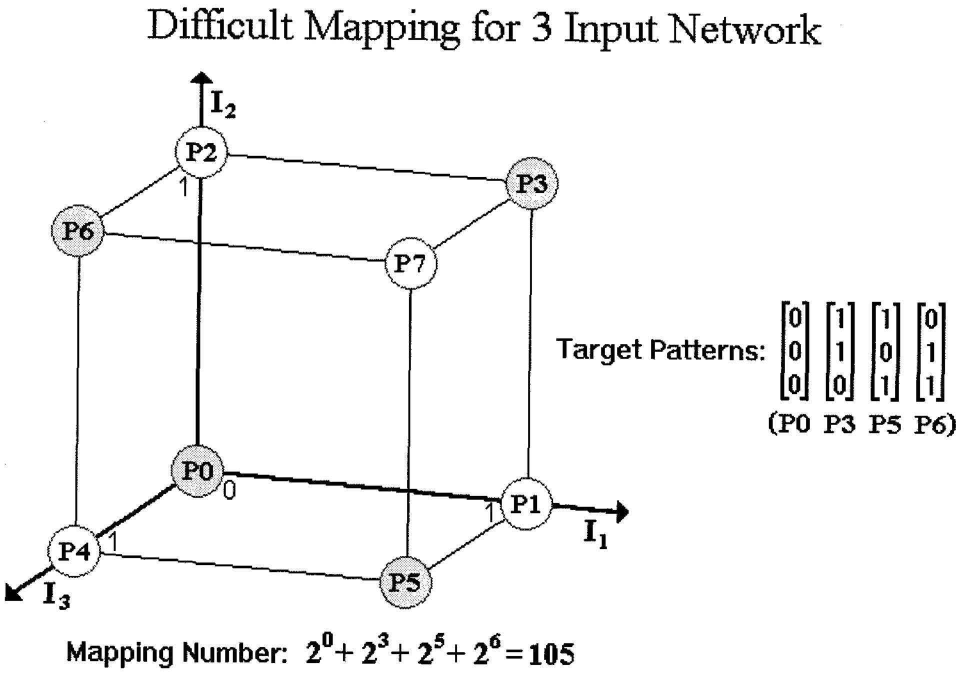

Difficult Mapping for 3 Input Network
I2
P2
1
P3
P6
P7
Target Patterns:
(P0 P3 P5 P6)
P0
0
P1
1
I1
P4
1
P5
I3
Mapping Number: 2^0+ 2^3+ 2^5+ 2^6=105

Figure 9.10: **Difficult Pattern Mapping.** This figure illustrates the mapping pattern that caused the most difficulty for the three input, one output network to learn. The network was only to output a 1 when presented with the four input patterns of: $[0;0;0]$, $P0$; $[1;1;0]$, $P3$; $[1;0;1]$, $P5$; and $[0;1;1]$, $P6$. Notice that with three inputs, there are eight possible input mappings, which are represented as vertices on a cube. There is a total of 256 different combinations (mappings) of the eight possible input patterns, which can be numbered from 0 to 255. The problem mapping shown here is #105 $(= 2^0 + 2^3 + 2^5 + 2^6)$.

from 735 to 188,808, with a mean of 39,498 and a median of 27,385. For comparison, the next most difficult pattern was #73, where the number of learning iterations ranged from 729 to 63,167, with a mean of 11,364 and a median of 7,984. Only seven of the 256 mappings had a mean number of iterations over 5000.

In an attempt to improve the overall learning rate, the network architecture was modified. Three additional *excitatory* hidden units were included (see Figure 9.11). These hidden neurons received connections from the inputs, and acted like extra input neurons with respect to the other neurons in the network; i.e., like the original inputs, their excitatory output signals connected to both the inhibitory hidden neurons and the output neuron. All of the inputs connected to each of these new excitatory hidden neurons through one synapse; i.e., their input synaptic weights were all fixed at 1.

While these input connections did not vary, the neurons' output connections changed during learning. Like the other neurons in the network, their threshold value was set at 2.5 ($= N - 0.5$), but each had a different number of input connections from the bias signal, which had the effect of producing different threshold values. Thus, one of the neurons would fire only if all three inputs were firing (no input from bias); another neuron would fire if two or more inputs were firing (one input connection from bias); and the third neuron would fire if one or more inputs were firing (two input connections from bias). Therefore, these "additional" hidden neurons provided signals that explicitly responded to symmetries in the input patterns that the network was spending a great deal of time learning. With this modified architecture, the network converged to solutions much faster for the difficult mappings. When this network was tested with the 256 mappings, the overall mean number of learning iterations was reduced to 316.4 and the median was reduced to 216, with the maximum number of iterations for any mapping in any of 50 trials being reduced from 188,808 to only 4,208.

Notice that when this three input network is configured with the appropriate synaptic time delays and time constants, it can be used to learn to solve the example temporal and spatial recognition tasks discussed in Section 7.3. Furthermore, it is possible to also have the time delays be adjustable parameters. In this case, each of the input connections into the output neuron starts out with a different time delay, and when the neurotransmitter contribution from an input spike does not coincide with the pseudospikes produced from the pace neuron, the time delays are adjusted as necessary for synchronization, independent of the actual or desired output of the neuron.

Finally, a network with five inputs and five inhibitory hidden units (original network architecture with no excitatory hidden units) was trained to learn the Boolean logic function described in Equation 7.18 of Section 7.4. Since this problem has only one unique solution, the network was able to learn the correct mapping relatively easily, with the minimum, maximum, mean, and median number of learning iterations over 50 trials being 63, 4731, 845.2, and 478, respectively.

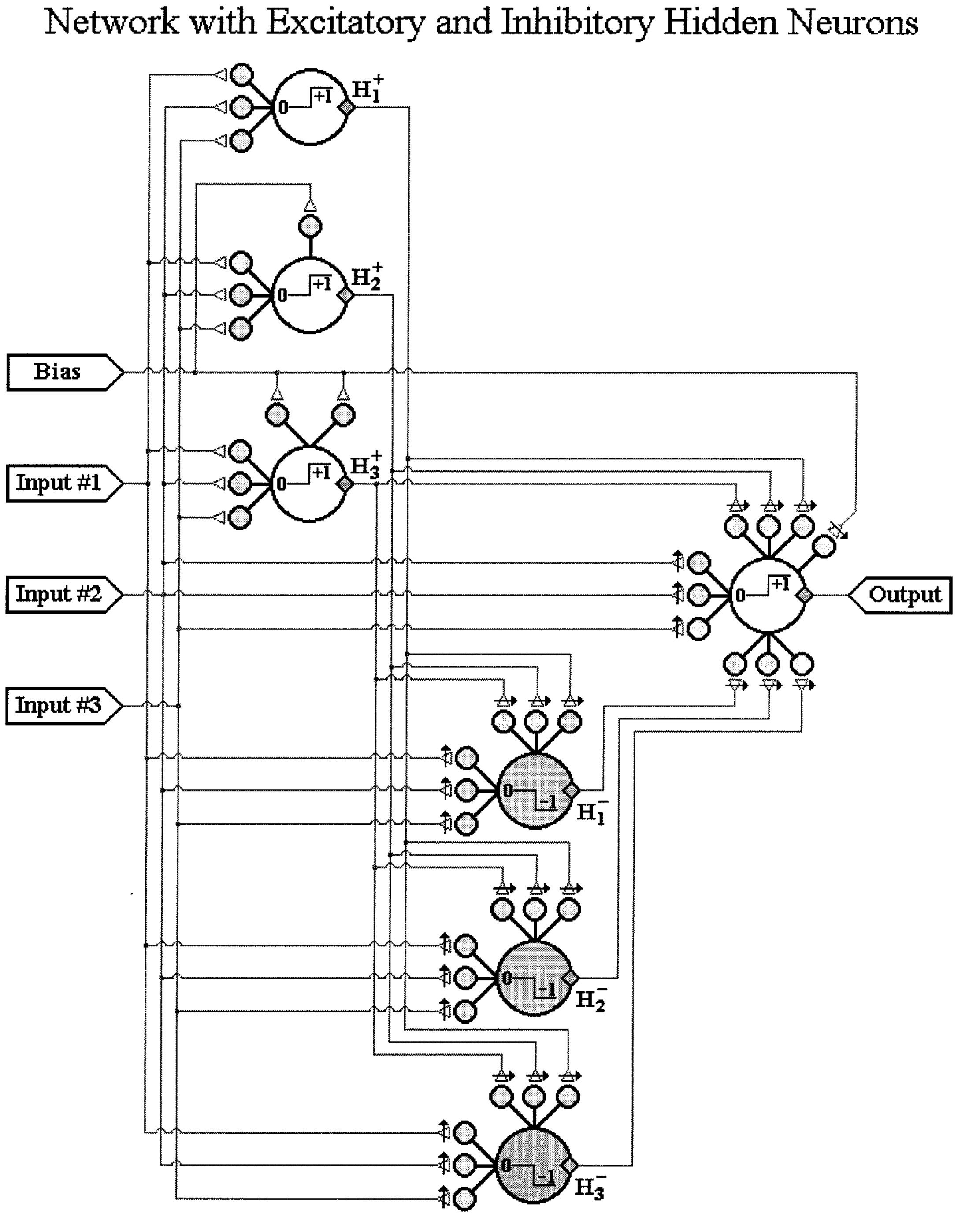


Figure 9.11: **Network with Excitatory and Inhibitory Neurons.** This figure shows a network with with an extra layer of hidden excitatory neurons, which acts like a set of additional inputs. All of their input connections are fixed at 1 and do not vary during learning, but each neuron has a different number of connections from the bias signal. With this extra layer of excitatory neurons, the network was able to significantly reduce the number of learning iterations for difficult pattern mappings.

## 9.6 Summary of Learning Methods

This chapter demonstrated how learning can be implemented within a network of spiking neurons. The chapter started by discussing the importance of genetics in governing the development of the neural connections and determining the type of functions a network can perform. Consequently, a network designer should not consider a fully-recurrent neural network a general solution to all computational problems. Only through millions of years of evolution has biology evolved the intricate network configurations required to perform specific tasks. Thus, a great deal of forethought should go into choosing an appropriate network configuration. Currently, most learning algorithms are only able to make minor adjustments to the neural connections, and not make the major changes that take place within developing nervous systems.

Next, Section 9.2 presented a brief overview of learning, which included a discussion of Hebbian learning and how it is believed to occur within real neurons. Section 9.3 described the McCulloch-Pitts threshold neuron and its use in perceptron networks. The perceptron learning rule is presented, which can be used within a single layer network to learn any linearly separable function. Section 9.4 described the backpropagation learning algorithm for multilayered feedforward networks of neurons with differentiable transfer functions. Unlike the perceptron network, multilayered networks can solve problems that are linearly inseparable.

Finally, Section 9.5 presented a novel network architecture and learning rule. Like the perceptron, the network is based on the McCulloch-Pitts threshold neuron, but it can easily be implemented within a network of spiking neurons. While it uses two layers, all of the inputs connect to *both* the hidden neurons and output neurons and can only form excitatory connections. The hidden neurons connect to the output neurons and can only form inhibitory connections. While its learning rule is similar to that of the perceptron, since it is a two layer network constructed from neurons with nonlinear transfer functions, it can solve problems that are linearly inseparable. By restricting all of the weights within the network to positive values (when the output of the hidden neurons is defined as -1 above threshold), the weight changes determined by the learning rule have the same sign as they would if the neurons' outputs were sigmoidal and backpropagation learning were used. However, unlike backpropagation, the learning rule does not require the error signal to be propagated backwards through biologically implausible bidirectional connections.

# Chapter 10 Conclusions

## 10.1 Purpose of Thesis

Most research in artificial neural networks has the primary goal of building better computing devices. Conversely, neuroscientists investigate biological neurons and use simulations to understand how neural systems function. Obviously, the findings from both approaches are mutually beneficial. The research presented in this thesis represents an attempt to further narrow this gap by using some known information about neurobiology to construct better computational devices composed of spiking neuron-like models. In some sense, this thesis fits neither into the category of engineering nor neurobiology, but instead attempts to carve out some middle ground, in much the same way that Hebb's [37] theoretically work on learning has had a profound affect in both computation and neurobiology. The research was based on the assumption that using spikes in artificial neural networks must be advantageous since biology uses spikes. (Spikes also have a number of known advantages in information transmission and hardware implementation, which are discussed further in this chapter.) The thesis then preceded to demonstrate how some computations could be implemented within networks of spiking neuron models. While there has been some previous research into the computational abilities of individual spiking neurons (e.g., [13, 107]), how a spiking neuron may encode information (e.g., [47, 67]), and the theoretic capabilities of large networks (e.g., [66, 68]), there has been little research into how specific computations would be performed within such a network. Obviously, many of the computations demonstrated in this thesis have engineering applications, but the methods in which the computations were performed also have biological implications which are addressed in Section 10.4.2. It is hoped that some of the computational ideas presented in this thesis may offer reasonable paradigms for biological systems, which stimulate further investigation by neurobiologists.

### 10.1.1 Why Artificial Neural Networks?

Artificial neural networks are an attempt to duplicate the basic machinery of biological intelligence. Biology has a computational advantage over conventional computers with its parallel structure, which allows vast numbers of neurons to process information simultaneously. Information is encoded in a distributed fashion, allowing neural networks to use noisy and incomplete data and still make "good" decisions quickly. While conventional computers are far superior to humans in serial tasks, such as arithmetic, the human brain is superior in tasks that require parallel processing, such as visual pattern recognition. For a computer to recognize an object in a scene, it must analyze the picture one pixel at a time, while the human visual system is able to process the entire scene simultaneously. Computers follow the commands in programs, whereas neural networks do not make precise calculations, and it even is difficult to reduce their processing into an algorithm. If there is an error in a computer program, the computer will produce the same incorrect output each time, but a neural network may be able to learn from its past experiences and produce new results. The process of self-modification gives neural networks their ability to adapt to new environments, and it has been one of the motivations for research into building more biologically based computational devices.

Due to the vast number of different types of computing devices referred to as "artificial neural network" (most of which are actually simulated on serial computers), it is best to describe the field of artificial neural networks as a paradigm shift in computing. The research presented in this thesis continued in that spirit, by presenting a new neuron model and developing networks based on it which are capable of computing functions in a distributed fashion. (All networks were simulated on a serial computer.) These networks were designed to deal with information that is noisy or imprecise and provide reasonable outputs, and they were not meant to be precise computational devices.

### 10.1.2 Why Spikes?

This thesis has explored new methods for computing with spiking neurons, but much of the previous research in artificial neural networks has focused on the use of neurons with continuous analog signals. While it has been well established that biological neurons display spiking behavior, the justification for using continuous analog signals is based on the tacit assumption that the output signal of a neuron primarily depends upon its mean spiking frequency, which can be represented as an analog signal (see [46] for example). However, it is particularly dubious whether networks of analog neurons can provide a useful paradigm for the fast computations demonstrated in cortical neural systems [67]. There is some experimental evidence that at least some information must be transmitted in individual spikes. It has been shown in macaque monkeys that the cortical area involved in visual processing can complete its computation in just 20 to 30 ms [92, 93], while the firing rates of theses neurons is usually below 100 Hz. Hence, an encoding scheme based on the average firing rates would require approximately 20 to 30 ms just to sample the frequency of these neurons [67].

Obviously, a coding scheme based on single spikes or interspike intervals yields a much greater information capacity than one using an average spike rate [21]. The information capacity of a signal increases linearly with bandwidth but only logarithmically with the signal-to-noise ratio [103]. Thus signals with a low signal-to-noise, wide bandwidth are much more effective at transmitting information. Like real neurons, the spikes produced by the SNM are only used for transmitting information. The actual computations performed within a neuron are analog in nature; i.e., the neurotransmitter released due to a spike and its effect on the postsynaptic neuron's potential is continuous. Thus, from an engineering standpoint, the SNM uses a mixed approach: analog signals, which have a narrow bandwidth but high signal-to-noise ratio, are used for processing, while spikes, which have a high bandwidth but low signal-to-noise ratio, are used for transmitting information between neurons. Obviously, when simulating the SNM on a computer, factors such as noise and transmission capacity are not significant; however, the SNM was specifically designed with the intention of eventually constructing large networks in VLSI hardware, where using pulses offers the possibility of combining extremely high computation speeds (one parallel computation step per pulse) with the noise-robust transmission of analog variables via the relative timing of spikes from different neurons [67]. (See Section 10.5.)

There are a number of problems in engineering where the ability to compute with spikes (extremely brief pulses) is advantageous with respect to conserving power and area on a VLSI chip. In general, most transducers convert a measured physical quantity into an analog signal. This analog signal is usually encoded digitally through a D/A converter and transmitted to a microprocessor. The microprocessor analyzes the input signal and computes a digital output for controlling a set of motors. This digital signal is usually converted back into an analog signal through an A/D converter and then transformed into series of pulses, which are amplified and used to drive the motors. However, it is possible to design relatively small, simple circuits that convert an analog signal into a series of spikes, where power is dissipated only during the pulse, and not during in the quiescent state [8]. Thus, a system of computation based on spikes eliminates the need to convert physical quantities into digital numbers for processing. This can reduce the overall size and power requirements of the system. (For examples of applications where spikes have been used to process information in VLSI hardware, see [8, 49, 97].)

## 10.2 Summary of Thesis

### 10.2.1 Spiking Neuron Model

A new spiking neuron model (SNM) was developed. It was designed to capture much of the interesting dynamics demonstrated by real neurons, while remaining simple enough to simulate and understand. The state of a neuron was completely determined by the amount of neurotransmitter at its input synapses and the time since it last produced a spike. The neuron's output spikes were treated as discrete events that had no width associated with them. Only the inter-spike time intervals mattered, as the spikes reset the neuron's potential and triggered the release of neurotransmitter at the interneuron connections.

A network was composed of neurons, which could receive inputs from external sources or spikes from other neurons. The arrival time of a spike was determined by the time the spike was produced, plus the delay associated with the connection (synapse). When a spike arrived at a synapse, neurotransmitter was released with the functional form of $te^{-t}$ (see Equation (3.2)). Because the neurotransmitter was modeled with a slowly decaying function, a burst of closely spaced input spikes caused a buildup of neurotransmitter at a synapse. This neurotransmitter then affected the voltage potential of the postsynaptic neuron (Equation (3.1)), and depending upon the sign of the connection, made it more (excitatory) or less (inhibitory) likely to produce spikes. The neurotransmitter's effect on the neuron's potential was modeled by the hyperbolic tangent of the total neurotransmitter concentration at each synapse (see Equation (3.9)). Modifications to the basic SNM were also presented, which mimicked the biological functions of presynaptic inhibition (Equation (3.13)), and synaptic clusters (Equations (3.14)-(3.15)).

One of the key properties that gave the SNM its unique characteristics was that the state of a neuron did not just depend upon the current input signals, but depended upon all previous input spikes. Each input spike triggered the release of neurotransmitter at a synapse. Successive input spikes led to a build up of neurotransmitter at the synapses, causing the neuron's potential to change by more than it could from individual spikes. Although the total neurotransmitter at a synapse depended upon the entire spike history of the presynaptic neurons, Section 4.2.2 showed how the neurotransmitter could be expressed in an iterative formula (Equation (4.14)), so that it was not necessary to recalculate the sum over all previous spikes every time a new spike arrived. And when the input spikes occurred with a repeating pattern, Section 4.2.3 showed how the iterative representation for the neurotransmitter could be further simplified (Equations (4.21)-(4.24)).

While neurotransmitter was used by neurons to exchange information, it was the neuron's potential that determined the production of spikes. Like real neurons, the SNM could exhibit latency (Section 4.3.2, Appendix E), refractory periods (Section 4.3.3), endogenous spiking behavior (Section 4.3.4), and adaptation (Section 4.3.5). With latency, strong input stimuli resulted in the rapid production of output spikes, while weak stimuli required more time and did not always lead to output spikes. After the SNM fired, there was an absolute refractory period during which it was unable to fire again, and as the SNM started to recover, there was a relative refractory period during which it was only able to produce output spikes when the input stimulus was stronger than normal. When the neuron parameters were set accordingly, the SNM could produce a beating or bursting output spike train in the absence of external stimulation. Of course, input signals could modify or inhibit the neuron's endogenous output spike pattern. With an inhibitory feedback connection, the SNM could adapt its output spike frequency to constant input signals. This allowed it to be more sensitive to changes within the input signal rather than the input's strength.

Like other computational devices, the SNM could be analyzed in terms of its transfer function. Since the neuron received input spikes and produced output spikes, the transfer function was expressed as the relationship between the neuron's input and output spike frequencies (Section 4.4). The SNM had both a maximum and minimum output frequency. Excitatory input spikes caused the output spike frequency to increase, but the absolute refractory period limited the maximum possible output spike frequency.

The net effect of having a refractory period along with the strength-latency relationship was that the SNM could encode information within its output spike frequency. As with real neurons, a stronger input stimulus produced a faster spike response and higher output frequency. (Chapter 8 discussed how spikes could be used to encode information within neural networks.) Overall, the SNM fulfills its purpose of providing a relatively simple spiking neuron model which can be used for spike based computations.

### 10.2.2 Special Function Neurons

Chapter 6 demonstrated a few examples of some simple functions that individual neurons can perform. These special function neurons were used as building blocks for the larger networks presented in Chapter 7, which were designed to accomplish more complex tasks. Each section in that chapter began by discussing the desired function the neuron was to accomplish, and then developing constraints on the neuron's parameters for ensuring that the function was performed. In general, the constraint equations were sufficient, but not necessary for implementing the desired function, and looser constraints could have been imposed, but usually at a cost of complicating the analysis. Each section also contained a sample output from a neuron performing the desired function, and gave the parameter values used. Since the constraint equations were continuous, the solution provided resides within a connected region of the parameter space, and the parameter values can be modified as desired to alter the neuron's output while remaining within the constraint boundaries.

Section 6.2 presented a high gain neuron, which output spikes at a nearly constant frequency when the input signal exceeded the neuron's threshold level. This neuron was used in the stimulus detector of Section 7.2 for detecting the presence of an input stimulus. The high gain neuron was designed by choosing the desired output frequency and the stimulus strength required to exceed threshold and produce spikes of that frequency. After deciding upon these specifications, the parameters for this neuron were constrained by Equations (6.1)-(6.3).

Section 6.3 presented two different types of memory oscillators, which were analogous to digital flip-flops and could be used for storing the state of an input signal. They varied in the configuration of the input signals used to control their behavior, and represented only a sampling of the numerous ways that the SNM can be connected to perform "memory-like functions." Section 6.3.2 discussed the Dual Control Memory Neuron, which used two input signals; one was excitatory and started oscillations, while the other was inhibitory and stopped oscillations. Equations (6.8)-(6.14) specified the sufficient parameter constraints for a neuron in the duel control configuration to cause it to switch between the oscillatory and silent states when input control spikes arrived. Section 6.3.3 discussed the Single Control Memory Neuron, which used only one input signal, and every time the neuron received a spike, it switched from oscillating to silent, or vice versa. Equations (6.16)-(6.20) specify the parameter constraints for a neuron in the single control configuration. Large arrays of memory oscillators can be used as binary memory units, with each neuron retaining one bit of information. Section 7.6 showed how memory oscillators can be connected to create counting networks.

In the SNM as well as in real neurons, a spiking output signal is produced when the input signal exceeds the neuron's threshold level; however, there are some situations where it is desirable to have the neuron only respond when the input signal is within a limited range. The bounded threshold neurons of Section 6.4 failed to produce any spikes for large or small input signals outside of their specified range. After choosing the range of input strengths for producing an output spike, the neuron's parameters were constrained by Equations (6.26)-(6.29). These neurons could be used to measure the strength of an input stimulus or signal the start and end of an input stimulus. Also, when a time delay was used on one of the input synapses, the neuron had a *gradient threshold*, and could be used to detect changes in the input signal.

For some computing problems, it is useful to have a neuron with an identity response, in which the output spikes are correlated with the input spikes. Section 6.5.2 described an unsynchronized identity neuron, which outputs a spike for each incoming spike. The neuron was designed by first choosing a maximum input frequency for which it is to be capable of producing a one-to-one correlation between the input and output spikes. After specifying this frequency, the parameters of this neuron are constrained by Equations (6.35)-(6.40). As long as output frequency did not exceed the limit for which it was designed, this neuron could be used for summing uncorrelated input signals onto a single output line. Equations (6.37) and (6.40) showed how the time lag between the input and output spikes depended upon the recent spike history; i.e., spikes occurring within a spike train produce output spikes with less delay than did isolated spikes. (An isolated spike was defined as an input spike with no other recent input spikes preceding it, which were still influencing the neuron's current state.)

Section 6.5.3 described a synchronized identity neuron. For this neuron, the output spikes occurred at regular intervals. It behaved like a beating neuron (see Section 4.3.4), which had no output beats unless an input spike occurred. The neuron *almost* output spikes at a regular frequency, but its potential remained slightly below threshold; however, when a spike was input, it sufficiently increased the neuron's potential to produce an actual spike at the time of the next "pseudospike," which was the time of maximum neurotransmitter contribution from the beating neuron. This type of neuron was necessary for those computations that required synchronized input signals, such as the pattern recognition or logic function evaluation tasks described in Chapter 7. In designing a synchronized identity neuron, there were seven constraints that needed to be satisfied, which were explicitly stated on page 126-129 using Equations (6.43)-(6.50). The actual time of an output spike slightly led the pseudospike, with the bound on the maximum lead time given in Equation (6.52).

Section 6.5.4 described a synchronized inverse neuron. This neuron operated the same as the synchronized identity neuron, except that its output signal was inverted; i.e., the neuron only output a spike when no spikes were being input. If an input signal was connected to both a synchronized identity neuron and a synchronized inverse neuron, they would produce output spikes at opposite times; i.e., during the pseudospike events in which the synchronized identity neuron produces an output spike, the synchronized inverse neuron would not, and vice versa. This type of neuron was necessary for evaluating the logic functions described in Section 7.4. The design of this neuron was very similar to the synchronized identity neuron and has similar constraints, except that the neurotransmitter contribution from the pace neuron caused the neuron's potential to exceed threshold as described in Equations (6.53)-(6.54), and input spikes had an inhibitory effect. Like the identity neuron the actual time of an output spike slightly led the pseudospike, but the maximum lead time occurred when there are no input spikes (see Equation (6.55)).

### 10.2.3 Computing with the SNM

Chapter 7 developed complete networks which were capable of performing complex tasks. It began by discussing a stimulus detector network (Section 7.2), which could be used to phase lock (synchronize) different input spike trains. Since some of the other networks presented required synchronized input signals, this network acted as a preprocessor for them. The network could be configured so that the phase locking only occurs if the total activity on all of the input lines is sufficiently strong, and when the total activity is weak, no signals pass through. The number of neurons required for a stimulus detector is simply equal to the total number of input lines plus one, for the high gain pacing neuron. The actual synchronization process can be thought of as sampling the input signals, with output spikes indicating the presence of an input spike within the time window associated with each pseudospike. The sampling frequency was set by the parameters of the high gain neuron driving the system (see Equations (6.1)-(6.3)). While the output spikes of a stimulus detector had a fixed underlying frequency, the starting and stopping of such oscillations were determined by the input stimulus; hence, the output spikes were "unclocked." The pattern recognizer networks and logic neurons discussed in the Sections 7.3 and 7.4 only required that the timing between events remained constant, but the input patterns could start at any time.

Section 7.3 presented several different type of networks for pattern recognition. A distinction was made between temporal, spatial, and spatial-temporal patterns. A temporal pattern consisted of one input signal with a sequence of spikes. A spatial pattern used several inputs, but with only one target event on each input line. (Events could be either spikes or non-spikes.) A spatial-temporal pattern used several input signals with at least two target events on one or more of the input lines. The network configuration depended upon the type of pattern being recognized.

For the temporal pattern recognition task described in Section 7.3.2, each neuron was designed to create a dividing hyperplane in the input space, where the dimension of the input space was defined by the number of events within the target pattern.

The neuron's parameters were chosen according to Equations (7.2)-(7.9). While the neuron was designed for input events occurring at a specific frequency, the location of the dividing hyperplane determined how much the frequency could differ from this frequency and still have the neuron recognize the pattern. Figure 7.7 showed how these neurons could be used in the first layer of a three layered network for more complex pattern recognition tasks. The second layer neurons were designed to "AND" their input spikes and only produce output spikes when all of their inputs were firing, while the third layer neuron was designed to "OR" its input spikes and produce an output spike when any of its input spikes were firing. After the first layer neurons decided which side of their respective hyperplanes the input signal lied on, the task of deciding if an input resides within a set of target regions was reduced to evaluating a Boolean logic expression. Since every Boolean logic expression can be reduced to conjunctive normal form (see footnote on page 153), a three layer network is capable of performing *any* classification problem. The number of neurons in the first layer equals the number of required hyperplanes; the number of neurons in the second layer equals the number of convex target regions in the input space (non-convex regions can be considered to be the union of convex regions); only one neuron is every required for the third layer, and it is only necessary if there is more than one convex region.

For the spatial recognition task, described in Section 7.3.3, a single neuron was designed to recognize a particular spiking pattern occurring on a set of input lines with one target event per line. When the neuron's parameters satisfied the constraints in Equations (7.10)-(7.11), the events in the target pattern could occur at different times, and the input lines could have different frequencies. If there was more than one target event on one or more of the input lines, the problem was a spatial-temporal recognition task, described in Section 7.3.4. Figure 7.9 showed that when a layer of temporal recognizing neurons was followed by a layer of spatial recognizing neurons, the network could recognize a particular spatial-temporal pattern. While the effect of this network was only to create a single dividing line in the spatial-temporal input space, several of these networks could be used with their outputs feeding into "AND" neurons whose outputs are then fed into an "OR" neuron, allowing such a network to perform any spatial temporal recognition task in much the same way as was described for the temporal recognition task. Section 7.3.5 went on to describe how variations in the input event frequencies affected the ability of these networks to recognize their target pattern.

The number of neurons within the second and third layers of a multilayered pattern recognizing network could be reduced to a single logic neuron, which was described in Section 7.4. A logic neuron can be used to evaluate any Boolean logic expression. The expression must first be reduced to conjunctive normal form (CNF), which is a product (AND) of sums (OR). Essentially, the neuron used its synaptic clusters to evaluate each of the sums, and its potential performed the product operation. An output spike is produced only if all of the synaptic clusters are receiving spiking inputs. The number of synaptic clusters required equals the number of clauses (products) in the CNF of the Boolean expression, and the number of inputs into each cluster equals the number of literals (variables) summed within its corresponding clause.

Section 7.5 showed how a network with feedback connections could be used to store entire segments of an input spike train. It could be thought of as a short-term memory, which requires no modifications to the network parameters; i.e., no learning. The spike pattern was preserved indefinitely, until it was replaced with a new memory. The total length of the memory was determined by the amount of delay on the feedback connections.

A counting network can be used to determine the number of spikes within a spike train. Section 7.6 described two different types of counters. The first was a linear counter, in which the number of neurons oscillating in the network represented the number of input spikes. The number of neurons in the network sets the limit as to how high it can count. The second was a sequential counter, in which only one neuron at a time oscillated, but each neuron in the network represented a specific number. The sequential counters could be cascaded together to create networks which encode different digits in a number. The number of cascaded networks determines the number of "digits" encoded in the network, and the number of neurons in each counting network sets the base for the encoding scheme; e.g., Figure 7.20 showed how two sequential counters with five neurons each could be cascaded together. Thus, this network encoded two digits of a base five number.

Section 7.7 presented a multiplexer network, which selects one of many input signals to be transferred to a single output. This network can be particularly useful for routing data within a complex nervous system. The network required two neurons for each input signal, plus one for the summing neuron. The section also discussed how a demultiplexer could be constructed from a multiplexer. A demultiplexer routes one input to one of several possible outputs.

While the pattern recognizing networks were useful for determining if an input spike train contained a specific spike sequence, it is sometimes necessary to determine if two or more spike trains are equal to each other. Section 7.8 described a comparator network, which could be used for such a task. If the input spike trains were equivalent, the network would output spikes, regardless of the actual input patterns. While the network shown in Figure 7.22 only compared two input signals, more signals can be compared by including one synchronizing identity neuron and two difference detecting neurons for each additional input. The number of synapses on the pattern comparator neuron sets the length of the patterns being compared.

And finally, Section 7.9 described how a Hopfield-like associative memory network could be constructed from spiking neurons. The basic problem was to store a set of patterns in such a way that when the network was presented with a new pattern, it responded by producing whichever one of the stored patterns most closely resembled the input pattern. The stored patterns could be taken to be either a 0 or 1 value at each neuron in the network, where 1 indicated that the neuron was oscillating, while 0 indicated that it was not.

### 10.2.4 Encoding with Spikes

Of course, when attempting to compute with spikes, it is necessary to consider how information is actually being encoded. Chapter 8 showed how neurons could be specifically designed to encode information. A synchronized identity neuron could be modified so that while an output spike indicated the presence of an input signal, the signal's strength was encoded within the spike's phase. This encoding scheme could actually be used within a two layer network, where the phase transfer functions of the first layer neurons were sigmoidal, and the phase transfer functions of the second layer neurons resembled radial basis functions. It was then shown that the strength of an input stimulus might also be logarithmically encoded in the amount of neurotransmitter released at a synapse. These encoding schemes could be exploited within a pattern recognizing networks so that it is able to respond to inputs with a particular ratio, over a wide range of strength magnitudes.

### 10.2.5 Learning Methods

Finally, a new network architecture and learning rule for networks of either spiking or threshold (McCulloch-Pitts) neurons were developed. The network used two layers, but all of the inputs connected to *both* the hidden and output neurons, and formed only excitatory connections. The hidden neurons connected just to the output neurons and formed only inhibitory connections. The learning was similar to that of the Perceptron; however, the weight changes were probabilistic, and all of the weights within the network were restricted to positive integer values. Also, since the network had two layers of neurons with nonlinear transfer functions, it was capable of solving problems that were linearly inseparable. The weight changes determined by the learning rule had the same sign as they would have if the neurons' outputs were sigmoidal and backpropagation learning was used. But unlike backpropagation, the neurons were not required to have a differentiable transfer function, and the error signal did not need to be propagated backwards through biologically implausible bidirectional connections.

## 10.3 Contributions of Thesis

### 10.3.1 The SNM

This thesis presented a new spiking neuron model (SNM). Although the mathematics behind the model was not based on any differential equations, the SNM was able to capture many of the properties believed to be important in real neurons, including: latency, refractory periods, and oscillatory spiking behavior. The state of a neuron is set by the amount of neurotransmitter at its input synapses and the time since it last fired. The SNM has several advantages over other neuron models:

1. While being able to duplicate many of the complex properties observed in real neurons, it is much simpler than those realistic channel-based models which attempt to simulate the ionic current flow through a neuron's membrane with differential equations. The best known examples of these are the Hodgkin-Huxley model [43] and the FitzHugh-Nagumo model [23, 79]. Unfortunately, these models are too complicated for a detailed mathematical analysis, and even when simulated on a computer, simplifying assumptions are usually unavoidable [59, pp. 1-2].

2. Although the SNM does not attempt to model a neuron's morphology, it is able to account for the time delays associated with the propagation of an action potential from the axon hillock to the output synapses, the accumulative effects from several synapses converging together at a synaptic cluster, and the ability of one synapse to affect the output neurotransmitter at another through presynaptic inhibition. While there has been a great deal of research into the modeling of the axon and dendritic branches (e.g., [9, 85, 86, 87, 88, 100]), the SNM is able to duplicate many of the computations believed to be performed within the neuron's structure (e.g., computer direction of motion, recognize spatial-temporal patterns, and implement logic operations [101]) without have to model it explicitly. While the modeling of the ionic currents and dendritic branches may be necessary by neurobiologists to *understand* the computations

being performed by neurons within biological systems, the SNM assumes that such details are not needed to *duplicate* the computations.

3. Unlike the leaky integrate-and-fire (LIF) models (e.g., [105, 13]), when the SNM fires, all of the information concerning the previous input signals is not lost; i.e., when input spikes arrive, they release neurotransmitter at the synapses, and this neurotransmitter is still there to affect the neuron's potential even after it fires. In most LIF models, the production of an output spikes completely resets the neuron's state, including its inputs. (While LIF models provide an analytically tractable description of the firing rate in terms of a neuron's time constant, threshold, and refractory period, they have mainly been used to model physiologically realistic spike trains, and not used for computational purposes [107].)

4. The dynamics of the SNM described in Equations (3.1)-(3.2) are *not* formulated as differential equations. Consequently, the stability of the simulated dynamics is not affected by the integration step size. Chapter 5 presented a novel simulation technique called "Simulating in Time Segments," which was specifically designed to rapidly simulate networks of the SNM.

Of the neuron models described in literature, the SNM is most similar to the "Spike Response Model" (SRM) of [25, 26]. In this model, the effect from each of the incoming spikes has a functional form of $te^{-t}$, which is analogous to the input neurotransmitter, $U_{ij}(t)$, in the SNM. However, there are several significant differences:

1. In the SRM, the effect of the neurotransmitter on the neuron's potential is not bounded by the $\tanh(-)$ function as it is for the SNM. Thus, a single synapse with a large amount of neurotransmitter can have an unrealistic influence over the neuron's potential. By using the $\tanh(-)$ of the neurotransmitter at a synapse, a burst of input spikes is able to saturate a synapse, which causes its effect on the neuron's potential to be relatively constant and not depend upon the high frequency effects from the individual spikes. This effect was exploited in the passing neurons of the multiplexer described in Section 7.7.

2. While the SRM uses delay for the axonal transmission time, it is a single delay parameter for *all* of the synapses within the network. Thus, it is not possible to account for difference in propagation times between different neurons. Without different delays associated with different synapses, many of the networks described in this thesis would not be possible, including: memory oscillator neurons, pattern recognizers, memory networks, counters, comparators, and Hopfield networks with limit cycles.

3. For the SRM, each postsynaptic neuron has a single synaptic time constant describing the neurotransmitter decay at all of its synapses, rather than a separate time constant associated with each synapse. Without using different synaptic time constants in the SNM, it would not be possible to design the synchronized identity/inverse neurons required for the stimulus detectors, logic networks, memory networks, comparators, and phase encoding schemes.

4. The neuron's refractory term takes a different form, and is additive rather than multiplicative; i.e., while the potential in the SNM is determined by the product of the refractory coefficient and the effects from the input synapses, the potential in the SRM is determined by the sum of the effects from the input synapses plus a refractory potential term, which goes to $-\infty$ immediately after the neuron fires and then rebounds slowly to zero. While the properties of the refractory coefficient were used for the logarithmic encoding scheme described in Section 8.4, the results in [107] (for a LIF neuron - not the SRM) seem to suggest that any model with an absolute refractory period can be used for logarithmic encoding over a limited range of inputs. Thus, it is unclear if one method for enforcing a refractory period is more advantageous than another.

5. The production of a spike in the SRM is probabilistic rather than deterministic. In reality, the probabilistic approach is more biologically plausible; however, it significantly complicates the analysis. The way in which the probability of an output spike is determined in the SRM is by comparing the difference between the neuron's potential and threshold. As the potential increases above

threshold, the probability of an output spike increases. While there are some differences in how the potentials are calculated, a similar probabilistic function can be applied to the SNM's potential to determine output spike production; i.e., the probability of the SNM producing an output spike can be expressed as:

$$P = \frac{1}{1 + e^{-\left(\frac{V_i - \Theta_i}{T}\right)}} \tag{10.1}$$

Here, the parameter of $T$ determines the amount of noise in the system. (See footnote on page 7.) As $T \to 0$, the SNM exhibits noise free behavior and becomes deterministic. Thus, if desired, it is easy to modify the SNM into a probabilistic framework.

Overall, the SNM fulfills its purpose of providing a relatively simple spiking neuron model which can be used for spike based computations. It is able to produce spike trains comparable to those of real neurons. While it does not attempt to model the details of a neuron's morphology, it has a sufficient number of parameters for duplicating some complex behaviors that simpler spiking models are unable to capture.

### 10.3.2 Computational Networks

Most spiking neuron models have been developed by neurobiologists for the purpose of understanding real neurons. Few researchers have studied spiking neurons for use in computing paradigms, and of those that have, most of the research has involved theoretical studies on information encoding or the computational limits of a network of spiking neurons (e.g., [67, 68]). (Section 10.4.1 describes how these results apply to the SNM.) This thesis presents some specific network configurations for performing complex computations. Through the use of synaptic delays and accumulative neurotransmitter, the SNM is able to easily perform temporal tasks that would not otherwise be possible, e.g., temporal pattern recognition, storing temporal patterns in memory, comparing temporal sequences, and storing limit cycles within a Hopfield network. Of the research that has involved the construction of actual networks of spiking neuron for a particular purpose, most has involved Hopfield associative memory networks. [25, 26] showed how a network of SRM neurons could be used to store a single spiking pattern based on rate encoding, while [69] showed how a network of compartmental neuron models, simulated with GENESIS (see [9]), could store multiple patterns using phase encoding. However, neither paper demonstrated how oscillatory limit cycles could be encoded, as demonstrated in Figure 7.29.

### 10.3.3 Encoding Information

Most research into networks of spiking neurons has assumed that information is encoded as an analog variable in terms of the firing rates. This thesis presented several alternative encoding schemes, and the one being used depended upon the task that the network was designed to perform. (In some cases, the spikes from different neurons in the same network have different encodings; e.g., each input spike into the counting networks represented one bit of information, while the oscillatory/silent states of the output neurons also represented one bit of information.) Chapter 8 showed how information can be encoded within the phase of an output neuron. Section 8.3.2 showed that when the SNM is receiving a slowly varying input signal, its parameters can be set so that the phase of its output spike, relative to a pseudospike produced from a pace neuron, encodes the magnitude of the input stimulus. If the neuron's parameters are such that it always outputs a spike with each input pseudospike, then its phase transfer function can be extremely well approximated by the $\tanh(-)$ function (see Equation (8.22). But if the neuron only outputs a spike when its input signal is above a certain level, then the phase transfer function can be approximated by the $\sqrt{\tanh(-)}$ function (see Equation (8.43)). Section 8.3.4 showed that when a neuron does not have a slowly varying input signal, but instead receive spikes from another neuron that is also phase encoding the strength of its stimulus, then the transfer function of its phase resembles a radial basis function. (The advantage of radial basis functions is that only one layer of hidden units is needed to represent any reasonable function [42, pp. 143, 248-249].) Thus, layered networks can be constructed from neurons which transmit information within their output spike phases. The neurons in the first layer use a sigmoidal transfer function, while the neurons in the subsequent layers use a radial basis transfer function.

Maass [67] showed how the strength of an input signal could be encoded in the output phase of a SRM-like neuron, where the time between the output spikes of a neuron and a pool of synchronously firing neurons resembles a sigmoidal function. (Unfortunately, these results could not be easily generalized to the SNM; however, the phase encoding schemes derived in Section 8.3 were motivated by these results.) The problem with this derivation was that it assumed that when a spike arrives at a synapse, its effect on the postsynaptic neuron's potential was linear (at least until the time that the postsynaptic neuron fired), and this can only be true if the time constant associated with the neurotransmitter is much larger than the maximum possible phase encoding. The derivation for phase encoding with the SNM made no such restriction. Also, except for the sigmoidal transfer function, no other transfer functions were demonstrated.

Section 8.4 showed how the SNM is able to approximate a logarithmic encoding of the input signal in its *output spike frequency*. This is consistent with some experimental results from biology (e.g., [89, 16, 63, 114]). Hopfield [47] was the first to suggest such a logarithmic encoding scheme using the *phase* of an output spike, relative to the network oscillations (analogous to the pace spikes used in the stimulus detector). He pointed out that such an encoding could be used by neurons which respond to coincident input spikes, with the time delays associated with the input synapses determining the *relative* strengths necessary for recognition of a pattern (analogous to the spatial recognition neuron). However, he did not present any neuron model or suggest how such a logarithmic encoding could actually be accomplished. With the SNM, the average neurotransmitter at a synapse is directly proportional to the input spike frequency (see Equation (4.27)), which allows the logarithmically encoded frequency to be used as the input signal of a phase encoding neuron. Thus, its output phase represents the logarithmically compressed input signal, and Hopfield's encoding scheme can be accomplished with two successive neurons. These spikes, with logarithmically encoded output phases, could then be input into the pattern recognizing networks of Section 7.3 to respond to input signals with a particular ratio. This allows the network to recognize patterns based upon their *relative* input strengths, independent of their *absolute* magnitudes. (This was the motive behind Hopfield's suggestion that neurons might encode the strength of their input stimulus logarithmically in their output phases. See [47].)

Finally, if there are two sensory neurons logarithmically encoding the same input signal in their firing rates, but they have slightly mismatched parameters, then the time duration of the signal is linearly encoded within the phase difference between their output spikes. (See Hopfield [48] for a discussion of why it is necessary to encode the duration of a stimulus for some problems.)

Thus, this thesis demonstrated that a single spiking neuron model, which captures many of the properties of real neurons, is able to encode information in a variety of ways. While some of these encoding schemes have been observed/postulated for real neurons and demonstrated in a variety of spiking models, this thesis shows how all of these encodings can result from a single model, and provided examples of computational networks where different encoding schemes are advantageous.

### 10.3.4 Learning Methods

This thesis presented a new learning paradigm for use with the SNM (or networks of binary threshold neurons, i.e., McCulloch-Pitts neurons [71]). It is based on a special layered network structure, where each neuron is either all excitatory or all inhibitory, and can not change during the learning process. Since the transfer function of a spiking (or threshold) neuron is discontinuous, the actual adjustments are similar to the Perceptron learning rule (see [94]); however, it uses a two layered network, which is capable of solving problems that are not linearly separable. (Perceptrons can only solve linearly separable problems - see [75]). Furthermore, the weights are restricted to positive *integer* values, and the occurrence of each weight adjustment is probabilistic. While little global information is required, Section 9.5.4 proves that the weight adjustments are actually in the same direction as they would be if gradient descent were applied to a network of neurons with sigmoidal transfer functions. The results showed that the network was capable of learning all mappings (pattern recognition tasks) that it was tested on. Thus, this thesis demonstrated that it is possible to implement learning within a network of spiking neurons, while the actual learning method developed was not limited to spiking neurons and could be used to train networks of threshold neurons.

## 10.4 Implications of Results

### 10.4.1 Computational Ability

When the output spikes of neurons are synchronized through a pace neuron, the occurrence or non-occurrence of a spike can be equated to 1 or 0 in a clocked binary system. Thus, a network of synchronized spikes is completely equivalent to a network of binary threshold (McCulloch-Pitts) neurons, and any function that can be performed with such neurons can also be performed with a network based upon the SNM. (Compare Figures 9.6-9.7.)

Among the computational networks presented in Chapters 6-7 are the basic building blocks required for constructing a microprocessor. According to [3], the basic components of a microprocessor are: (1) digital flip-flops, which store the state of a single binary input; (2) gates, which control the flow of data through a device by being either open or closed; (3) registers, which are logic circuits comprised of discrete memory devices (digital flip-flops with associated gating circuitry); (4) counters, which increment themselves based on their inputs; (5) decoders, which are logic circuits used to select data; and (6) multiplexers/demultiplexers, which are used to route data. The memory oscillators of Section 6.3 are equivalent to digital flip-flops. The controlling oscillators and passing neurons of the multiplexer network shown in Figure 7.22 are equivalent to gates. Registers can be easily constructed from memory oscillators, with one oscillator used for each bit of the register. Several counters were presented in Section 7.6. Although no networks were presented for a decoder, it can be constructed from the multiplexer's "gates." (A decoder has $N$ inputs, which encode a binary number, and selects one of its $2^N$ outputs.) Section 7.7 presented a multiplexer circuit and described how it can easily be modified into a demultiplexer. Therefore, in theory it would be possible to construct an entire microprocessor out of the SNM; however, this would not take advantage of the parallel computing capabilities associated with neural networks. Furthermore, the network would no longer be able to deal with noisy and precise inputs and provide reasonable outputs, nor be robust with respect to its neurons. Thus, networks of spiking neurons are best used for those applications where digital computers are inept.

While this thesis did not attempt to prove any bounds on the computational power of networks of spiking neurons, many of the previously derived results for other neuron models apply. Perhaps the most significant of these were given by Maass:

1. In [68], Maass showed that a relative general spiking neuron model (of which the SNM satisfies all of the necessary constraints) can be used in networks to simulate in real-time arbitrary threshold circuits, finite automatons, or Turing machines. It is interesting to note that the construction of these devices was based on modules that synchronize the spiking of different network parts in much the same way that the pace neurons in many of the computational networks work, as presented in Chapter 7. Furthermore, information is encoded in the phase difference between neurons, as described in Section 8.3. One important consequence of this result is that in order to show that a network of spiking neurons can carry out a specific task, it suffices to show that either a threshold circuit, a finite automaton, or a Turing machine can carry out that task in an efficient manner. Also, this result yields a lower bound on the VC-dimension of networks of spiking neurons, and hence on the number of training examples needed for learning by such networks. (Maass did not present any learning algorithms.)

2. In [67], Maass showed that when analog input and output signals are encoded

within the phase of a neuron's spikes, a feedforward network of noisy spiking neurons with one hidden layer is a "universal approximators" in the sense that it can approximate with regard to temporal coding any given continuous function of several variables. The phase encoding scheme used by Maass resembled a sigmoidal transfer function of the input strength; however, his results apply to any continuous phase encoding scheme. (Chapter 8 showed how in feedforward networks of the SNM, only the phase encoding scheme used by first the layer of neurons is a sigmoidal function, while the phase encoding scheme used by subsequent layers resembles a radial basis function.) Thus, networks of the SNM can also be used as "universal approximators." While the standard SNM does not include the effects of noise, the results did not depend upon the neurons being noisy, but instead demonstrated that computations are robust with respect to noise. (See Equation (10.1) for modeling noise in the SNM.)

Together, these two results imply that networks of spiking neurons have in fact strictly more computational power then analog networks of roughly the same size and depth. But, while Maass's papers [66, 67, 68] derived bounds on the theoretic computing capabilities of networks of spiking neurons, they did not show how any networks could be constructed to perform specific tasks, which was one of the goals of the research presented in this thesis.

### 10.4.2 Biological Systems

While this thesis was not a neurobiology thesis, much of it was inspired by neurobiology, and some of the results presented in it have biological implications. Of course, it would be ridiculous to suppose that the simple SNM and the computational networks based on it are even close to the brain's intricate computational machinery. While many of the networks developed perform functions that the brain is obviously capable of performing, this thesis offers no evidence that the networks in the brain actually resemble these. However, the mechanisms employed by these networks suggest that it is possible for similar paradigms to exist in biology.

The thesis began by developing the SNM, which was meant to capture a few of the salient properties of real neurons, and then developed a few basic networks which were capable of performing some of the well-known functions associated with biological systems, such as pattern recognition and memory. Obviously, the fact that a network of spiking neurons was able to performing these functions comes as no surprise to any neurobiologist, the only revealing part is the level of detail required in the model and the methods in which the computations were performed. While allowing for presynaptic inhibition and synaptic clusters, the SNM ignored all geometry associated with a neuron's axon and dendrites (except through the time delay and time constant parameters associated with each synaptic connection).

Some of the networks used "synchronized" input spikes (pattern recognizers, logic neuron), or had synchronizing preprocessing neurons on its front end (memory networks, comparators). However, these spikes were not synchronized in the traditional sense. First, when a high gain neuron was used as the pacing neuron (see stimulus detector in Figure 7.1), the synchronization process was triggered by the inputs themselves, and there was no master clock driving the system. If several of these networks were used together within a larger system, there would be no need to have an overall synchronization process for the entire system; i.e., the synchronizations could be relatively local to the individual tasks. Second, the synchronization process did not necessarily cause an alignment in the output spikes. Notice that very few of the output spikes of the stimulus detector shown in Figure 7.2 were actually aligned with each other, and if the network used different delays from the pacing neuron to the identity neurons, none of the spikes would be aligned.[1] Furthermore, when the output phase of the spikes relative to the pacing spike, encodes the strength of the input stimulus, as described in Section 8.3, the spikes are even further out of alignment. Thus, since none of the spikes actually align, and most of the neurons are not producing spikes at a regular frequency, an outside observer would have a very difficult task detecting any synchronization at all. However, the key indicator

[1]It is not a problem to have misaligned spikes, provided that the delays on the synapses to which they connect are appropriately set to align their *effects* on the neurons to which they connect.

of synchronization in such a system is: *the time between output spikes of different neurons are integer multiples of the period of the synchronizing signal.* Therefore, while the output spikes of the stimulus detector shown in Figure 7.2 may not align, the underlying frequency of events (both spikes and non-spikes) is the same as that generated by the high gain pacing neuron. And while the frequency of the high gain neuron may slightly vary with its input signal (see Equation (7.1) and Figure 6.1), many of the networks presented in this thesis required the frequency of events to be relatively stable so the synaptic delays could be appropriately set to accomplishing the desired task. (See Section 7.3.5 for a discussion of problems associated with a variable synchronization frequency.) Thus, by observing enough of the output spikes from neurons within a synchronized network, the synchronization frequency can be determined;[2] however, it is possible for different networks within the system, which have their own synchronization machinery, to use different frequencies.

While a large system may have different network components with their own synchronizing pace neurons, it is also possible for the spikes from these neurons to be connected to each other through inhibitory synapses. The synaptic parameters could be chosen so that the pacing neurons implement a winner-take-all behavior; i.e., only one pacing neuron produces output spikes, while the others are inhibited. A large input stimulus would be required to overcome the inhibitory input and cause one of the "off" stimulus detectors to switch "on" and become the "winner." In this sense, the structure of the stimulus detector network can be considered a primitive model for consciousness. Strong input stimuli draw the system's *attention*, resulting in active information processing. Weak stimuli still enter the system, but do not lead to active processing. The system's "awareness" can switch between different types of stimuli, depending upon their input strengths.

Chapters 7 and 8 demonstrated that it is possible to have a variety of encod-

[2]It is possible to design the stimulus detector so that input spikes are still passed through to the output lines without there being a sufficiently strong stimulus to create the necessary beating behavior in the high gain neuron required for synchronization. In this case, these unsynchronized output spikes would not be correctly interpreted by the next layer of neurons. Thus, it may be necessary for the network to be "actively processing" its inputs to guarantee that the synchronization is actually occurring.

ing schemes all based on the same neuron model, namely: (1) the oscillation/silent states of a neuron can encode a single bit of information (e.g., memory oscillators - Section 6.3); (2) a single event (spike/non-spike) within a synchronized spike train can encode one bit of information (e.g., pattern recognizers – Section 7.3); (3) the output frequency of a neuron can encode the strength of its input stimulus as a sigmoidal-like transfer function (Section 4.4); (4) the output frequency of a neuron can logarithmically encode the strength of its input stimulus (Section 8.4); (5) the phase difference between an output spike and the input spike from a pace neuron can encode the strength of an input signal as a sigmoidal transfer function (Section 8.3.2); and (6) the phase difference between an output spike and the input spike from a pace neuron can encode the strength of an input signal as a radial basis transfer function (Section 8.3.4). Other neuron models have also been shown to have similar encoding possibilities. [107, 13] showed how a leaky integrate-and-fire neuron is capable of logarithmically encoding the strength of an input stimulus in its output frequency. (Although a quantitative comparison has not been made, the logarithmic compression shown in Figures 8.7-8.8 for the SNM appears to be much more accurate over a larger range in frequencies.) [67] showed how a sigmoidal transfer function in the output phase of a neuron is possible for a similar neuron model (i.e., the SRM neuron). While it is difficult to speculate on the complex operations performed within a real nervous systems, it is reasonable to postulate that the brain probably also uses variations in encoding. For at least some tasks, it has been observed/hypothesized that real neurons use a logarithmic transfer function for their output frequency (e.g., [89, 16, 58]), while for other tasks the timing of single action potentials may be used to encode information (e.g., [1, 4, 7, 47, 91, 102]). Thus, the research presented in this thesis supports the possibility of these various encoding schemes, and further suggests some alternative ones.

Finally, the new learning method developed in Chapter 9 suggests that the differentiation of cells as being all inhibitory or excitatory may not just occur for biological convenience, but may have important implications in learning. With the new network architecture developed (see Figures 9.6, 9.7, and 9.11), the input signals connect to both an inhibitory (hidden) layer and the output layer. These inputs only have an excitatory effect on all of the neurons to which they connect, while the hidden neurons only have an inhibitory effect on the neurons to which they connect. The weights (interconnection strengths) were restricted to positive *integer* values. This is analogous to assuming that: *there is not significant variation between the strengths of synapses, but the amount of influence that one neuron has on another can be predominantly controlled through the creation or destruction of synapses.* Thus, while there has been a great deal of interest expressed in the ability of synapses to modulate their interconnection strengths through LTP and LTD (e.g., [70, 106]), these results indicate that learning may be predominantly controlled through the growth of new synapses and the pruning away of existing ones, which is consistent with the development of the nervous system in infants and toddlers, where the number of synapses to each neuron grows to a maximum at about age two, and then from age two until age seven the number of synapses reduces by about one-half. This drop occurs during the intense learning of basic skills, and after age seven, the population remains nearly constant [35, p. 23].

While it would be extremely naive to suggest that the learning rule used in biological systems (or even to suggest that there is a single learning rule) is similar to the primitive method developed to learn binary mapping as in Equations (9.21)-(9.24), the details of this rule have some possible biological implications. First, learning can be controlled through the probabilistic formation/destruction of synapses. Second, it is possible to have a learning rule that requires very little global information. To adjust the connections into the output neuron, it is only necessary to know the actual output, the desired output, and the outputs of the neurons forming the connections. While the synaptic changes are in the same direction as gradient descent (i.e., each individual change is guaranteed to be in the direction which reduces the error for the current input/output mapping), there is no need for bidirectional connections. Instead, such a rule could be implemented within a biological system, with a special neuron releasing a chemical if the output is incorrect, and another special neuron releasing a chemical when the desired output is 1 (or 0). These chemicals could diffuse through the network, with the first increasing the probabilities for synaptic changes and the second indicating which synapses should increase or decrease.

## 10.5 Future Directions

Obviously, this thesis only represents an incremental step toward the ultimate goal of building more powerful computational devices based on neurobiology. While the simple networks presented were able to perform a variety of tasks, they need to be integrated together into much larger networks to become truly useful. However, as long as these networks are only simulated on a serial computer, their use is somewhat restricted. The primary advantage of neural networks lies in their parallelism, and it will never be feasible to simulate very large networks in real time on a serial computer. Thus, the next step in developing useful devices is to construct networks in hardware, i.e., silicon VLSI circuits.

While some progress has already been made in the area of VLSI implementation, most artificial neural networks are only simulated in software on a computer [17, p. 17]. One of the main problems with making neural network chips is that they require a lot of interconnections between neurons, and the space taken up by these connections becomes the limiting factor in the size of the network [42, p. 9]. However, with the SNM, it is theoretically possible to significantly reduce the number of interconnections between neurons.

Since the output spikes of the SNM are discrete events, only the times of the spikes matter. If the probability of any two neurons producing nearly simultaneous output spikes is low, then the output spike signals can be sent out on an address bus of $\lceil \log_2 (N+1) \rceil$ lines that connects to all $N$ neurons in the network. Each neuron would have a unique address, and could only output spikes on the lines representing the binary encoding of its address. While all of the neurons in the network would receive information about each firing neuron, only those synapses set to receive spikes with that particular address encoding would be influenced.

For example, a network with 15 neurons would only require four address lines,

($\lceil \log_2(16) \rceil = 4$), numbered 0 through 3. If the sixth neuron in the network produced an output spike it would simultaneously send spikes along the bus lines numbered 1 and 2, which represent the binary encoding of six, ($6 = 2^2 + 2^1$). All of the neurons in the network would receive the output spikes on these two lines, but the input synapses on each neuron would be designed to respond to only one particular spatial spike pattern, which represents the address of the neuron forming the connection. All of the synaptic parameters could be stored at the synapse and would not need to be transmitted by the spiking neuron. Thus, the information concerning which neuron fired is encoded within a binary address, and the *time* of the output spike is determined by the occurrence of the actual spikes along the address line.

Since many of the networks presented in this thesis used synchronous spikes, the restriction of non-simultaneous spikes may seem troubling. However, the input signal from a pace neuron into the synchronizing identity neurons may have different delays; e.g., pace synapses of the stimulus detector in Figure 7.1 may have different delays, causing all of the output spikes in Figure 7.2 to still have the same underlying frequency of events, but be misaligned. Computationally, this is not a problem when the synaptic delays of the neurons to which these output spikes connect are set so that their *effects* align.[3] Thus, the "synchronization" process can actually be used to guarantee that simultaneous spikes do not occur.

Furthermore, if the synapses could be designed so that the address to which they respond can be changed, then the network structure can effectively be modified after the chip is constructed. It would then be possible to build a general purpose network chip, where the neuron interconnections could be set as necessary for the task to be performed. Thus, this would represent yet another step towards the development of more powerful computing devices, which will someday rule the earth and destroy mankind.

[3]Notice that the synaptic delays would need to be implemented at the synapses and not through the propagation of spikes to the synapses. This could be accomplished by having an input spike trigger the release of current, which charges up a capacitor at the synapse. When the voltage on the capacitor reaches some preset level, it then triggers the "neurotransmitter circuitry," which directly affects the potential of the neuron. The amount of current used to charge up the synaptic capacitor would set the time delay associated with the synapse.

# Appendix A Properties of $F(x)$ and $G(x)$

The defining equations for $F(x)$ and $G(x)$ are:

$$F(x) \equiv x\mathrm{e}^{-x} \tag{A.1}$$

$$G(x) \equiv \mathrm{e}^{-x} \tag{A.2}$$

Also, the functions of $f(x)$ and $g(x)$ are defined as $F(x)$ and $G(x)$ multiplied by $u(x)$, the unit step function (see Equations (4.2), (4.7), and (4.4)).

The functions of $F(x)$ and $G(x)$ are shown in Figure A.1 for $x \geq 0$. Notice:

$$F(x) = G(x) = \mathrm{e}^{-1} \quad \text{at:} \quad x = 1 \tag{A.3}$$

$$\max\{F(x)\} = \mathrm{e}^{-1} \quad \text{at:} \quad x = 1 \tag{A.4}$$

$$\lim_{x\to\infty}\{F(x)\} = \lim_{x\to\infty}\{G(x)\} = 0 \tag{A.5}$$

$$\int_0^\infty F(x)dx = \int_0^\infty G(x)dx = 1 \tag{A.6}$$

The derivatives of $F(x)$ and $G(x)$ are given by:

$$\begin{aligned} F^{(n)}(x) &\equiv \frac{\partial^n}{\partial x^n}F(x) = (-1)^n(x-n)\mathrm{e}^{-x} \\ &= (-1)^n\left(F(x) - nG(x)\right) = \left(\frac{-1}{\mathrm{e}}\right)^n F(x-n) \end{aligned} \tag{A.7}$$

$$G^{(n)}(x) \equiv \frac{\partial^n}{\partial x^n}G(x) = (-1)^n\mathrm{e}^{-x} = (-1)^nG(x) \tag{A.8}$$

Notice from Equation (A.7) that $G(t)$ may be expressed as:

$$G(t) = F(t) + \dot{F}(t) \tag{A.9}$$

Also notice that Equations (A.7)-(A.8) actually hold for both positive and negative

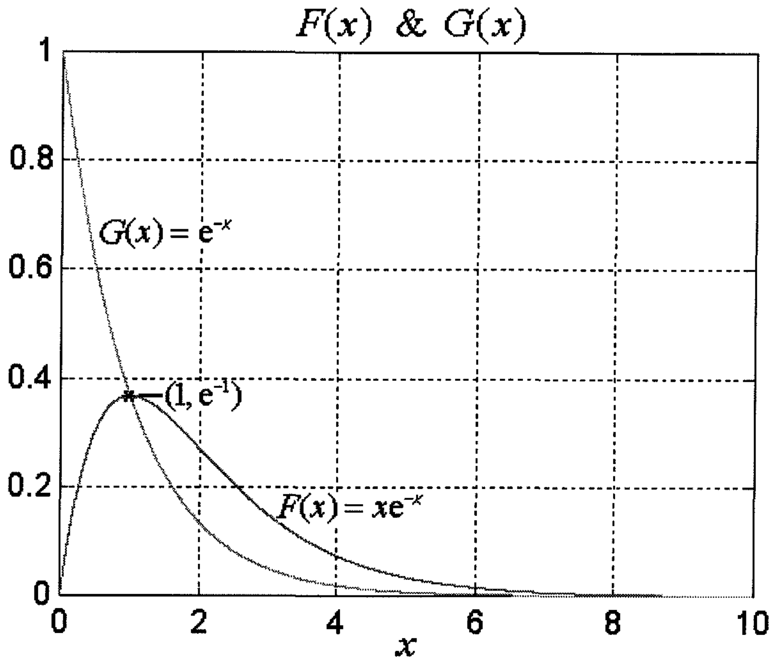


Figure A.1: **Plot of $F(x)$ and $G(x)$.** The function of $F(x)$ rises to its maximum of $\mathrm{e}^{-1}$ at $x = 1$ before decaying back to 0. The function of $G(x)$ decays to 0.

values of $n$, when negative values are interpreted as integrals of $F(x)$ or $G(x)$, e.g.:

$$F^{(-n)}(x) = \underbrace{\int dx \cdots \int dx}_{n} \cdot F(x) = (-1)^{-n}(x+n)\mathrm{e}^{-x} \tag{A.10}$$

$$G^{(-n)}(x) = \underbrace{\int dx \cdots \int dx}_{n} \cdot G(x) = (-1)^{-n}\mathrm{e}^{-x} \tag{A.11}$$

In fact the derivatives and integrals of $F(x)$ and $G(x)$ are related according to:

$$F^{(n)}(x) + F^{(-n)}(x) = 2(-1)^n F(x) \tag{A.12}$$

$$G^{(n)}(x) + G^{(-n)}(x) = 2(-1)^n G(x) \tag{A.13}$$

Now, the time evolution of $F(x)$ and $G(x)$ can be expressed as:

$$F(x) = F(x_o + \delta) = (x_o + \delta)\mathrm{e}^{-(x_o+\delta)} = G(x_o)F(\delta) + F(x_o)G(\delta) \tag{A.14}$$

$$G(x) = G(x_o + \delta) = \mathrm{e}^{-(x_o+\delta)} = G(x_o)G(\delta) \tag{A.15}$$

Finally, a linear combination of $F(x)$ and $G(x)$ may be expressed as a time shift in $F(x)$ multiplied by a scaling constant; i.e.:

$$\begin{aligned} AF(x)+BG(x) &= (Ax+B)\cdot \mathrm{e}^{-x} \\ &= A\mathrm{e}^{\left(\frac{B}{A}\right)}\cdot\left(x+\frac{B}{A}\right)\cdot \mathrm{e}^{-\left(x+\frac{B}{A}\right)} \\ &= A\mathrm{e}^{\left(\frac{B}{A}\right)}\cdot F\left(x+\frac{B}{A}\right) \end{aligned} \tag{A.16}$$

Notice that the converse is also true: a time shift in $F(x)$ may be expressed as a linear combination of $F(x)$ and $G(x)$. When this result is combined with Equations (A.14)-(A.15), a linear combination of time shifted functions of $F(x)$ and $G(x)$ may be expressed as another time shifted function of $F(x)$ or $G(x)$; i.e., assume $x_o$ is a constant, with $F(x_o)$ and $G(x_o)$ being known, then:

$$\begin{aligned} AF(\delta)+BF(x_o+\delta) &= AF(\delta)+B\left[G(x_o)F(\delta)+F(x_o)G(\delta)\right] \\ &= \left[A+BG(x_o)\right]F(\delta)+BF(x_o)G(\delta) \\ &= \left[A+BG(x_o)\right]\mathrm{e}^{\left(\frac{BF(x_o)}{A+BG(x_o)}\right)}\cdot F\left(\delta+\frac{BF(x_o)}{A+BG(x_o)}\right) \end{aligned} \tag{A.17}$$

$$AG(\delta)+BG(x_o+\delta) = AG(\delta)+BG(x_o)G(\delta) = \left[A+BG(x_o)\right]G(\delta) \tag{A.18}$$

$$\begin{aligned} AF(\delta)+BG(x_o+\delta) &= AF(\delta)+BG(x_o)G(\delta) \\ &= A\mathrm{e}^{\left(\frac{BG(x_o)}{A}\right)}\cdot F\left(\delta+\frac{BG(x_o)}{A}\right) \end{aligned} \tag{A.19}$$

$$\begin{aligned} AG(\delta)+BF(x_o+\delta) &= AG(\delta)+B\left[G(x_o)F(\delta)+F(x_o)G(\delta)\right] \\ &= BG(x_o)F(\delta)+\left[A+BF(x_o)\right]G(\delta) \\ &= BG(x_o)\mathrm{e}^{\left(\frac{A+BF(x_o)}{BG(x_o)}\right)}\cdot F\left(\delta+\frac{A+BF(x_o)}{BG(x_o)}\right) \end{aligned} \tag{A.20}$$

These equations imply that any function, such as $U(t)$ or $\mu(t)$ discussed next, which is a linear combination of the time shifted function of $F(x)$ or $G(x)$, can itself be represented by a time shifted version of $F(x)$ or $G(x)$.

# Appendix B Properties of $U(t)$ and $\mu(t)$

The defining equations for $U(t)$ and $\mu(t)$ are:

$$\begin{aligned} U(t) &\equiv \sum_{t^k \leq t} f\left(\frac{t-t^k}{\tau}\right) = \sum_{t^k \leq t} \left(\frac{t-t^k}{\tau}\right) \mathrm{e}^{-\left(\frac{t-t^k}{\tau}\right)} && \text{(B.1)} \\ \mu(t) &\equiv \sum_{t^k \leq t} g\left(\frac{t-t^k}{\tau}\right) = \sum_{t^k \leq t} \mathrm{e}^{-\left(\frac{t-t^k}{\tau}\right)} && \text{(B.2)} \end{aligned}$$

The derivatives ($n > 0$), and integrals ($n < 0$), of $U(t)$ and $\mu(t)$ are given by:

$$\begin{aligned} U^{(n)}(t) &\equiv \frac{\partial^n}{\partial t^n} U(t) = \sum_{t^k \leq t} \frac{\partial^n}{\partial t^n} f\left(\frac{t-t^k}{\tau}\right) = \left(\frac{-1}{\mathrm{e}\tau}\right)^n \sum_{t^k \leq t} f\left(\frac{t-t^k-n\tau}{\tau}\right) \\ &= \left(\frac{-1}{\tau}\right)^n \sum_{t^k \leq t} \left(\frac{t-t^k}{\tau} - n\right) \mathrm{e}^{-\left(\frac{t-t^k}{\tau}\right)} = \left(\frac{-1}{\tau}\right)^n [U(t) - n\mu(t)] \\ &= \left(\frac{-1}{\mathrm{e}\tau}\right)^n U(t-n\tau) && \text{(B.3)} \\ \mu^{(n)}(t) &\equiv \frac{\partial^n}{\partial t^n} \mu(t) = \sum_{t^k \leq t} \frac{\partial^n}{\partial t^n} g\left(\frac{t-t^k}{\tau}\right) = \left(\frac{-1}{\tau}\right)^n \sum_{t^k \leq t} g\left(\frac{t-t^k}{\tau}\right) \\ &= \left(\frac{-1}{\tau}\right)^n \mu(t) && \text{(B.4)} \end{aligned}$$

Notice from Equation (B.3) that $\mu(t)$ may be expressed as:

$$\mu(t) = U(t) + \tau \dot{U}(t) \tag{B.5}$$

Also, the derivatives and integrals of $U(t)$ and $\mu(t)$ are related according to:

$$\begin{aligned} U^{(n)}(t) + U^{(-n)}(t) &= 2\left(\frac{-1}{\tau}\right)^n U(t) && \text{(B.6)} \\ \mu^{(n)}(t) + \mu^{(-n)}(t) &= 2\left(\frac{-1}{\tau}\right)^n \mu(t) && \text{(B.7)} \end{aligned}$$

Similar to $F(x)$ and $G(x)$, the time evolution of $U(t)$ and $\mu(t)$ can be expressed as:

$$\begin{aligned}
U(t) &= U(t_o+\delta) = \sum_{t^k \le t} f\left(\frac{t_o+\delta-t^k}{\tau}\right) = \sum_{t^k \le t}\left(\frac{t_o+\delta-t^k}{\tau}\right) \mathrm{e}^{-\left(\frac{t_o+\delta-t^k}{\tau}\right)} \\
&= \mathrm{e}^{-\left(\frac{\delta}{\tau}\right)}\left[\sum_{t^k \le t}\left(\frac{t_o-t^k}{\tau}\right)\mathrm{e}^{-\left(\frac{t_o-t^k}{\tau}\right)} + \left(\frac{\delta}{\tau}\right)\sum_{t^k \le t}\mathrm{e}^{-\left(\frac{t_o-t^k}{\tau}\right)}\right] \\
&= \mathrm{e}^{-\left(\frac{\delta}{\tau}\right)}\left[U(t_o) + \left(\frac{\delta}{\tau}\right)\mu(t_o)\right] = G\left(\frac{\delta}{\tau}\right)U(t_o) + F\left(\frac{\delta}{\tau}\right)\mu(t_o) \qquad \text{(B.8)} \\
\mu(t) &= \mu(t_o+\delta) = \sum_{t^k \le t} g\left(\frac{t_o+\delta-t^k}{\tau}\right) = \sum_{t^k \le t}\mathrm{e}^{-\left(\frac{t_o+\delta-t^k}{\tau}\right)} \\
&= \mathrm{e}^{-\left(\frac{\delta}{\tau}\right)}\left[\sum_{t^k \le t}\mathrm{e}^{-\left(\frac{t_o-t^k}{\tau}\right)}\right] = \mu(t_o)\cdot G\left(\frac{\delta}{\tau}\right) \qquad \text{(B.9)}
\end{aligned}$$

These equations for $U(t)$ and $\mu(t)$ are valid if no additional spikes arrive between $t_o$ and $t_o+\delta$. If a spike arrives during this time, the number of terms in the summation changes and the equations must be separated into different regions.

Using Equation (A.16), the expression for $U(t)$ in Equation (B.8) can be simplified to:

$$U(t) = \mu(t_o)\mathrm{e}^{\left(\frac{U(t_o)}{\mu(t_o)}\right)}\cdot F\left(\frac{\delta}{\tau} + \frac{U(t_o)}{\mu(t_o)}\right) \qquad \text{(B.10)}$$

This equation implies that the sum of an arbitrary number of $f(-)$ functions can always be expressed in terms of a single $F(-)$ function. The coefficient of $\mu(t_o)\mathrm{e}^{\left(\frac{U(t_o)}{\mu(t_o)}\right)}$ is always greater than or equal to 1. (Equality occurs when there is only one term within the summation.)

# Appendix C Function of $\psi(x)$ & $\kappa(x)$

It is useful to define some additional functions that are related to $f(x)$ and appear in the analysis of the SNM. Figure C.1 shows $f(x)$ along with two distinct points on the curve having the same value. These two points, $a$ and $(a+\Delta)$, may be expressed as a function of the distance between them, $\Delta$:

$$f(a) = a\mathrm{e}^{-a} = (a+\Delta)\,\mathrm{e}^{-(a+\Delta)} = f(a+\Delta) \tag{C.1}$$

$$a = \psi_L(\Delta) \equiv \frac{\Delta\mathrm{e}^{-\Delta}}{1-\mathrm{e}^{-\Delta}} = \frac{\Delta\mathrm{e}^{-\Delta/2}}{\mathrm{e}^{\Delta/2}-\mathrm{e}^{-\Delta/2}} = \left(\frac{\Delta}{2}\right)\cdot\operatorname{csch}\left(\frac{\Delta}{2}\right)\cdot\mathrm{e}^{-\Delta/2} \tag{C.2}$$

$$a+\Delta = \psi_R(\Delta) \equiv \frac{\Delta}{1-\mathrm{e}^{-\Delta}} = \frac{\Delta\mathrm{e}^{\Delta/2}}{\mathrm{e}^{\Delta/2}-\mathrm{e}^{-\Delta/2}} = \left(\frac{\Delta}{2}\right)\cdot\operatorname{csch}\left(\frac{\Delta}{2}\right)\cdot\mathrm{e}^{\Delta/2} \tag{C.3}$$

The subscripts of "L" and "R" refer to the left and right points respectively. Also, the average of these endpoints, $a$ and $(a+\Delta)$, is given by:

$$\langle\psi_L(\Delta)+\psi_R(\Delta)\rangle = a+\left(\frac{\Delta}{2}\right) = \left(\frac{\Delta}{2}\right)\cdot\left(\frac{\mathrm{e}^{\Delta/2}+\mathrm{e}^{-\Delta/2}}{\mathrm{e}^{\Delta/2}-\mathrm{e}^{-\Delta/2}}\right) = \left(\frac{\Delta}{2}\right)\cdot\coth\left(\frac{\Delta}{2}\right) \tag{C.4}$$

When negative values are allowed for $\Delta$, the functions of $\psi_L$ and $\psi_R$ reverse roles, i.e., $\psi_L(-\Delta) = \psi_R(\Delta)$. Therefore, it is really only necessary to define a single function, $\psi$,[1] to describe both the left and right points; i.e.:

$$\psi(x) \equiv \frac{x\mathrm{e}^{-x}}{1-\mathrm{e}^{-x}} \tag{C.5}$$

[1]The $\psi(x)$ function is also the Bernoulli generating function; i.e.:

$$\psi(x) = \frac{x\mathrm{e}^{-x}}{1-\mathrm{e}^{-x}} = \frac{x}{\mathrm{e}^{x}-1} = \sum_{n=0}^{\infty} B_n\frac{x^n}{n!} \qquad \text{where:} \quad |x| < 2\pi$$

The $B_n$ coefficients are called "Bernoulli numbers," where $B_0 = 1$, $B_1 = -\frac{1}{2}$, $B_2 = \frac{1}{6}$, .... See [29, Section 9.61].

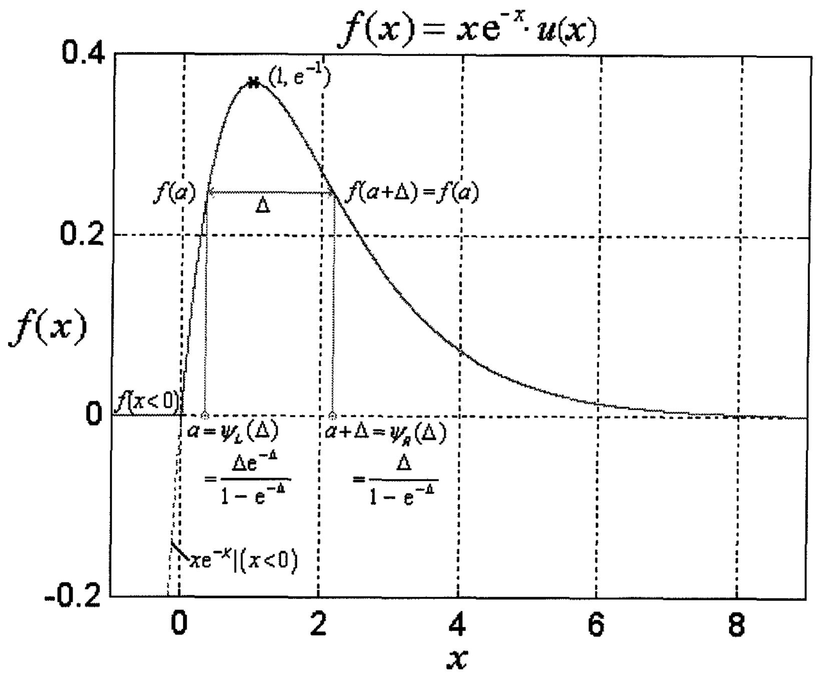


Figure C.1: **Plot of** $f(x)$. If $0 < y \leq e^{-1}$, then there must exist two points, $a$ and $(a+\Delta)$, where $0 < a \leq 1$ and $1 \leq a+\Delta$, such that $f(a) = f(a+\Delta) = y$.

$$\text{where:} \qquad \psi_L(x) = \psi(x) \qquad \text{and} \qquad \psi_R(x) = \psi(-x) \qquad \text{for} \quad x \geq 0 \tag{C.6}$$

A plot of $\psi(x)$ is given in Figure C.2. Notice:

$$\psi(-x) - \psi(x) = x \tag{C.7}$$

$$\lim_{x \to 0} \{\psi(x)\} = 1 \tag{C.8}$$

$$\lim_{x \to \infty} \{\psi(x)\} = 0 \tag{C.9}$$

$$\lim_{x \to -\infty} \{\psi(x)\} = \lim_{x \to -\infty} \{-x\} = \infty \tag{C.10}$$

$$\int_0^\infty \psi(x)\, dx = \int_0^\infty \frac{xe^{-x}}{1-e^{-x}} dx = \int_0^\infty \frac{x}{e^x - 1} dx = \frac{\pi^2}{6} \tag{C.11}$$

$$\psi(x) \approx \begin{cases} e^{-\left(\frac{6x}{\pi^2}\right)} & \text{if } x \geq 0 \\ -x + e^{\left(\frac{6x}{\pi^2}\right)} & \text{if } x < 0 \end{cases} \tag{C.12}$$

Another frequently occurring function is $\kappa(\Delta)$, which is the integral of $f(x)$ be-

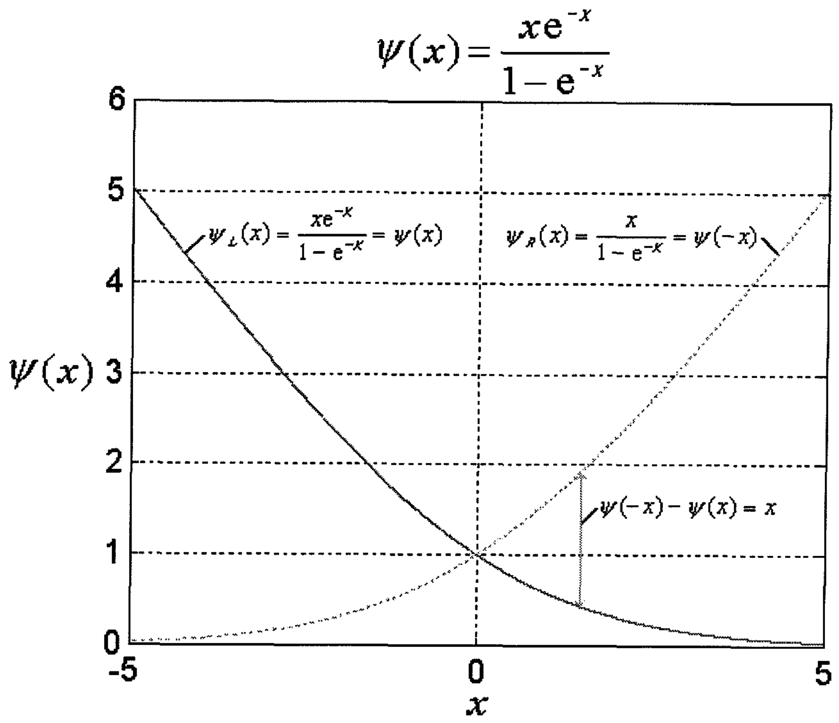


Figure C.2: **Plot of $\psi(x)$.** The function only assumes positive values.

tween $a$ and $(a+\Delta)$ in Figure C.1. The expression for $\kappa(\Delta)$ is given by:

$$\int_a^{a+\Delta} f(x)dx = \int_{\psi(\Delta)}^{\psi(-\Delta)} xe^{-x}dx = \left(1 - e^{-\Delta}\right)\cdot e^{-\psi(\Delta)} \equiv \kappa(\Delta) \tag{C.13}$$

A plot of $\kappa(x)$ is given in Figure C.3. Notice:

$$\kappa(-x) = -\kappa(x) \tag{C.14}$$

$$\lim_{x\to-\infty}\{\kappa(x)\} = -1 \tag{C.15}$$

$$\lim_{x\to\infty}\{\kappa(x)\} = 1 \tag{C.16}$$

$$\kappa(x) \approx \tanh\left(e^{-1}x\right) \tag{C.17}$$

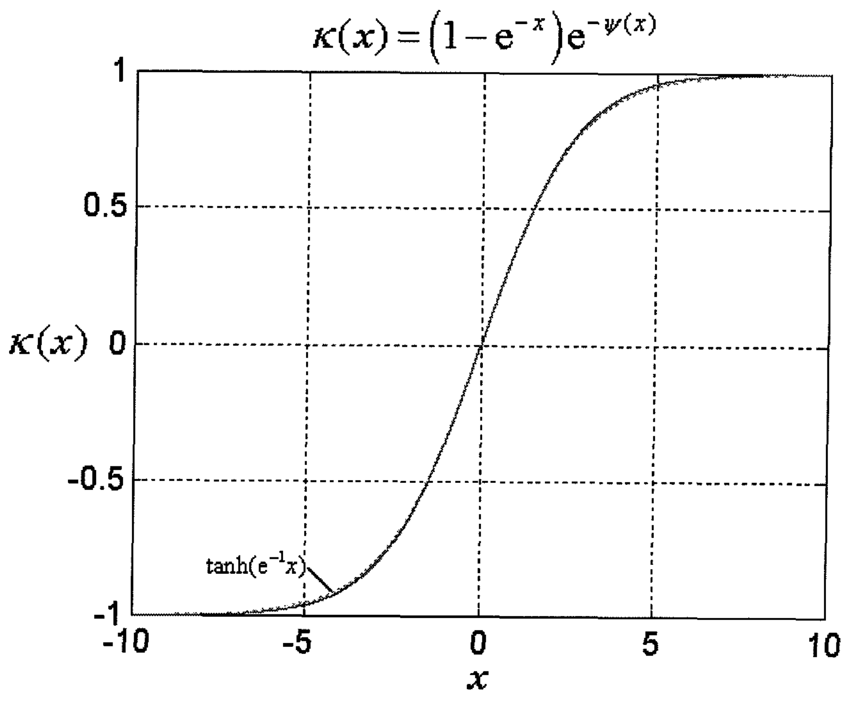


Figure C.3: **Plot of $\kappa(x)$.** The function can be well approximated by $\tanh(e^{-1}x)$.

# Appendix D Inverse Functions

When using the functions defined in the previous sections, it is sometimes necessary to calculate their inverses. This section shows how the inverses for $F(x)$, $\psi(x)$, and $\kappa(x)$ may be calculated. (The inverse for $G(x)$ is $-\ln(x)$.) The inverting algorithms where derived by expanding each of the functions as polynomials in $x$, dropping all powers higher than $x^2$, and solving for the quadratic roots. Higher order corrections were included to expand the regions and improve the rates of convergence. The details of the derivations are omitted for brevity, and only the final algorithms are presented.

Figure D.1 shows the inverse function of $F(x)$. Notice that the inverse is only defined for values less than $e^{-1}$. For values of $F(x)$ between 0 and $e^{-1}$, there are two inverse values: one is less than 1 and the other is greater than 1. For negative values of $F(x)$, there is only one inverse value.[1]

The inverse of $F(x)$ can be calculated iteratively using the following algorithm:

$$\text{if:} \qquad y = F(\alpha) = \alpha e^{-\alpha} \tag{D.1}$$

$$\text{then:} \qquad \lim_{n\to\infty} x_n = \alpha \tag{D.2}$$

where:

$$x_{n+1} = x_n + \left(\frac{y - e^{-x_n}}{y}\right)\cdot\left[-1 + \operatorname{sgn}(s_n)\cdot\sqrt{|s_n|}\right] \tag{D.3}$$

$$s_n = 1 - 2y\left(\frac{y - x_n e^{-x_n}}{(y - e^{-x_n})^2}\right) \tag{D.4}$$

[1]When using $F^{-1}(y)$ the solution can be written as: $[a, b] = F^{-1}(y)$, where $a$ is the smaller of the two solutions ($a \leq 1$). If $y$ is negative, then $b$ is undefined. When only the smaller solution is required, the $b$ term is omitted and the equation is written as: $a = F^{-1}(y)$.

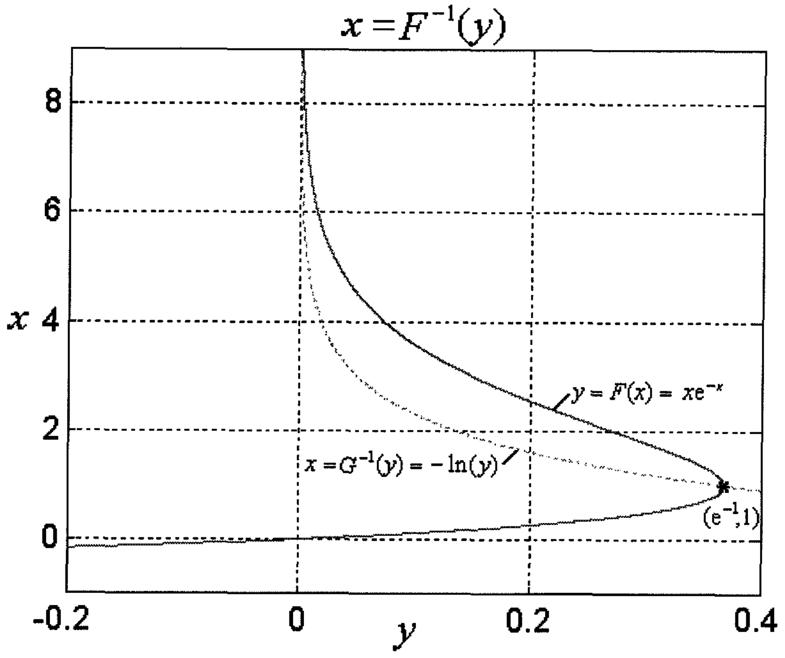


Figure D.1: **Plot of** $F^{-1}(y)$**.** The inverse function of $F(x)$ has two solutions for values between 0 and $\mathrm{e}^{-1}$. One solution is less than 1, while the other is greater than 1. The plot also shows a dotted curve for the inverse function of $G(x)$, which is $-\ln(y)$.

$$x_0 = \begin{cases} \ln\left(\frac{-\ln(-y)}{y}\right) & \text{if } y < -\mathrm{e} \quad (\alpha < -1) \\ 1 - \sqrt{1 - 3y\mathrm{e}^{-1}} & \text{if } -\mathrm{e} \le y < 0 \quad (-1 \le \alpha < 0) \\ 1 - \sqrt{1 - y\mathrm{e}} & \text{if } 0 \le y < \mathrm{e}^{-1} \quad (0 \le \alpha < 1) \\ \ln\left(\frac{-\ln(y)}{y}\right) & \text{if } 0 < y \le \mathrm{e}^{-1} \quad (1 \le \alpha) \end{cases} \tag{D.5}$$

Notice that if $0 < y < \mathrm{e}^{-1}$, then there are two possible choices for $x_0$. The smaller solution ($0 \le \alpha < 1$), can be thought of as the "left" solution in Figure C.1, while the larger solution ($1 \le \alpha$), is the "right" solution. These initial choices for $x_0$ give a close approximation to $\alpha$, but if $y > 0$, then any $x_0 < -\ln(y)$ will converge to the smaller value and any $x_0 > -\ln(y)$ will converge to the larger value. If $y < 0$, then all choices for $x_0$ converge to the only solution. With the initial choices suggested in Equation (D.5), the algorithm converged in 3 iterations to within 1x$10^{-12}$ of any $\alpha$ value between -350 and 350 ($-3.5248\text{x}10^{154} < y < 3.4754\text{x}10^{-150}$).[2] In Equation

[2]The algorithm was tested on a PC running MATLAB. First $\alpha$ was chosen and used to calculate $y$. Then the inverting algorithm was implemented and the result from the third iteration, $x_3$, was compared with $\alpha$. For $\alpha < -350$ ($y = -3.5248\text{x}10^{154}$) or $\alpha > 350$ ($y = 3.4754\text{x}10^{-150}$), rounding

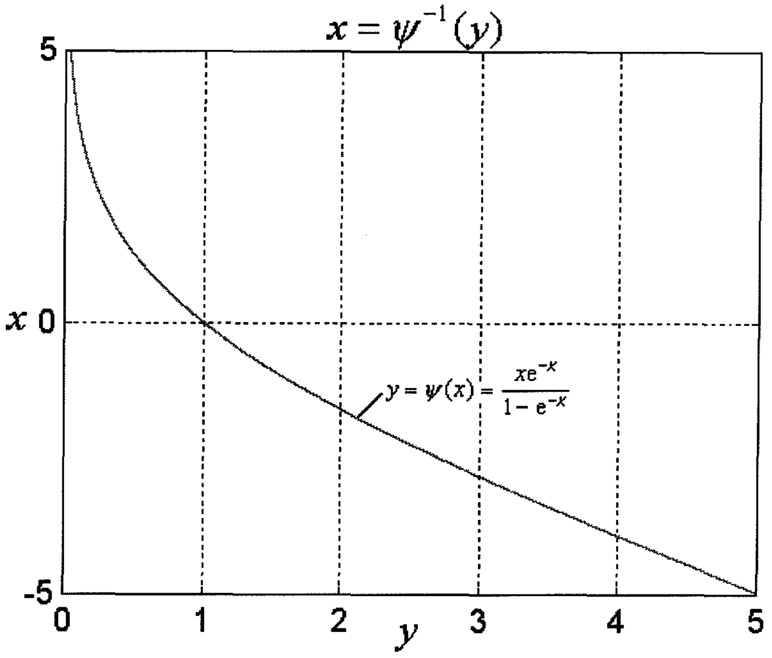


Figure D.2: **Plot of $\psi^{-1}(y)$.** The inverse function of $\psi(x)$ is defined for all positive values, and has only one unique solution.

(D.3), sgn$(x)$ function is as previously defined in Equation (4.40).

Figure D.2 shows the inverse of $\psi(x)$, which can be calculated iteratively using the following algorithm:

$$\text{if:} \quad y = \psi(\alpha) = \frac{\alpha e^{-\alpha}}{1 - e^{-\alpha}} \tag{D.6}$$

$$\text{then:} \quad \lim_{n \to \infty} x_n = \alpha \tag{D.7}$$

where:

$$x_{n+1} = x_n + \left(\frac{y - e^{-x_n}}{y}\right) \cdot \left[-1 + \operatorname{sgn}(s_n) \cdot \sqrt{|s_n|}\right] \tag{D.8}$$

$$s_n = 1 - 2y\left(\frac{y - x_n e^{-x_n} - ye^{-x_n}}{(y - e^{-x_n})^2}\right) \tag{D.9}$$

$$x_0 = \begin{cases} -\ln(y) + \sqrt{2(1 - ye)} & \text{if } y < 0.31922 \\ -y + e^{-(y-1)} & \text{if } y \geq 0.31922 \end{cases} \tag{D.10}$$

errors in $y$ dominate.

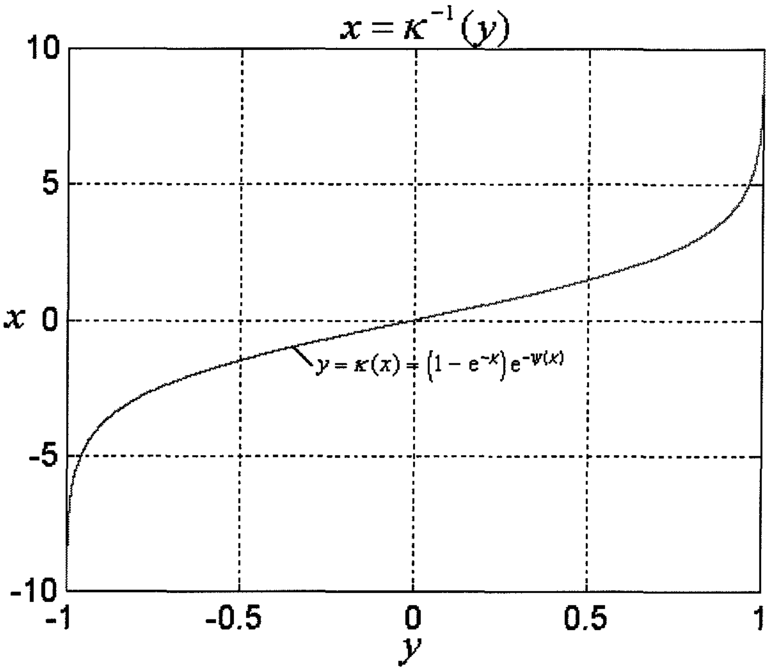


Figure D.3: **Plot of $\kappa^{-1}(y)$.** The inverse function of $\kappa(x)$ is defined for values between $-1$ and $+1$, and has only one unique solution.

Notice that these equations are identical to those used for calculating the inverse of $F(x)$ except for the extra term of $-ye^{-x_n}$ in the numerator of $s_n$, and the initial choices for $x_0$. These initial values for $x_0$ give a close approximation to $\alpha$, but any $x_0 < -\ln(y)|(y > 1)$ or $x_0 > -\ln(y)|(y < 1)$ will start converging to $\alpha$.[3] With the initial choices suggested in Equation (D.10), the algorithm converged in 8 iterations to within 1x10$^{-8}$ of any $\alpha$ value between -703 and 365 ($703 < y < 1.108682\text{x}10^{-156}$). Outside this range, rounding errors dominated.

Figure D.3 shows the inverse of $\kappa(x)$, which can be calculated iteratively using the following algorithm:

$$\text{if:} \qquad y = \kappa(\alpha) = \left(1 - e^{-\alpha}\right)e^{-\psi(\alpha)} \tag{D.11}$$

[3]There are some choices for the initial values of $x_0 > -\ln(y)|(y < 1)$ which will cause $x_{n+1}$ to go below $-\ln(y)$. The algorithm starts to converge to $\alpha$, with $|x_{n+1} - \alpha| < |x_n - \alpha|$; however, once $x_{n+1}$ falls below $-\ln(y)$, $x_\infty$ then proceeds to converge to 0. This problem only occurs for $y$ slightly below 1; e.g. if $\alpha = 0.1 \Rightarrow y = 0.95083319$ ($-\ln(y) = 0.05041663$ and $|-\ln(y) - \alpha| = 0.04958337$), then any $x_0 > 0.25830503$ results in $x_1 < 0.05041663$ and fails to converge to 0.1. The choices for $x_0$ suggested in Equation (D.10) are closer to $\alpha$ than is $-\ln(y)$ for all $y$, consequently, convergence is assured.

$$\text{then:} \qquad \lim_{n\to\infty} x_n = -\mathrm{sgn}(y)\cdot\alpha \tag{D.12}$$

where:

$$x_{n+1} = x_n + \left(\frac{1-\psi(x_n) - \mathrm{e}^{-x_n} + |y|\,\mathrm{e}^{\psi(x_n)}}{\psi(x_n) + \mathrm{e}^{-x_n} - |y|\,\mathrm{e}^{\psi(x_n)}}\right)\cdot\left[-1 + \mathrm{sgn}(s_n)\cdot\sqrt{|s_n|}\right] \tag{D.13}$$

$$s_n = -1 + 2\left(\frac{1 + (\psi(x_n) - 1)\cdot\left(\psi(x_n) + \mathrm{e}^{-x_n} - |y|\,\mathrm{e}^{\psi(x_n)}\right)}{\left(1 - \psi(x_n) - \mathrm{e}^{-x_n} + |y|\,\mathrm{e}^{\psi(x_n)}\right)^2}\right) \tag{D.14}$$

$$x_0 = -\mathrm{e}\cdot\mathrm{Atanh}\,(|y|) \tag{D.15}$$

Notice that this algorithm does not converge to $\alpha$ but to $-\mathrm{sgn}(y)\cdot\alpha$ instead. The algorithm could have been made to converge to $\alpha$ by using $-y$ in place of $|y|$ in Equations (D.13)-(D.15); however, it only converges when the numerator in Equation (D.13) is less than zero. But for $x_0 = \mathrm{e}\cdot\mathrm{Atanh}\,(y)$, this convergence constraint is not satisfied when $0.9901307956 < y < 1.0$. To solve this problem, the algorithm was modified to only deal with negative values of $y$; i.e., $y \to -|y|$, and since $\kappa(\alpha) = -\kappa(-\alpha)$, the sign of $\alpha$ is determined by the sign of $y$. (Another advantage to this modification is that the algorithm converges more rapidly for underestimates in $\alpha$ rather than over estimates, and $\mathrm{e}\cdot\mathrm{Atanh}\,(y)$ underestimates $\alpha$ for $y < 0$ and overestimates $\alpha$ for $y > 0$.) The algorithm converged in 5 iterations to within 3.5x10$^{-10}$ of any $\alpha$ value between -15 and 15 ($-0.99999510557340 < y < 0.99999510557340$).

# Appendix E Analysis of Latency

Previously, Section 4.3.2 discussed the phenomenon of latency in real neurons and qualitatively demonstrated this effect in the SNM; however, no mathematical analysis was provided. This appendix discusses how latency can be estimated in the SNM.

In Equations (3.1)-(3.4), the number of input synapses, $N_i$, can set to one, and the number of external inputs, $P_i$, can be set to zero, since only a single stimulus from another spiking neuron is being considered. Also, when analyzing latency, the neuron can be assumed to have not recently produced a spike prior to the onset of the stimulus; i.e.:

$$\tanh^2\left(\frac{t - t_i^K}{\tau_i}\right) = 1.0 \tag{E.1}$$

Therefore, the neuron will generate a spike when:

$$U_{ij}(t) = \left(\frac{R_{ij}}{T_{ij}}\right)\cdot\text{Atanh}\left(\frac{\Theta_i - R_{i0}}{R_{ij}}\right) \equiv U^{\text{spike}} \tag{E.2}$$

Figure 4.4 of Section 4.3.2, which showed the latency effect in the SNM, represents an actual simulation of a neuron with the parameters of: $\Theta_i = 0.665$; $R_{i0} = 0.0$; $R_{ij} = 1.0$; $T_{ij} = 1.0$; $\tau_{ij} = 1.0$; $d_{ij} = 0.0$. Substituting these values into Equation (E.2), indicates that the neuron will not fire until $U_{ij}(t) = 0.801725$. Notice that this value is larger than the third input stimulus of 0.75, yet the stimulus was sufficiently large for generating an output spike. This situation occurred because the time between input spikes, $\Delta$, was set so that the *time average* of $U_{ij}(t)$ would converge to 0.75, $\left(\frac{\Delta}{\tau_{ij}} = 1.333\right)$. But from Equation (4.25), the maximum value for $U_{ij}(t)$ is 0.805129, which is slightly above the minimum required to reach threshold.

Setting Equation (E.2) equal to $\max\{U_{ij}(t)\}$ in Equation (4.25), gives the maximum time between periodic input spikes which results in an output spike; i.e.:

$$\left(\frac{\Delta}{\tau_{ij}}\right)^{\max} = \kappa^{-1}\left(\frac{T_{ij}\cdot \mathrm{e}^{-1}}{R_{ij}\cdot \mathrm{Atanh}\left(\frac{\Theta_i - R_{i0}}{R_{ij}}\right)}\right) \tag{E.3}$$

This equation is plotted in Figure E.1. Also shown is a "$*$" at $\left(\frac{\Theta_i - R_{i0}}{R_{ij}}\right) = 0.665$, which corresponds to the simulation of Section 4.3.2. At this point, $\left(\frac{\Delta}{\tau_{ij}}\right)^{\max} = 1.339892$, which is the maximum time between input spikes that yields output spikes.

Since only one neuron with one input synapse is being considered, henceforth the parameter subscripts will be omitted to simplify the notation. Also, the time delay parameter, $d_{ij}$, will be assumed to be zero. A non-zero delay term can be added directly onto the neuron's latency.

The step input stimulus was simulated by using a sudden onset of periodic spikes, with the time between the pulses, $\left(\frac{\Delta}{\tau}\right)$, equaling the inverse of the desired magnitude of the stimulus strength (see Equation (4.27)). If there were no neurotransmitter in the synaptic gap prior to the start of the stimulus, then the neurotransmitter concentration in the synaptic gap after the $n^{\text{th}}$ input spike is:

$$U((n-1)\Delta \le t < n\Delta) = \sum_{k=0}^{n-1}\left(\frac{t-k\Delta}{\tau}\right)\mathrm{e}^{-\left(\frac{t-k\Delta}{\tau}\right)} \tag{E.4}$$

where $t = 0$ corresponds to the start of the input stimulus (first input spike). By letting $x \equiv t-(n-1)\Delta$ and using Equation (B.10), this equation can be written as:

$$U(n;x) \equiv \mu^n \mathrm{e}^{\left(\frac{U^n}{\mu^n}\right)}\cdot F\left(\frac{x}{\tau}+\frac{U^n}{\mu^n}\right) \tag{E.5}$$

$$\text{where:}\quad \mu^n = \sum_{k=0}^{n-1}\mathrm{e}^{-\left(\frac{k\Delta}{\tau}\right)} \quad\text{and}\quad U^n = \sum_{k=0}^{n-1}\left(\frac{k\Delta}{\tau}\right)\mathrm{e}^{-\left(\frac{k\Delta}{\tau}\right)} \tag{E.6}$$

Here, $U^n$ is the value of $U(t)$ at the arrival time of the $n^{\text{th}}$ spike.[1] With $\mu^1 = 1$, $U^1 = 0$, and $t_m^K - t_m^{K-1} = \Delta$, the values of $\mu^n$ and $U^n$ may be calculated iteratively

[1]Notice that $U^n(x)$ refers to $U(t)$ at the time of the $n^{\text{th}}$ spike, while $U^{(n)}(x)$ refers to the $n^{\text{th}}$ derivative of $U(x)$. Compare Equation (E.6) with Equation (B.3).

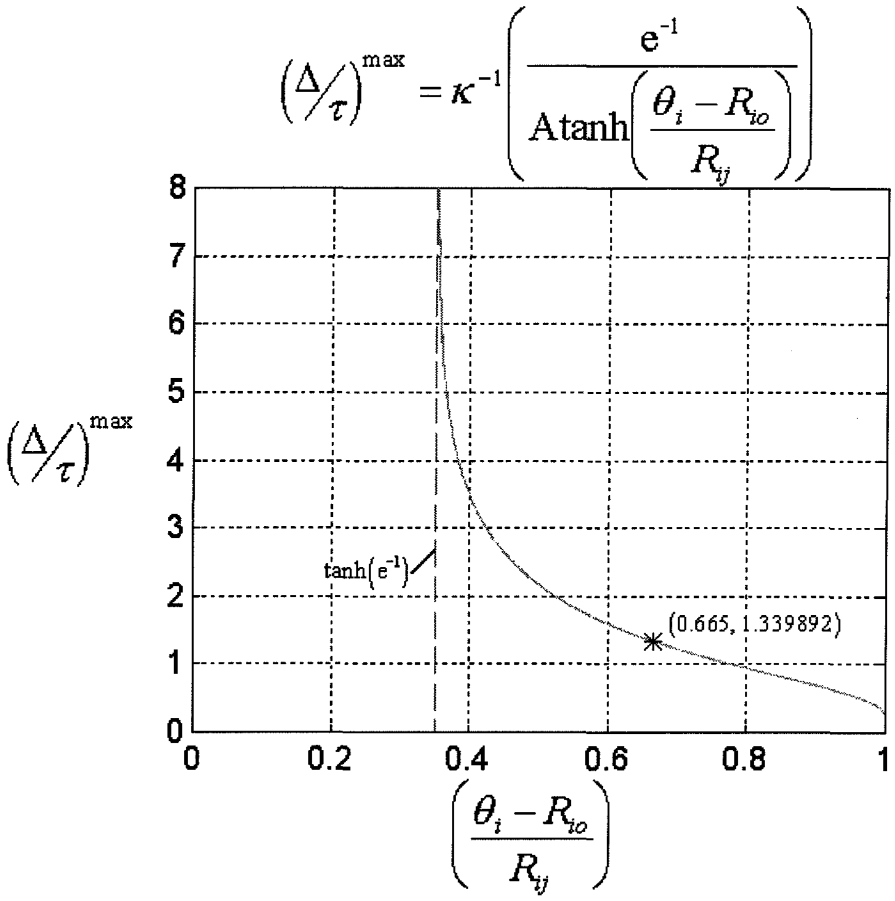


Figure E.1: **Plot of** $\left(\frac{\Delta}{\tau_{ij}}\right)^{\max}$. The plot shows the maximum time between periodic input spikes which can produce an output spike. When the time between input spikes is above the curve, no output spikes are produced. Points on the curve require an infinite number of spikes to reach the threshold. As the time between input spikes goes below the curve, fewer spikes are required to reach threshold. When $\left(\frac{\Theta_i - R_{i0}}{R_{ij}}\right)$ is below $\tanh\left(e^{-1}\right)$, only one input spike is required to produce an output spike, and when $\left(\frac{\Theta_i - R_{i0}}{R_{ij}}\right)$ is below zero, no input is required to produce a spike (see Section 4.3.4).

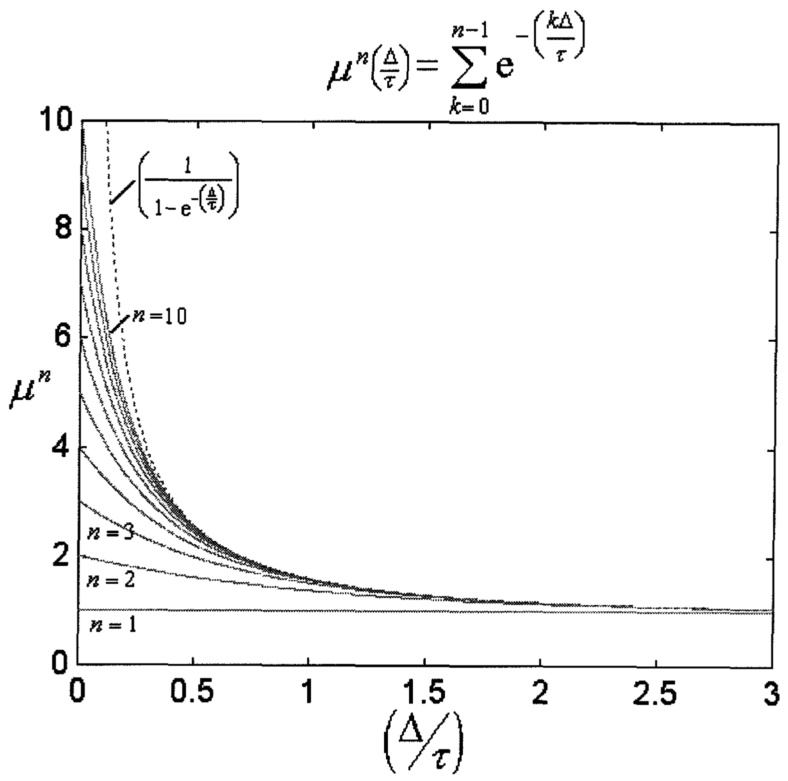


Figure E.2: **Plot of $\mu^n$.** The plot shows $\mu^n = \sum_{k=0}^{n-1} e^{-\left(\frac{k\Delta}{\tau}\right)}$ vs. $\left(\frac{\Delta}{\tau}\right)$ for the first 10 input spikes.

using Equation (4.14). Notice:

$$\lim_{n\to\infty}\{\mu^n\} = \left(\frac{1}{1-e^{-\left(\frac{\Delta}{\tau}\right)}}\right) \tag{E.7}$$

$$\lim_{n\to\infty}\{U^n\} = \left(\frac{\left(\frac{\Delta}{\tau}\right)e^{-\left(\frac{\Delta}{\tau}\right)}}{\left(1-e^{-\left(\frac{\Delta}{\tau}\right)}\right)^2}\right) = \left(\frac{\psi\left(\frac{\Delta}{\tau}\right)}{1-e^{-\left(\frac{\Delta}{\tau}\right)}}\right) \tag{E.8}$$

$$\lim_{n\to\infty}\left\{\frac{U^n}{\mu^n}\right\} = \psi\left(\frac{\Delta}{\tau}\right) \tag{E.9}$$

$$\lim_{n\to\infty}\left\{\mu^n e^{\frac{U^n}{\mu^n}}\right\} = \left(\frac{e^{\psi\left(\frac{\Delta}{\tau}\right)}}{1-e^{-\left(\frac{\Delta}{\tau}\right)}}\right) = \left(\frac{1}{\kappa\left(\frac{\Delta}{\tau}\right)}\right) \tag{E.10}$$

When these limits are substituted into Equation (E.5), it reduces to Equation (4.23), which is the result for a continuous periodic spike train. Plots of $\mu^n$ and $U^n$ are given in Figures E.2-E.3.

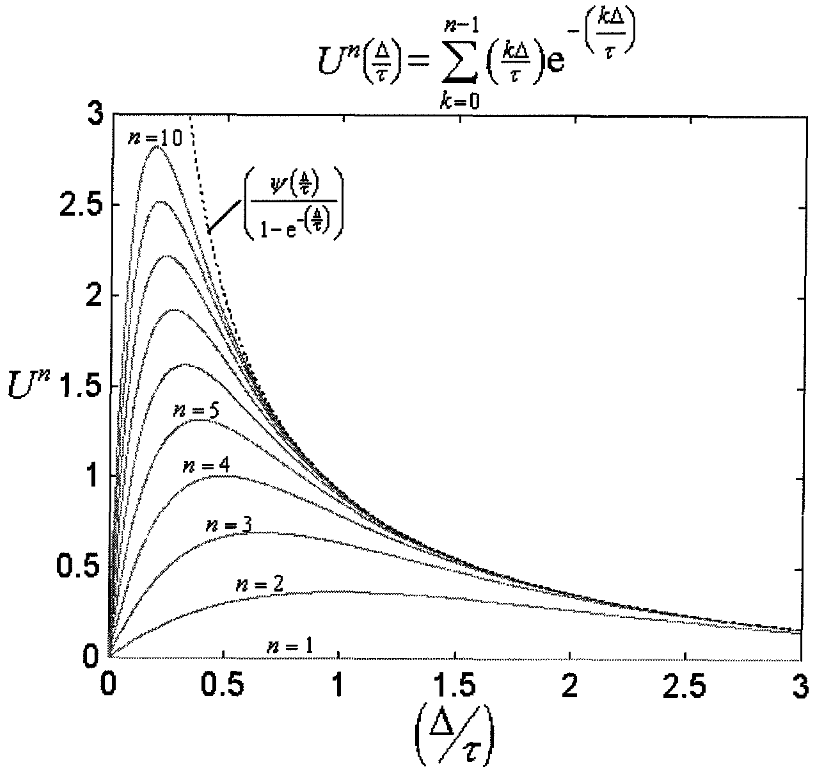


Figure E.3: **Plot of $U^n$.** The plot shows $U^n = \sum_{k=0}^{n-1}\left(\frac{k\Delta}{\tau}\right)e^{-\left(\frac{k\Delta}{\tau}\right)}$ vs. $\left(\frac{\Delta}{\tau}\right)$ for the first 10 input spikes.

Now, using Equation (A.7), the derivative of $U(n;x)$ is given by:

$$\frac{d}{dx}U(n;x) = -\left(\frac{\mu^n}{\tau}\right) \mathrm{e}^{-\left(1-\frac{U^n}{\mu^n}\right)} \cdot F\left(\frac{x}{\tau} + \frac{U^n}{\mu^n} - 1\right) \tag{E.11}$$

The maximum value for $U(n;x)$ can be found by setting this derivative to zero. However, if the time for the maximum $U(n;x)$ value is found to occur after the next input spike ($x^{\max} > \Delta$), then $U(n;x)$ must be continuously increasing between the $n^{\text{th}}$ and $(n+1)^{\text{th}}$ spike and its maximum value occurs at the time of the next spike, when $x = \Delta$. (For $x > \Delta$, the values for $\mu^n$ and $U^n$ are no longer valid, and the values for $\mu^{n+1}$ and $U^{n+1}$ must be used instead.) Thus:

$$\max\{U(n;x)\} = \begin{cases} \mu^n \mathrm{e}^{-\left(1-\frac{U^n}{\mu^n}\right)} & \text{at: } \frac{x}{\tau} = \left(1 - \frac{U^n}{\mu^n}\right) \quad \text{if: } \left(1 - \frac{U^n}{\mu^n}\right) < \left(\frac{\Delta}{\tau}\right) \\ U^{n+1} & \text{at: } \frac{x}{\tau} = \frac{\Delta}{\tau} \quad \text{if: } \left(1 - \frac{U^n}{\mu^n}\right) \geq \left(\frac{\Delta}{\tau}\right) \end{cases} \tag{E.12}$$

Figure E.4 shows the time of the maximum value for $U(n;x)$ as a function of $\frac{\Delta}{\tau}$.

As an example of an input stimulus, Figure E.4 also shows a vertical line at $\frac{\Delta}{\tau} = 0.75$. For the spike curves of $n = 1$ and $n = 2$, where the vertical line is at or above the line of $x = \Delta$, the maximum of $U(n;x)$ occurs at $\frac{x}{\tau} = 0.75$, and for points below the line of $x = \Delta$, the maximum occurs where the $n^{\text{th}}$ spike curve intersects the vertical line. At these points, $U(n;x)$ has already reached its maximum and is decreasing before the next input spike occurs. (See Figure E.6.)

Figure E.5 shows a plot of the maximum $U(n;x)$ values as a function of $\frac{\Delta}{\tau}$. Also shown is a "$*$" at $(0.75, 0.801725)$, which corresponds to the example input stimulus of $\frac{\Delta}{\tau} = 0.75$ shown in Figure E.4 and the value $U(t)$ needs to reach for the neuron to produce a spike when $U^{\text{spike}} = 0.801725$.[2] Since this point is located above the second spike curve but below the third spike curve, the neuron can not reach its threshold value until after the third input spike. The actual time of the output spike is found

[2]Notice that this example input stimulus is different than the third input stimulus of Figure 4.4, previously discussed. Both examples use the same value for $U^{\text{spike}}$, but now $\frac{\Delta}{\tau} = 0.75$, whereas before $\frac{\Delta}{\tau} = 1.333$.

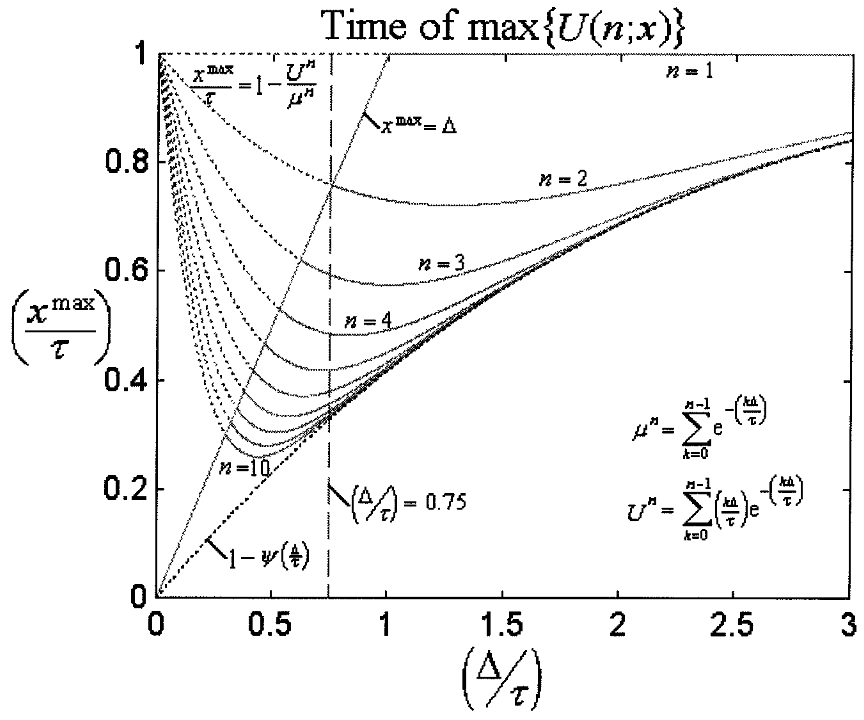


Figure E.4: **Time of** $\max\{U(n;x)\}$**.** The plot shows $\left(\frac{x^{\max}}{\tau}\right)$ vs. $\left(\frac{\Delta}{\tau}\right)$ for the first 10 input spikes. If $\left(1 - \frac{U^n}{\mu^n}\right)$ is less than $\left(\frac{\Delta}{\tau}\right)$, then the maximum value occurs before the next input spike; otherwise, the maximum occurs at the time of the next spike (when $x = \Delta$). The dotted curves above the line of $x^{\max} = \Delta$ represent when the maximum would occur in the absence of a subsequent input spike.

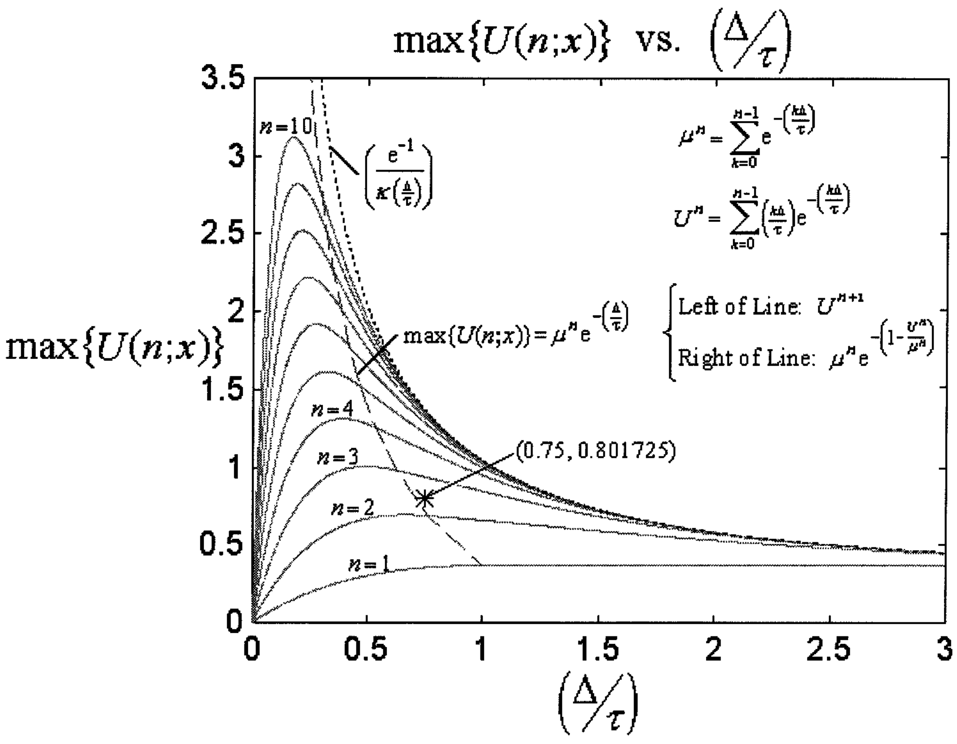


Figure E.5: **Plot of** $\max\{U(n;x)\}$**.** The plot shows the maximum values of $U(n;x)$ for the first 10 input spikes. When the time of the maximum value occurred before the next input spike, $\left(1-\frac{U^n}{\mu^n}\right)<\left(\frac{\Delta}{\tau}\right)$, then $\max\{U(n;x)\}=\mu^n \mathrm{e}^{-\left(1-\frac{U^n}{\mu^n}\right)}$, and when it was greater, $\max\{U(n;x)\}=U^{n+1}$. The boundary line between the two regions for the different equations of $\max\{U(n;x)\}$, which is given by $\max\{U(n;x)\}=\mu^n \mathrm{e}^{-\left(\frac{\Delta}{\tau}\right)}$, is shown as a dashed line. The maximum values for $U(n;x)$ are bounded above by the maximum value for an infinite spike train, which is given in Equation (4.25) and shown as a dotted line.

by setting $U(n;x) = 0.801725$ and $\frac{\Delta}{\tau} = 0.75$ in Equations (E.5)-(E.6), and using the inverse function of $F(x)$; i.e.:

$$t = (N-1)\cdot\Delta + x \tag{E.13}$$

where:

$$N = \min\left\{n \middle| U^{\text{spike}} \leq \max\left\{U(n;x)\right\}\right\} \tag{E.14}$$

$$\frac{x}{\tau} = F^{-1}\left(\frac{U^{\text{spike}}\cdot \mathrm{e}^{-\frac{U^N}{\mu^N}}}{\mu^N}\right) - \frac{U^N}{\mu^N} \tag{E.15}$$

Solving this equation with $\mu^3 = 1.695497$ and $U^3 = 0.688970$, gives $x = 0.134668$ or $t = 1.634668$. The actual plot of $U(t)$ is shown in Figure E.6. While the number of input spikes required to reach threshold was visually determined from Figure E.5, in general, the values for $\mu^n$ and $U^n$ must be iteratively calculated and substituted into Equation (E.12) to find $\max\{U(n;x)\}$. Then, when Equation (E.14) is satisfied, the appropriate values for $\mu^N$ and $U^N$ can be used to calculate $x$ and $t$ in Equations (E.15) and (E.13). Figure E.7 shows the total time and number of input spikes required to reach threshold when varying $\left(\frac{\Delta}{\tau}\right)$, but using the same threshold of $U^{\text{spike}} = 0.801725$.

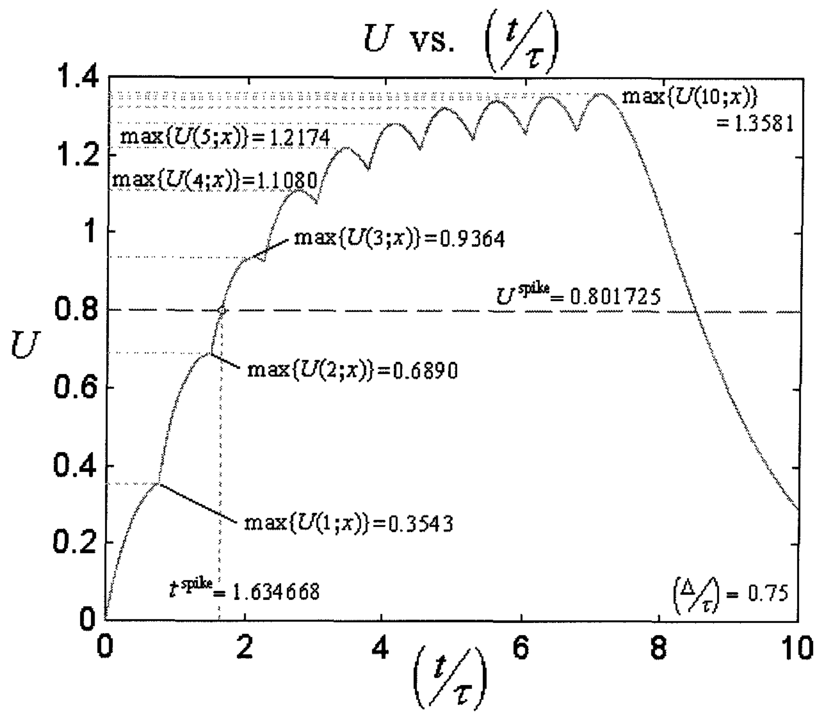


Figure E.6: **Plot of $U(t)$ for a Periodic Spike Train Stimulus.** The plot shows how $U(t)$ increases with each input spike. The period of the input spikes was $\left(\frac{\Delta}{\tau}\right) = 0.75$. The horizontal lines indicate the maximum value $U(n;x)$ attains between spikes. Notice that $U(n;x)$ is still increasing when the next spike is received for $n = 1$ and 2.

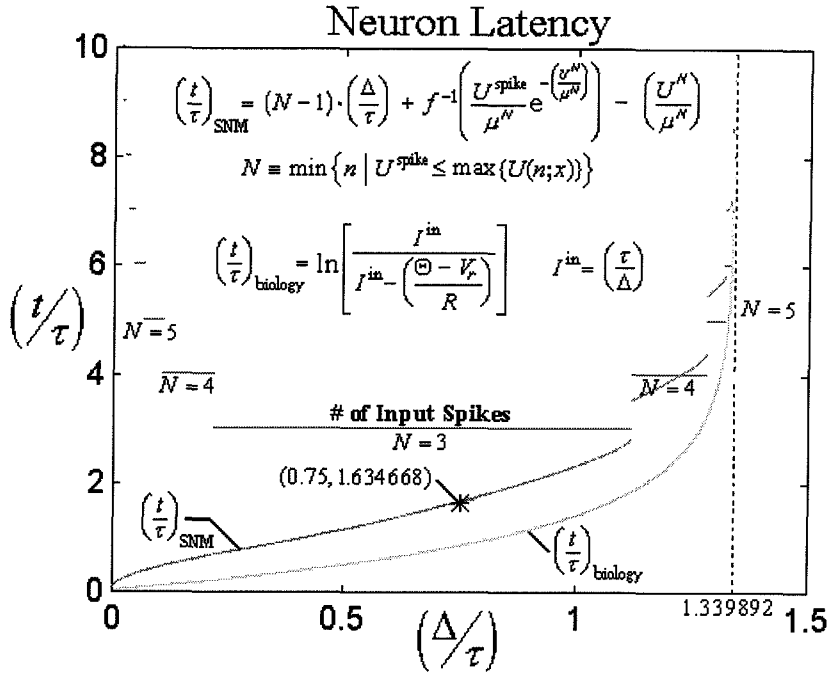


Figure E.7: **Plot of Neuron Latency.** The plot shows the amount of time and number of input spikes it takes the neuron to reach its neurotransmitter threshold level of $U^{\text{spike}} = 0.801725$. If $\left(\frac{\Delta}{\tau}\right)$ is greater than 1.339892, then the neuron never reaches threshold, and as it approaches this limit, the number of input spikes required increases to infinity (see Figure E.1). Notice that as $\left(\frac{\Delta}{\tau}\right)$ increases, the latency curve becomes discontinuous. This occurs when $U(t)$ reaches its maximum and is decreasing before the onset of the next spike. Therefore, if $\left(\frac{\Delta}{\tau}\right)$ is increased enough so that the maximum of $U(n;x)$ is just below threshold, then the neuron does not fire until after the next input spike has increased $U$ above threshold, thus causing a jump in the amount of time required. As $\left(\frac{\Delta}{\tau}\right)$ decreases toward zero, the number of input spikes required also increases, but there is no discontinuity in the latency curve since the maximum for $U(n;x)$ occurs at the start of the next input spike (see Equation (E.12)); i.e., discontinuities in the time curve only occur when there is a change in the number of input spikes required, *and* $U(t)$ is decreasing before the start of the final input pulse. For comparison, the plot also shows the estimated latency of a real neuron. The curve was generated from Equation (4.35), with $I^{\text{in}} = \frac{\tau}{\Delta}$ (see Equation (4.27) for justification), and $\left(\frac{\Theta - V_r}{R}\right) = 1.339892$. With this value for $\left(\frac{\Theta - V_r}{R}\right)$, the real neuron requires the same minimum stimulus intensity as the SNM to produce an output spike.

# Bibliography

[1] M. Abeles, H. Bergman, E. Margalit, and E. Vaadia. Spatiotemporal Firing Patterns in the Fronal Cortex of Behaving Monkeys. *Journal of Neurophysiology,* Vol. 70, 1629-1638, 1993.

[2] E. D. Adrian. *The Basis of Sensation.* London: Christophers, 1928.

[3] Michael Andrews. *Computer Organization.* Rockville, Maryland: Computer Science Press, Inc., 1987.

[4] W. Blair, C. Koch, W. Newsome, and K. Britten. Reliable Temporal Modulation in Cortical Spike Trains in the Awake Monkey. *Proceedings of the Symposium on Dynamics of Neural Processing,* Washington, D.C., 1994.

[5] Pierre Baldi. Gradient Descent Learning Algorithm Overview: A General Dynamical Systems Perspective. *IEEE Transactions on Neural Networks,* Vol. 6, No. 1, 182-195, January 1995.

[6] William H. Beyer. *CRC Standard Mathematical Tables.* 28th Edition. Boca Raton, Florida: CRC Press, Inc., 1987.

[7] W. Bialek and F. Rieke. Reliability and Information Transmission in Spiking Neurons. *Trends in Neurosciences,* Vol. 15, No. 11, 428-434, 1992.

[8] Kwabena A. Boahen. *Retinomorphic Vision Systems: Reverse Engineering the Vertebrate Retina.* Ph.D. Thesis, California Institute of Technology, Pasadena, CA 1997.

[9] J. Bower and D. Beeman. *The Book of Genesis.* New York: Springer-Verlag, 1995.

[10] T. H. Brown, A. H. Ganong, E. W. Kairiss, C. L. Keenan, and S. R. Kelso. Long-term Potentiation in Two Synaptic Systems of the Hippocampal Brain Slice. In *Neural Models of Plasticity,* pp. 266-306. Edited by J. H. Byrne and W. O. Berry. San Diego, CA: Academic Press, 1989.

[11] T. H. Brown, A. M. Zador, Z. F. Mainen, and B. J. Claiborne. Hebbian Modifications in Hippocampal Neurons. In *Long-term Potentiation,* pp. 357-389. Edited by M. Baudry and J. L. Davis. Cambridge, Massachusetts: MIT Press, 1991.

[12] A. E. Bryson and Y. C. Ho. *Applied Optimal Control.* New York: Blaisdell, 1969.

[13] G. Bugmann. Summation and Multiplication: Two Distinct Operation Domains of Leaky Integrate-and-Fire Neurons. *Network: Computation in Neural Systems,* Vol. 2, 489-509, 1991.

[14] G. A. Carpeter and S. Grossberg. ART 2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns. *Applied Optics,* Vol. 26, No. 23, 4919-4930, December 1, 1987.

[15] G. A. Carpenter and S. Grossberg. Associative Learning, Adaptive Pattern Recognition, and Cooperative-Competitive Decision Making by Neural Networks. In *Optical and Hybrid Computing,* pp. 218-247. Edited by H. Szu. SPIE Institute Series, published as *SPIE Proc.*, Vol. 634. Bellingham, WA: 1986.

[16] T. N. Cornsweet. *Visual Perception.* New York: Academic Press, Inc., 1970.

[17] Judith Dayhoff. *Neural Network Architectures.* New York: Van Nostrand Reinhold, 1990.

[18] C. Diorio, S. Mahajan, P. Hasler, B. Minch, and C. Mead. A High-Resolution Non-Volatile Analog Memory Cell. *Proceedings of the IEEE International Symposium on Circuits and Systems,* Seattle, WA, Vol. 3, 2233-2236, 1995.

[19] D. A. Durfee and F. S. Shoucair. Comparison of Floating Gate Neural Network Memory Cells in Standard VLSI CMOS Technology. *IEEE Transactions on Neural Networks,* Vol. 3, No. 3, 347-353, May 1992.

[20] John P. Dworetzky. *Psychology.* St. Paul, Minnesota: West Publishing, 1982.

[21] R. Eckhorn, O. J. Grusser, J. Kroller, K. Pellnitz, and B. Popel. Efficiency of Different Neural Codes: Information Transfer Calculations for Three Different Neural Systems. *Biological Cybernetics,* Vol. 22, 49-60, 1976.

[22] Eberhard E. Fetz. Temporal Coding in Neural Populations? *Science,* Vol. 278, 1901-1902, December 12, 1997.

[23] R. FitzHugh. Impulses and Physiological States in Models of Nerve Membrane. *Biophysical Journal,* Vol. 1, 445-466, 1961.

[24] A. Gelperin, J. J. Hopfield, and D. W. Tank. The Logic of *Limax* Learning. In *Model Neural Networks and Behavior,* pp. 237-261. Edited by Allen I. Selverston. New York: Plenum Press, 1985.

[25] W. Gerstner and J. L. van Hemmen. Associative Memory in a Network of 'Spiking' Neurons. *Network,* Vol. 3, 139-164, 1992.

[26] W. Gerstner and J. L. van Hemmen. How to Describe Neuronal Activity: Spikes, Rates, or Assemblies? *Advances in Neural Information Processing Systems,* Vol. 6, 463-470, 1994.

[27] W. Gerstner, R. Ritz, and J. L. van Hemmen. Why Spikes? Hebbian Learning and Retrieval of Time-Resolved Excitation Patterns. *Biological Cybernetics,* Vol. 69, 503-515, 1993.

[28] Scott F. Gilbert. *Developmental Biology.* Third Edition. Sunderland, Massachusetts: Sinauer Associates Inc., 1991.

[29] I. S. Gradshteyn and I. M. Ryzhik. *Tables of Integrals, Series, and Products.* San Diego: Academic Press, Inc., 1980.

[30] C. M. Gray and D. A. McCormick. Chattering Cells: Superficial Pyramidal Neurons Contributing to the Generation of Synchonous Oscillations in the Visual Cortex. *Science,* Vol. 274, 109-113, October 4, 1996.

[31] Stephen Grossberg. Classical and Instrumental Learning by Neural Networks. *Progress in Theoretical Biology,* Vol. 3, 51-141, 1974.

[32] Stephen Grossberg. Embedding Fields: A Theory of Learning with Physiological Implications. *Journal of Mathematical Psychology,* Vol. 6, 209-239, 1969.

[33] Stephen Grossberg. *Neural Networks and Natural Intelligence.* A Bradford Book. Cambridge, Massachusetts: MIT Press, 1988.

[34] Stephen Grossberg. *Studies of Mind and Brain.* Boston, Massachusetts: D. Reidel Publishing, 1982.

[35] Robert L. Harvey. *Neural Network Principles.* Englewood Cliffs, New Jersey: Prentice Hall, 1994.

[36] Paul Hasler. *Foundations of Learning in Analog VLSI.* Ph.D. Thesis, Department of Computation and Neural Systems, California Institute of Technology, Pasadena, CA 1997.

[37] Donald O. Hebb *The Organization of Behavior.* New York: John Wiley & Sons, 1949.

[38] Robert Hecht-Nielsen. Counterpropagation Networks. *Applied Optics.* Vol. 26, No. 23, 4979-4984, December 1, 1987.

[39] Robert Hecht-Nielsen. *Neurocomputing.* Reading, Massachusetts: Addison-Wesley Publishing, 1990.

[40] Robert Hecht-Nielsen. Theory of the Backpropagation Neural Network. *Proceedings of the International Joint Conference on Neural Networks,* I. 593-611, New York: IEEE Press, June 1989.

[41] A. Hertz, B. Sulzer, R. Kuhn, and J. L. van Hemmen. Hebbian Learning Reconsidered: Representation of Static and Dynamic Objects in Associative Neural Nets. *Biological Cybernetics,* Vol. 60, 457-467, 1989.

[42] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation.* Redwood City, California: Addison-Wesley Publishing, 1991.

[43] A. L. Hodgkin and A. F. Huxley. A Quantitative Description of Membrane Current and Its Application to Conduction and Excitation. *Journal of Physiology, (London),* Vol. 117, 500-544, 1952.

[44] A. L. Hodgkin and W. A. H. Rushton. The Electrical Constants of a Crustacean Nerve Fibre. *Proc. of the Royal Society London, B,* Vol. 133, 444-479, 1946.

[45] John J. Hopfield. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences,* Vol. 79, 2554-2558, April 1982.

[46] John J. Hopfield. Neurons with Graded Responses Have Collective Computational Properties Like Those of Two-State Neurons. *Proceedings of the National Academy of Sciences,* Vol. 81, 3088-3092, May 1984.

[47] John J. Hopfield. Pattern Recognition Computation Using Action Potential Timing for Stimulus Representation. *Nature,* Vol. 376, 33-36, July 6, 1995.

[48] John J. Hopfield. Transforming Neural Computations and Representating Time. *Proceedings of the National Academy of Sciences,* Vol. 93, 15440-15444, December 1996.

[49] Tim Horiuchi, John Lazzaro, Andrew Moore, and Christof Koch. A Delay-Line Based Motion Detection Chip. *Advances in Neural Information Processing Systems,* Vol. 3, 406-412, 1991.

[50] D. Johnston and S. M. Wu. *Foundations of Cellular Neurophysiology.* Cambridge, Massachusetts: MIT Press, 1995.

[51] J. J. B. Jack, D. Nobel, and R. W. Tsien. *Electric Current Flow in Excitable Cells.* Oxford: Clarendon Press, 1975.

[52] Eric R. Kandel. *Cellular Basis of Behavior.* San Francisco, California: W. H. Freeman and Company, 1976.

[53] E. R. Kandel, J. H. Schwartz, and T. M. Jessel. *Principles of Neural Science.* Third Edition. Norwalk, Connecticut: Appleton & Lange, 1991.

[54] Shlomo Karni. *Analysis of Electrical Networks.* New York: John Wiley & Sons, 1986.

[55] S. R. Kelso, A. H. Ganong, and T. H. Brown. Hebbian Synapses in Hippocampus. *Proceedings of the National Academy of Sciences,* Vol. 83, 5326-5330, 1986.

[56] Christof Koch. Computation and the Single Neuron. *Nature,* Vol. 385, No. 6613, 207-210, January 16, 1997.

[57] Christof Koch. When Looking is Not Seeing: Towards a Neurobiological View of Awareness. *Engineering & Science,* (California Institute of Technology), Vol. LVI, No. 3, 2-13, Spring 1993.

[58] C. Koch and T. Poggio. Multiplying with Synapses and Neurons. In *Single Neuron Computation,* pp. 315-345. Edited by T. McKenna, J. Davis, and S. F. Zornetzer. San Diego, California: Harcourt Brace Jovanovich, 1992.

[59] C. Koch and I. Segev. *Methods in Neuronal Modeling.* A Bradford Book. Cambridge, Massachusetts: MIT Press, 1989.

[60] Teuvo Kohonen. *Self-Organization and Associative Memory.* Third Edition. Berlin: Springer-Verlag, 1989.

[61] S. W. Kuffler, J. G. Nicholls, and A. R. Martin. *From Neuron to Brain.* Second Edition. Sunderland, Massachusetts: Sinauer Associates Inc., 1984.

[62] I. Kupfermann and K. R. Weiss. The Command Neuron Concept. *Behavioral and Brain Sciences,* Vol. 1, 3-39, 1978.

[63] E. H. Land. The Retinex Theory of Color Vision. *Scientific American,* Vol. 237, No. 6, 108-128, 1977.

[64] Clifford Lau. *Neural Networks: Theoretical Foundations and Analysis.* New York: The Institute of Electrical and Electronics Engineers, Inc., 1992.

[65] I. Levitan and L. Kaczmarek. *The Neuron: Cell and Molecular Biology.* New York: Oxford University Press, 1991.

[66] Wolfgang Maass. Bounds for the Computational Power and Learning Complexity of Analog Neural Networks. *SIAM Journal of Computation,* Vol. 26, No. 3, 708-732, 1997.

[67] Wolfgang Maass. Fast Sigmoidal Networks via Spiking Neurons. *Neural Computation,* Vol. 9, No. 2, 279-304, 1997.

[68] Wolfgang Maass. Lower Bounds for the Computational Power of Networks of Spiking Neurons. *Neural Computation,* Vol. 8, No. 1, 1-40, 1996.

[69] Wolfgang Maass and Thomas Natschlager. Networks of Spiking Neurons can Emulate Arbitrary Hopfield Nets in Temporal Coding. *Network: Computation in Neural Systems,* Vol. 8, 355-371, 1997.

[70] H. Markram, J. Lubke, M. Frotscher, and B. Sakmann. Regulation of Synaptic Efficacy by Coincidence of Postsynaptic APs and EPSPs. *Science,* Vol. 275, No. 5297, 213-215, January 10, 1997.

[71] W. S. McCulloch and W. Pitts. A Logical Calculus of Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics,* Vol. 5, 115-133, 1943.

[72] Carver Mead. *Analog VLSI and Neural Systems.* Reading, Massachusetts: Addison-Wesley Publishing, 1989.

[73] C. Mead and M. Ismail, Ed. *Analog VLSI Implementation of Neural Systems.* Boston, Massachusetts: Kluwer Academic Publishers, 1989.

[74] John Milton. *Dynamics of Small Neural Populations.* CRM Monograph Series, Vol. 7. Providence, Rhode Island: American Mathematical Society, 1996.

[75] M. L. Minsky and S. A. Papert. *Perceptrons.* Expanded Edition, 1988. Cambridge, Massachusetts: MIT Press, 1969.

[76] Keith L. Moore. *The Developing Human.* Fourth Edition Philadelphia: W. B. Saunders Company, 1988.

[77] B. Muller and J. Reinhardt. *Neural Networks, An Introduction.* New York: Springer-Verlag, 1990.

[78] A. F. Murray. Silicon Implementations of Neural Networks. *IEE Proceedings – F,* Vol. 138, No. 1, 3-12, February 1991.

[79] J. S. Nagumo, S. Arimoto, and S. Yoshizawa. An Active Pulse Transmission Line Simulation of a Nerve Axon. *Proceedings – Institute of Radio Engineers,* Vol. 50, 2061-2070, 1962.

[80] M. M. Nelson and W. T. Illingworth. *A Practical Guide to Neural Nets.* Reading, Massachusetts: Addison-Wesley Publishing, 1991.

[81] Barak A. Pearlmutter. Learning State Space Trajectories in Recurrent Neural Networks. *Neural Computation,* 1, 263-269, 1989.

[82] Barak A. Pearlmutter. Learning State Space Trajectories in Recurrent Neural Networks. *Technical Report CMU-CS-88-191,* School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1988.

[83] Fernando J. Pineda. Generalization of Back-Propagation to Recurrent Neural Networks. *Physical Review Letters,* 59, 2229-2232, 1987.

[84] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes: The Art of Scientific Computing* (FORTRAN Version). Cambridge: Cambridge University Press, 1989.

[85] Wilfrid Rall. Branching Dendritic Trees and Motorneuron Membrane Resistivity. *Experimental Neurology,* Vol. 2, 503-532, 1959.

[86] Wilfrid Rall. Cable Theory for Dendritic Neurons. In *Methods in Neuronal Modeling,* pp. 9-62. Edited by C. Koch and I. Segev. A Bradford Book. Cambridge, Massachusetts: MIT Press, 1989.

[87] Wilfrid Rall. Theoretical Significance of Dendritic Tree for Input-Output Relation. In *Neural Theory and Modeling,* pp. 73-97. Edited by R. F. Reiss. Stanford, California: Stanford University Press, 1964.

[88] Wilfrid Rall. Time Constant and Electrotonic Length of Membrane Cylinders and Neurons. *Journal of Biophysiology,* Vol. 9, 1483-1508, 1969.

[89] F. Ratliff. *Mach Bands: Quantitative Studies on Neural Networks in the Retina.* New York: Holden-Day, 1965.

[90] A. Riehle, S. Grun, M. Diesmann, and A. Aertsen. Spike Synchronization and Rate Modulation Differentially Involved in Motor Cortical Function. *Science,* Vol. 278, 1950-1953, December 12, 1997.

[91] F. Rieke, D. Ruyter van Steveninck, and W. Bialek. *SPIKES: Exploring the Neural Code,* Cambridge, Massachusetts: MIT Press, 1996.

[92] E. T. Rolls. Brain Mechanisms for Invariant Visual Recognition and Learning. *Behavioural Processes,* Vol. 33, 113-138, 1994.

[93] E. T. Rolls, M. J. Tovee. Processing Speed in the Cerebral Cortex, and the Neurophysiology of Visual Backward Masking. *Proc. of the Royal Society London, B,* Vol. 257, 9-15, 1994.

[94] Frank Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychology Review,* Vol. 65, 386-408, 1958.

[95] Frank Rosenblatt. *Principles of Neurodynamics.* New York: Spartan, 1962.

[96] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group. *Parallel Distributed Processing.* Vol. 1 & 2. Cambridge, Massachusetts: MIT Press, 1986.

[97] Rahul Sarpeshkar. *Efficient Precise Computation with Noisy Components: Extrapolating from an Electronic Cochlea to the Brain.* Ph.D. Thesis, California Institute of Technology, Pasadena, CA 1997.

[98] Bruce Schechter. How the Brain Gets Rhythm. *Science,* Vol. 274, 339-340, October 18, 1996.

[99] Robert E. Schell, Ed. *Developmental Psychology Today.* Second Edition. New York: Random House, 1975.

[100] I. Segev, J. Fleshman, and R. Burke. Compartmental Models of Complex Neurons. In *Methods in Neuronal Modeling,* pp. 63-96. Edited by C. Koch and I. Segev. A Bradford Book. Cambridge, Massachusetts: MIT Press, 1989.

[101] Idan Segev. Single Neurone Models: Oversimple, Complex and Reduced. *Trends in Neurosciences,* Vol. 15, No. 11, 414-421, 1992.

[102] T. J. Sejnowski. Time for a New Neural Code? *Nature,* Vol. 376, 21-22, 1995.

[103] C. E. Shannon and W. Weaver. *The Mathematical Theory of Communication.* Urbana, Illinois: Univ. Illinois Press, 1949.

[104] Murray R. Spiegel. *Schaum's Outline Series: Mathematical Handbook of Formulas and Tables.* New York: McGraw-Hill Book Company, 1968.

[105] R. B. Stein. The Frequency of Nerve Action Potentials Generated by Applied Currents. *Proc. of the Royal Society London, B,* Vol. 167, 64-86, 1967.

[106] Charles F. Stevens. Strengths and Weaknesses in Memory. *Nature,* Vol. 381, No. 6582, 471-472, June 6, 1996.

[107] Doron Tal and Eric L. Schwartz. Computing with the Leaky Integrate-and-Fire Neuron: Logarithmic Computation and Multiplication. *Neural Computation,* Vol. 9, No. 2, 305-318, 1997.

[108] N. B. Toomarian and J. Barhen. Learning a Trajectory Using Adjoint Functions and Teacher Forcing. *Neural Networks,* 5, 473-484, 1992.

[109] Philip D. Wasserman. *Neural Computing Theory and Practice.* New York: Van Nostrand Reinhold, 1989.

[110] P. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* Ph.D. Thesis, Harvard University, 1974.

[111] B. Widrow and M. E. Hoff. Adaptive Switching Circuits. *Institute of Radio Engineers: WESCON Convention Record,* Part 4, 96-104, August, 1960.

[112] R. J. Williams and D. Zipser. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation,* 1, 270-280, 1989.

[113] R. K. S. Wong, D. A. Prince, and A. F. Basbaum. Intradendritic Recordings from Hippocampal Neurons. *Proceedings of the National Academy of Sciences,* Vol. 76, 986-990, 1979.

[114] Y. Yeshurun and E. L. Schwartz. Cepstral Filtering on a Columnar Image Architecture: A Fast Algorithm for Binocular Stereo Segmentation. *IEEE Transactions: Pattern Analysis and Machine Intelligence,* Vol. 11, No. 7, 759-767, 1989.