

RICE UNIVERSITY

**Efficient and Accurate Simulation of Integrate-and-Fire  
Neuronal Networks in the Hippocampus**


by

**Anthony Richard Kellems**

A THESIS SUBMITTED  
IN PARTIAL FULFILLMENT OF THE  
REQUIREMENTS FOR THE DEGREE


**Master of Arts**

APPROVED, THESIS COMMITTEE:



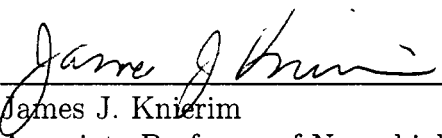
---

Steven J. Cox, Advisor  
Professor of Computational and Applied Mathematics



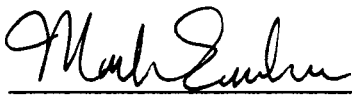
---

Danny C. Sorensen  
Noah G. Harding Professor of Computational and Applied Mathematics



---

James J. Knierim  
Associate Professor of Neurobiology and Anatomy  
University of Texas Health Science Center at Houston



---

Mark P. Embree  
Assistant Professor of Computational and Applied Mathematics

HOUSTON, TEXAS

MAY 2007

UMI Number: 1441828

### INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

**UMI<sup>®</sup>**

---

UMI Microform 1441828

Copyright 2007 by ProQuest Information and Learning Company.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company  
300 North Zeeb Road  
P.O. Box 1346  
Ann Arbor, MI 48106-1346

## **Abstract**

### Efficient and Accurate Simulation of Integrate-and-Fire Neuronal Networks in the Hippocampus

by

Anthony Richard Kellems

This thesis evaluates a method of computing highly accurate solutions for network simulations of integrate-and-fire (IAF) neurons. Simulations are typically evolved using time-stepping, but since the IAF model is composed of linear first-order ODEs with hard thresholds, explicit solutions in terms of integrals of exponentials exist and can be approximated using quadrature. The technique presented here utilizes Clenshaw–Curtis quadrature to approximate these integrals to high accuracy. It uses the secant method to more precisely identify spike times, thus yielding more accurate solutions than do time-stepping methods. Additionally, modeling synaptic input with delta functions permits the quadrature method to be practical for simulating large-scale networks. I determine general conditions under which the quadrature method is faster and more accurate than time-stepping methods. In order to make these methods accessible to other researchers, I introduce and develop software designed for simulating networks of IAF hippocampal cells.

## **Acknowledgements**

I would first like to thank the members of my committee for their guidance, especially my advisor Dr. Cox, who first introduced me to applied mathematics when I was an undergraduate, and Dr. Embree, who has been the most instrumental in shaping my applied mathematics career, and Dr. Sorensen and Dr. Knierim for their assistance in preparing me for my defense.

I also thank Dr. Chip Levy and his lab members, Ben Hocking and Andrew Howe, for their hospitality and dialogue when I visited the University of Virginia to meet with them and discuss hippocampal simulations.

Thanks also to one of my best friends, Alana Abernathy, who not only has given me support and inspiration but who is a student of cognitive science and has an appreciation for the applications of my work.

Finally, I thank my family, foremost Mom and Dad, for their unending support and love and for giving me the chances and opportunities to get to where I am now, and I thank God for the gifts and talents He has given me.

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation for Modeling Hippocampal Neurons . . . . .	1
1.2	Approaches to Neuronal Modeling . . . . .	4
<b>2</b>	<b>Methods</b>	<b>9</b>
2.1	The Integrate-and-Fire Model . . . . .	9
2.1.1	Spike Rate Adaptation . . . . .	11
2.1.2	Synaptic Input . . . . .	12
2.2	Solution Methods . . . . .	15
2.2.1	Time-stepping . . . . .	16
2.2.2	Analytic Solution Derivation . . . . .	18
2.2.3	Quadrature Methods . . . . .	21
2.2.4	Analytic Solution via Composite CC Quadrature . . . . .	22
2.2.5	Spike Time Identification . . . . .	23
2.2.6	Multi-Cell Solution via Quadrature . . . . .	25
2.3	Modeling Synapses with Delta Functions . . . . .	26
2.3.1	Modeling the EC/DG Layer . . . . .	27
2.3.2	Delta-synapse Equations . . . . .	28
2.4	Simulation Platform and Visualization Tools . . . . .	31
<b>3</b>	<b>Results</b>	<b>35</b>
3.1	Single Neuron Dynamics . . . . .	35
3.2	A Simple Function: The XOR Network . . . . .	38
3.3	Efficiency and Accuracy of the Quadrature Method . . . . .	39
3.3.1	Computational Complexity: Using Alpha Functions . . . . .	40
3.3.2	Computational Complexity: Using Delta-synapses . . . . .	44
3.3.3	Effect of STI Tolerances . . . . .	47
3.3.4	Clenshaw–Curtis Quadrature vs. Gaussian Quadrature . . . . .	48
3.4	Network Rhythmogenesis . . . . .	50
3.5	Summary of Results . . . . .	54
<b>4</b>	<b>Future Work</b>	<b>55</b>
<b>5</b>	<b>Conclusion</b>	<b>58</b>

## List of Figures

1.1	Hippocampus with main regions indicated . . . . .	2
2.1	Basic circuit diagram of a single integrate-and-fire cell . . . . .	10
2.2	A small network with synaptic connections . . . . .	13
2.3	Alpha function for a stimulated cell . . . . .	14
2.4	Full circuit diagram of the IAF model . . . . .	15
2.5	Synaptic connections in the delta-synapse model . . . . .	28
2.6	Synaptic conductance functions in the delta-synapse model . . . . .	29
2.7	The simulation GUI in MATLAB . . . . .	32
2.8	The visualization GUI in MATLAB . . . . .	33
3.1	Single neuron voltage trajectories and error curves . . . . .	37
3.2	Error curves for a single neuron with spike rate adaptation . . . . .	38
3.3	XOR firing and voltage plots . . . . .	39
3.4	Errors vs. number of quadrature nodes for alpha function model . . .	41
3.5	Quadrature method errors and timings vs. $N$ for alpha function model	43
3.6	Quadrature method errors and timings vs. $N$ for delta-synapse model	45
3.7	Errors vs. number of quadrature nodes for delta-synapse model . . .	46
3.8	Voltage errors for neurons in oscillatory network . . . . .	47
3.9	CC vs. Gaussian quadrature error plot . . . . .	48
3.10	CC vs. Gauss–Lobatto quadrature error plot . . . . .	49
3.11	Population oscillation from synchronized firing . . . . .	50
3.12	Firing rates and power spectra for synchronized firing . . . . .	51
3.13	Voltage errors for neurons in oscillatory network . . . . .	52
3.14	Timestep distributions for quadrature method simulations . . . . .	53

**List of Tables**

2.1	Integrate-and-Fire Model Parameters . . . . .	16
2.2	Delta-synapse Conductance Function Parameters . . . . .	30
3.1	Complexity of Quadrature Method Simulation: Alpha Function Model	43
3.2	Complexity of Quadrature Method Simulation: Delta-synapse Model	44

# Chapter 1

## Introduction

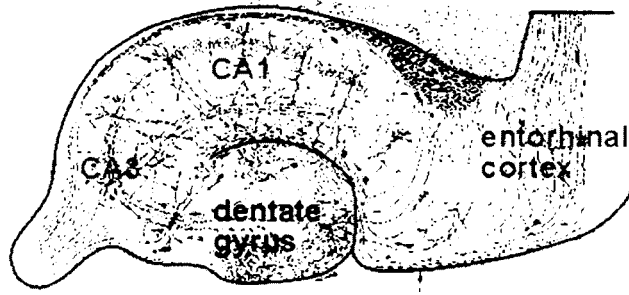
This thesis quantifies the efficiency and accuracy of quadrature methods for simulating networks of hippocampal neurons using the integrate-and-fire model. The primary questions are, how accurately can individual voltage trajectories be resolved during a network simulation, and how long does it take to resolve them? Using new software that I developed specifically for hippocampal network simulations, I am able to answer the aforementioned questions and provide a simulation platform for use with future research.

### 1.1 Motivation for Modeling Hippocampal Neurons

Neurons in the brain form an interconnected network that gives rise to cognitive functions, one of which is memory formation. This formation occurs in the hippocampus, a brain region which has been studied extensively (Traub and Miles [25], pp. 1–4). The inability to form new long-term memories is associated with damage to the hippocampus, as has been demonstrated in human case studies and verified in spatial memory experiments on rats (Johnston and Amaral [16], pp. 455–456). Not



only is the hippocampus of interest because of its connection to memory, but also because it is “the simplest type of cortex,” making it suitable for computational studies (Traub and Miles [25], p. 1). However, simplicity in relation to other brain regions does not mean that hippocampal simulations are in any way easy.



**Figure 1.1:** Hippocampus with main regions indicated (Byrne [9]). The cells that are modeled in this thesis correspond to those located in Region CA3.

The hippocampus is divided into four main regions, depicted in Figure 1.1, each responsible for a portion of the network dynamics. The entorhinal cortex (EC) sends inputs to region CA3 and the dentate gyrus (DG); region CA3 contains recurrent connections and also sends input to region CA1, which connects back to the EC (Johnston and Amaral [16], p. 424). Estimates from rat hippocampal studies indicate that about 300,000 cells comprise region CA1, 1,000,000 cells make up the DG, 200,000 cells comprise the EC, and about 200,000 cells comprise region CA3, and that the density of recurrent connections within region CA3 is estimated at 2% (Ascoli et al. [2]).

It is region CA3 which is of great interest because of this relatively high percentage

of recurrent connections. These connections appear to be responsible for generating the neuronal firing patterns characteristic of epileptic seizures (Johnston and Amaral [16], p. 457). Region CA3 also plays a role in generation of network oscillations, such as the gamma rhythm observed in rats (Bragin et al. [6], Buzsáki et al. [8]).

In large-scale simulations of hippocampal networks it is typical to model only region CA3 cells explicitly because recurrent connections are more prevalent than in other regions, which can be modeled as feedforward black boxes (e.g., see August and Levy [3], August [4]). Model neurons have a wide range of granularity, from highly realistic single-cell models to less-realistic large-scale neuronal networks. Simulating at any point on this spectrum requires sophisticated experimental techniques to obtain reliable parameter estimates (e.g., see Hodgkin and Huxley [15]) and reliable cell firing data (e.g., see Yoganarasimha et al. [30]), as well as development of models that can be accurately solved.

This thesis simulates networks of region CA3 cells by using the integrate-and-fire neuron model, which ignores the action potential generated by neuronal firings and instead captures subthreshold dynamics. This model was first proposed by Lapicque in 1907; over time it has been expanded to accommodate various cellular mechanisms, but it still has the form of a linear ODE (Abbott [1]). The ultimate question is, can realistic cognitive behaviors be reproduced by this model in a reasonable amount of time? To answer this question requires techniques for solving the model equations to be efficient and accurate. The results of this thesis quantify just how efficient and

accurate two solution techniques are when applied to IAF networks.

## 1.2 Approaches to Neuronal Modeling

The unavoidable reality of neuronal simulation is that the more realistic the model is, the slower the simulation will be, and thus a variety of models and solution techniques are used in practice. The integrate-and-fire (IAF) neuron is a simplified version of a single cell that is useful because it captures spiking behavior without much computational cost (Dayan and Abbott [12], p. 163). However, solving linear ODEs with hard thresholds to high accuracy remains a difficult task due to discontinuities caused by cell firings (Shelley and Tao [23]). Computational efforts have primarily focused on time-stepping schemes, addressing either how to improve their accuracy (Shelley and Tao [23]) or how to take advantage of their simplicity to facilitate simulations (August and Levy [3]). However, recent work from 2007 claims that, depending on the network topology, exact solution techniques provide statistically accurate solutions in a fraction of the time required for time-stepping (Rangan and Cai [21]).

A biologically realistic model requires a spatial discretization of the cell into compartments, as was done by Traub and Miles in pioneering work in the early 1990s (Traub and Miles [25]). In efforts to understand the dynamics of epileptic bursts, Traub and Miles developed a 19-compartment model, but such detailed models as this often preclude efficient large-scale network simulation because of the high computational cost. To improve simulation speed, reduced multi-compartment models

have been developed, notably that of Pinsky and Rinzel (PR) (Pinsky and Rinzel [20]). They divided the cell into a somatic and a dendritic compartment and showed that such cells exhibit neuronal firing patterns that are very similar to those of the more complex 19-compartment model of Traub and Miles.

In contrast to the IAF model, which is a single-compartment reduction, the equations that govern the PR model are continuous but coupled. No closed form solutions exist, so they are approximated using high-order Runge–Kutta (RK) time-stepping. For a desired accuracy this scheme generally permits larger timesteps than do lower order methods. It would be useful to have such a scheme available for the IAF model, but this cannot be done without losing accuracy unless modifications are made.

Discontinuities will relegate higher-order time-stepping methods to first order, but interpolation schemes can be used to regain this lost accuracy. Linear interpolation was first used to regain accuracy for second-order schemes (Hansel et al. [14]) and was later generalized for higher-order schemes (Shelley and Tao [23]). If the IAF model equations are sufficiently smooth, then spike times can be more accurately captured by constructing an interpolating polynomial through recent data points and finding where it crosses the threshold. This new spike time approximation is used to adjust the post-spike voltage (Shelley and Tao [23]).

As an example of the effectiveness of this correction, a 128-neuron network simulated using this technique in an RK 4th-order scheme can regain 4th-order accuracy for a timestep of  $0.5 \times 10^{-3}$  versus  $10^{-9}$  for a first-order (Backward Euler) scheme

(Shelley and Tao [23]). While this is very useful, it relies upon continuity assumptions that may not hold for certain IAF equations, and for highly active or strongly connected networks the timestep may need to be much smaller (Rangan and Cai [21]).

As an alternative to time-stepping, quadrature can be used to approximate the analytic solution. Traditionally this approach is avoided because of the immense computational cost associated with the quadrature routines. Rangan and Cai propose a way to reduce computational complexity by taking advantage of the connectivity of the network, paying particular attention to the visual cortex. This is a region where neurons have strong local connections but sparse connections otherwise. When a cell spikes, the spatial structure of the network is used to predict which cells are likely to spike in response to this synaptic input (Rangan and Cai [21]).

Published results claim that 260,000-neuron networks can be simulated for 128 ms in an hour on a laptop (Rangan and Cai [21]). This appears to require much more storage than any laptop could have, but because neurons are only connected to their neighbors on a two-dimensional lattice the memory requirements drop significantly, as does the computational complexity. However, such a spatial network topology has not been observed in the hippocampus, and computational resources force simulations to use far fewer cells than occur in a real hippocampus.

As a consequence, depending on the size of the network, the density of connections used in these simulations takes on a wide range of values. Traub and Miles used an average connectivity of 0.2% in a 9,000-cell network ([25], p. 130), while Levy and

August used 10% connectivity for a 1,000 cell network ([3]). Hence even using a very low connectivity would require a prohibitive amount of memory just to store the connections, which seems to render Rangan and Cai's approach infeasible, though this has not yet been tested.

The only real drawback to the findings of Rangan and Cai is that they do not present timing data for resolving voltage trajectories. Large-scale network simulations in their paper show that statistical properties of the network can be captured in much less time than required for time-stepping methods, but individual voltages and firing times are not presented except for the case of a single neuron receiving input (Rangan and Cai [21]). Although they claim that such solutions in large networks can be achieved using a sufficiently small timestep, the lack of substantiating data makes it unclear whether the speed gains from the algorithm still hold. In this thesis I clarify the effect of the timestep on voltage accuracy of individual neurons in order to give a better representation of the accuracy of the network simulation.

Accuracy of large network simulations becomes an issue when the goal is to get IAF neurons to exhibit behaviors observed in real experiments. One such behavior, temporal compression, is the spontaneous replay of a firing sequence, with the replay occurring at a much faster rate than was experienced (August and Levy [3]). August and Levy show that this compression can be achieved using a network of 1000 IAF neurons (August and Levy [3]). However, it is not clear from their work whether the replay is a numerical artifact or the true solution. Convergence analysis could not be

done because I was unable to reproduce their figures. Furthermore, their use of the Forward Euler method with a 0.25 ms timestep appears dubious; this time-stepping method is not stable for arbitrary timesteps, but the standard method, Backward Euler, is unconditionally stable. It is because of discrepancies such as this that the results in this thesis are useful, for if neuroscientists can use mathematically-sound IAF simulation software, such as that presented here, then they will be more confident in asserting the accuracy of their findings.

# Chapter 2

## Methods

The main focus of this chapter is twofold: construction of the integrate-and-fire model, and solution methods for the resulting equations. Starting from the standard leaky cell model, spike rate adaptation and synaptic input components are added to it to build the full model. I present two solution techniques, time-stepping and quadrature, and discuss in detail how the latter is used to achieve highly accurate solutions. The chapter concludes with my algorithm for simulating multi-cell networks using the quadrature technique.

### 2.1 The Integrate-and-Fire Model

The basic leaky integrate-and-fire neuron has a simple ODE representation and can be drawn as a circuit, depicted in Figure 2.1. Using Kirchhoff's Current Law the change in voltage for the  $k$ th cell is

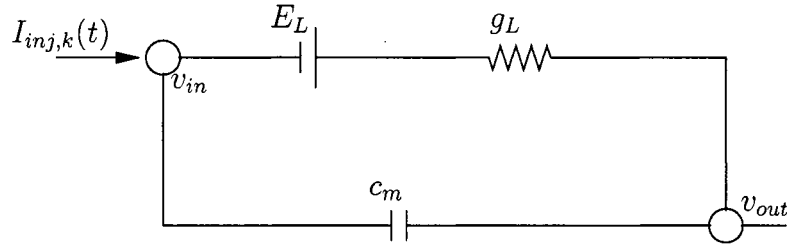
**while**  $V_k(t) \leq V_{th}$

$$Ac_m \frac{dV_k}{dt}(t) = -Ag_L(V_k(t) - E_L) + I_{inj,k}(t) \quad (2.1)$$

**end**



The cell membrane has surface area  $A$  and capacitance per unit area  $c_m$ . The membrane potential  $V_k(t)$  is the voltage difference across the membrane, and it tends to decay to its resting (reversal) potential  $E_L$  when no external stimulus is applied. This current leak is scaled by the conductance per unit area  $g_L$ , which is the reciprocal of  $r_m$ , the membrane resistance times area,  $r_m$ . An injected current  $I_{inj,k}(t)$  may stimulate the cell, thus inducing a change in  $V_k$ . A hard threshold  $V_{th}$  is imposed so that if  $V_k \geq V_{th}$  at time  $t_{th}$ , then cell  $k$  is said to have fired (or spiked) and instantaneously  $V_k$  is set to  $V_{reset}$ . A refractory period (a period after spiking during which the cell remains at rest and receives no input) may also be imposed for  $\tau_{ref}$  ms.



**Figure 2.1:** Basic circuit diagram of a single integrate-and-fire cell. The voltage  $V_k$  is defined as  $v_{in} - v_{out}$ . In this simple cell the leakage current flows through the top branch, while the capacitive current flows through the bottom branch.

The rate at which  $V_k$  decays to its resting potential is quantified by the membrane time constant,  $\tau_m = r_m c_m$ . The effect of injected current on the whole cell is characterized by the total membrane resistance,  $R_m = r_m/A$ . Inserting these constants into

(2.1) yields the more elegant expression of the IAF model neuron:

$$\tau_m \frac{dV_k}{dt}(t) = -(V_k(t) - E_L) + R_m I_{inj,k}(t). \quad (2.2)$$

Equation (2.2) is the fundamental IAF equation to which other currents may be added to represent different biophysical properties of the cell. In this thesis I treat two currents, one arising from spike rate adaptation and the other from synaptic input, because they are the most commonly used in the literature.

### 2.1.1 Spike Rate Adaptation

Hippocampal pyramidal neurons receiving constant injected current do not fire in a completely steady fashion, but rather the time between spikes increases slowly until a steady rate of firing is reached. This adaptation is mediated by the flow of potassium ( $K^+$ ) ions through the membrane (Liu and Wang [18]) and can be achieved in the model by including a conductance so that

$$\tau_m \frac{dV_k}{dt}(t) = -(V_k(t) - E_L) - r_m g_{sra,k}(t)(V_k(t) - E_K) + R_m I_{inj,k}(t), \quad (2.3)$$

where  $E_K$  is the reversal potential due to  $K^+$  and  $g_{sra,k}(t)$  is the spike rate adaptation conductance per unit area for cell  $k$ . Initially this conductance is zero, but whenever the cell fires the conductance is instantaneously increased by some fixed amount  $\Delta g_{sra}$ .

Hence the equation is discontinuous and is given by

**while**  $V_k(t) \leq V_{th}$

$$\tau_{sra} \frac{dg_{sra,k}}{dt}(t) = -g_{sra,k}(t). \quad (2.4)$$

**else**

$$g_{sra,k}(t) := g_{sra,k}(\hat{t}_k) + \Delta g_{sra}. \quad (2.5)$$

**end**

Integrating (2.4) directly yields

$$g_{sra,k}(t) = g_{sra,k}(\hat{t}_k) e^{-(t-\hat{t}_k)/\tau_{sra}} \quad (2.6)$$

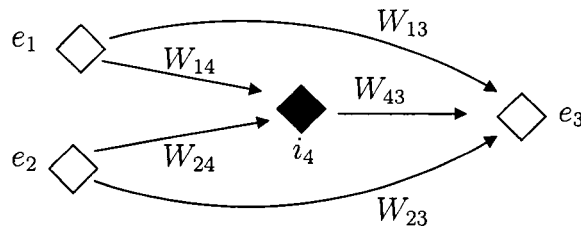
where  $t$  is the simulation time and  $\hat{t}_k$  is the start time of the ODE (i.e. the last firing time of cell  $k$ ). This equation is only valid from the most recent spike time until the next spike time.

### 2.1.2 Synaptic Input

The conductance which produces the richest variation in firing patterns for IAF cells is that due to intercellular communication, or synaptic input. In a network of neurons, each cell may connect to other cells via synapses, and each synapse can have a different strength (weight) of connection. These connections are represented in the matrix  $W$ , where  $W_{ij}$  is the weight from cell  $i$  (presynaptic cell) to cell  $j$  (postsynaptic cell).

Based on the general functions of neurons, two types of cells exist in this model: excitatory and inhibitory. Excitatory cells tend to increase the postsynaptic cell's

voltage, and hence their synaptic conductance will have a suprathreshold reversal potential  $E_{ex}$ , whereas inhibitory cells tend to decrease postsynaptic voltage and have a subthreshold reversal potential  $E_{inh}$ . These potentials are represented by the matrix  $E$ , where  $E_{ij}$  is the reversal potential of the synapse from cell  $i$  to cell  $j$ .



**Figure 2.2:** A 4-cell network with synaptic connections. Excitatory cells are white diamonds, and the lone inhibitory cell is a black diamond. Arrows represent the direction of synaptic connections, and the  $W_{jk}$  indicate the strength of each connection.

Firing of the presynaptic cell  $j$  causes a neurotransmitter to be released to the postsynaptic cells, a process governed by a nondimensional “alpha function”  $\alpha_j(t)$ . Though typical exponential forms for such functions are given in (Shelley and Tao [23]), I use a variant that is normalized to give intuition about the biophysics:

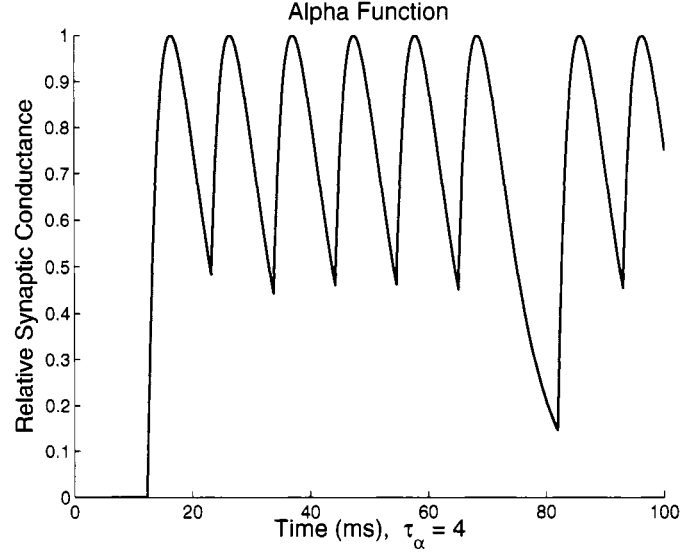
$$\alpha_j(t) = \begin{cases} \frac{(t - \hat{t}_j + t_{adj,j})}{\tau_\alpha} e^{1 - (t - \hat{t}_j + t_{adj,j})/\tau_\alpha} & \text{spike occurred at } \hat{t}_j \\ 0 & \text{otherwise} \end{cases} \quad (2.7)$$

where  $\hat{t}_j$  is the time of the most recent firing of cell  $j$ . The term  $t_{adj,j}$  is used as an adjustment so that  $\alpha_j$  starts from its value at the firing time. This adjustment time is easily computed via Newton’s Method. Note that  $\alpha_j(t)$  is nonnegative, and from

the derivative

$$\alpha'_j(t) = \frac{e^{1-(t-\hat{t}_j+t_{adj,j})/\tau_\alpha}}{\tau_\alpha} \left( 1 - \frac{t-\hat{t}_j+t_{adj,j}}{\tau_\alpha} \right) \quad (2.8)$$

it is clear that (2.7) peaks uniquely at  $\alpha_j(t) = 1$  when  $t - \hat{t}_j = \tau_\alpha$ . Therefore,  $\alpha_j$  is bounded by  $[0, 1]$  so it can be interpreted as the relative measure of synaptic conductance (see Figure 2.3).



**Figure 2.3:** The alpha function with  $\tau_\alpha = 4$  for a cell that is stimulated by synaptic input. Notice that the function peaks at 1, indicating saturation of neurotransmitter release. The firing times of this cell are observed as the points at which the alpha function begins to increase.

Putting the above pieces together yields the synaptic input to cell  $k$ :

$$I_{syn,k}(t) = \sum_{j=1}^N W_{jk} \alpha_j(t) (V_k(t) - E_{jk}). \quad (2.9)$$

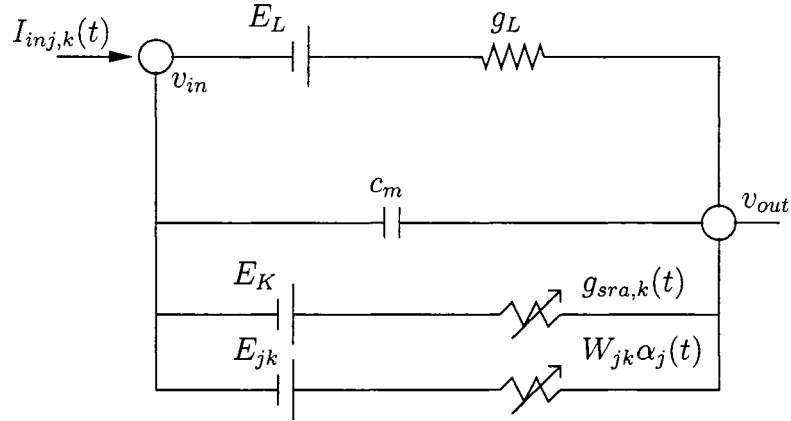
Since  $I_{syn,k}$  is a current per unit area, the synaptic weight  $W_{jk}$  is really a conductance

per unit area, and in fact it is interpreted as the maximum synaptic conductance from cell  $j$  to cell  $k$ .

Thus the full integrate-and-fire model equation is

$$\tau_m \frac{dV_k}{dt}(t) = E_L - V_k(t) - r_m g_{sra,k}(t)(V_k(t) - E_K) - r_m I_{syn,k}(t) + R_m I_{inj,k}(t). \quad (2.10)$$

This is shown schematically in Figure 2.4, and typical parameter values for this model are given in Table 2.1.



**Figure 2.4:** Full circuit diagram of a single integrate-and-fire cell. In this schematic only one synaptic input, from cell  $j$ , is shown, but in reality there is one branch per synaptic input.

## 2.2 Solution Methods

The IAF model equations are linear ODE's, and traditionally two techniques have been used to approximately solve them. Nearly all practical work has been performed using time-stepping because of its ease of implementation and speed. However, an-

Table 2.1. Integrate-and-Fire Model Parameters

Parameter	Value	Units	Description
$V_{th}$	-50	mV	Threshold potential
$V_{reset}$	-65	mV	Reset potential
$E_L$	-65	mV	Leakage potential
$E_K$	-70	mV	Potassium reversal potential
$E_{ex}$	0	mV	Excitatory reversal potential
$E_{inh}$	-70	mV	Inhibitory reversal potential
$c_m$	10	nF/mm <sup>2</sup>	Membrane capacitance
$r_m$	1	MΩmm <sup>2</sup>	Membrane resistance
$A$	0.1	mm <sup>2</sup>	Surface area of the membrane
$\tau_m$	10	ms	Membrane time constant
$\tau_{sra}$	10	ms	$g_{sra}$ time constant
$\tau_\alpha$	3	ms	Synaptic time constant
$I_{inj,k}$	Varies	nA	Injected current to cell $j$
$\Delta g_{sra}$	3	μS/mm <sup>2</sup>	Spike rate adaptation conductance increment
$W_{jk}$	Varies	μS/mm <sup>2</sup>	Synaptic weight from cell $j$ to cell $k$

alytic solutions exist and can be approximated using quadrature, which is the main focus of this thesis.

### 2.2.1 Time-stepping

Time-stepping methods partition the simulation domain  $[0, T]$  into subintervals and approximate the solution over each of these subintervals. Partitioning is achieved by choosing a timestep  $\Delta t$  and discretizing  $[0, T]$  into a set of points  $t_j = j\Delta t$ , and the approximation is computed using a finite difference approximation for the derivative  $dV_k/dt$ . Given the value  $V_k(t_j)$ , the next value  $V_k(t_{j+1})$  is easily computed via the time-stepping formula.

First-order schemes are commonly used in the neuroscience literature, and the most basic time-stepping scheme is Forward Euler, which finds  $V_k(t_{j+1})$  by moving a distance  $\Delta t$  along the line tangent to  $V_k(t)$ . Thus the derivative is approximated as

$$\frac{dV_k}{dt}(t) \approx \frac{V_k(t_{j+1}) - V_k(t_j)}{\Delta t} \quad (2.11)$$

and all other terms are evaluated at time  $t_j$  so that (2.10) becomes

$$\tau_m \frac{V_k(t_{j+1}) - V_k(t_j)}{\Delta t} = E_L - V_k(t_j) - r_m g_{sra,k}(t_j)(V_k(t_j) - E_K) - r_m I_{syn,k}(t_j) + R_m I_{inj,k}(t_j). \quad (2.12)$$

Solving for  $V_k(t_{j+1})$  yields

$$V_k(t_{j+1}) = \frac{\tau_m - \Delta t}{\tau_m} V_k(t_j) + \frac{\Delta t}{\tau_m} \left( E_L - r_m g_{sra,k}(t_j)(V_k(t_j) - E_K) - r_m I_{syn,k}(t_j) + R_m I_{inj,k}(t_j) \right). \quad (2.13)$$

The spike rate adaptation function is discretized in the same manner.

An alternative first-order scheme can be found by discretizing the derivative as in (2.11) but evaluating all other terms at the next timestep, i.e., at  $t_{j+1}$ . This leads to the Backward Euler scheme

$$V_k(t_{j+1}) = \frac{V_k(t_j) + \frac{\Delta t}{\tau_m} q(t_{j+1})}{1 + \frac{\Delta t}{\tau_m} p(t_{j+1})} \quad (2.14)$$



where, for clarity, I define

$$p(t) = 1 + r_m g_{sra,k}(t) + r_m \sum_{\ell=1}^N W_{\ell k} \alpha_{\ell}(t) \quad (2.15)$$

$$q(t) = E_L + E_K r_m g_{sra,k}(t) + r_m \sum_{\ell=1}^N W_{\ell k} \alpha_{\ell}(t) E_{\ell k} + R_m I_{inj,k}(t). \quad (2.16)$$

This method has an advantage over Forward Euler because it is unconditionally stable, which ensures that, over a given interval, solutions will not grow unboundedly (Süli and Mayers [24], p. 349).

Higher-order integrators exist, but dealing with the discontinuity due to firing limits their effectiveness because the firing time is only known to  $\mathcal{O}(\Delta t)$ . Thus schemes like classical Runge–Kutta will perform no better than first-order schemes unless they are modified (Shelley and Tao [23]). I have chosen here to fully consider only first-order schemes because higher-order methods do not perform well with discontinuous synaptic conductances, which I will introduce in §2.3. I use Backward Euler for the simulations in this thesis because of its stability property.

### 2.2.2 Analytic Solution Derivation

Observe that (2.10) only depends linearly on  $V_k$ . Defining

$$\begin{aligned} P_k(t) &= \frac{1}{\tau_m} \left( 1 + r_m g_{sra,k}(t) + r_m \sum_{j=1}^n W_{jk} \alpha_j(t) \right) \\ Q_k(t) &= \frac{1}{\tau_m} \left( E_L + E_K r_m g_{sra,k}(t) + R_m I_{inj,k}(t) + r_m \sum_{j=1}^n W_{jk} \alpha_j(t) E_{jk} \right) \end{aligned} \quad (2.17)$$

allows (2.10) to be written in the general form of a first-order non-constant coefficient ODE:

$$\frac{dV_k}{dt}(t) + P_k(t)V_k(t) = Q_k(t). \quad (2.18)$$

It is a well-known result that this equation has a closed-form solution, which can be obtained by using an integrating factor. Multiplying both sides of (2.18) by  $e^{\int_{t_0}^t P_k(s)ds}$  gives

$$e^{\int_{t_0}^t P_k(s)ds} \left[ \frac{dV_k}{dt}(t) + P_k(t)V_k(t) \right] = e^{\int_{t_0}^t P_k(s)ds} Q_k(t). \quad (2.19)$$

The left-hand side is just a derivative

$$\frac{d}{dt} \left[ e^{\int_{t_0}^t P_k(s)ds} V_k(t) \right] = e^{\int_{t_0}^t P_k(s)ds} Q_k(t), \quad (2.20)$$

and so integrating both sides with respect to  $t$  from  $t_0$  to  $t_f$  gives

$$V_k(t) e^{\int_{t_0}^t P_k(s)ds} \Big|_{t_0}^{t_f} = \int_{t_0}^{t_f} Q_k(t) e^{\int_{t_0}^t P_k(s)ds} dt. \quad (2.21)$$

Evaluating the left-hand side yields

$$V_k(t_f) e^{\int_{t_0}^{t_f} P_k(s)ds} - V_k(t_0) = \int_{t_0}^{t_f} Q_k(t) e^{\int_{t_0}^t P_k(s)ds} dt, \quad (2.22)$$

which, upon solving for  $V_k(t_f)$ , becomes

$$V_k(t_f) = e^{-\int_{t_0}^{t_f} P_k(t) dt} \left( \int_{t_0}^{t_f} Q_k(t) e^{\int_{t_0}^t P_k(s) ds} dt + V_k(t_0) \right). \quad (2.23)$$

The integral of  $P_k$  can be evaluated explicitly because  $\alpha_j$  and  $g_{sra,k}$  can be integrated explicitly. Integrating (2.6) yields

$$\int_{t_0}^t g_{sra,k}(s) ds = -\tau_{sra} g_{sra,k}(\hat{t}_k) (e^{-(t-\hat{t}_k)/\tau_{sra}} - e^{-(t_0-\hat{t}_k)/\tau_{sra}}) \quad (2.24)$$

where  $\hat{t}_k$  is the time of cell  $k$ 's most recent spike; if cell  $k$  has not fired then  $\hat{t}_k = 0$ .

Now for the integral of  $\alpha_j$ . If  $\hat{t}_j = 0$  then cell  $j$  has not fired and consequently  $\int_{t_0}^{t_f} \alpha_j(t) dt = 0$ . Otherwise it is clear that  $\hat{t}_j \leq t_0$ , so the integral of (2.7) is

$$\int_{t_0}^{t_f} \alpha_j(t) dt = \left[ -(t - \hat{t}_j + t_{adj,j} + \tau_\alpha) e^{\left(1 - \frac{t - \hat{t}_j + t_{adj,j}}{\tau_\alpha}\right)} \right]_{t_0}^{t_f} \quad (2.25)$$

With these two integrals in hand the integral of  $P_k$  is

$$\begin{aligned} \int_{t_0}^t P_k(t') dt' = \frac{1}{\tau_m} & \left( t - t_0 - r_m \tau_{sra} g_{sra,k}(\hat{t}_k) (e^{-(t-\hat{t}_k)/\tau_{sra}} - e^{-(t_0-\hat{t}_k)/\tau_{sra}}) \right. \\ & \left. + r_m \sum_{j=1}^n W_{jk} \left[ -(t - \hat{t}_j + t_{adj,j} + \tau_\alpha) e^{\left(1 - \frac{t - \hat{t}_j + t_{adj,j}}{\tau_\alpha}\right)} \right]_{t_0}^t \right). \end{aligned} \quad (2.26)$$

Substituting this into (2.23) reveals a difficulty in the analytic solution: evaluating (2.23) requires computing the integral of an exponential of an exponential, for which no closed-form solution exists. This difficulty necessitates the use of numerical

quadrature, which is the focus of this thesis.

### 2.2.3 Quadrature Methods

Quadrature is a technique whereby nodes  $x_j$  and weights  $w_j$  are chosen so that the sum of the weighted function values at the nodes approximates the integral, i.e.

$$\sum_{j=0}^n w_j f(x_j) \approx \int_a^b f(x) dx. \quad (2.27)$$

Gauss–Legendre quadrature is the most accurate method in principle, since it optimally chooses  $x_j$  and  $w_j$  to exactly integrate the highest-degree polynomial possible (Trefethen and Bau [28]). The nodes in (2.27) are the eigenvalues of a Jacobi matrix of dimension  $(n+1) \times (n+1)$ , and the weights are  $w_j = 2v_{j,1}^2$ , where  $v_{j,1}$  is the first component of the eigenvector  $v_j$ . This rule will exactly integrate a polynomial of degree  $\leq 2n+1$  (Trefethen and Bau [28]).

An alternative method, Clenshaw–Curtis (CC) quadrature, uses Chebyshev points as the nodes and optimally determines the weights. In this case the generic nodes, lying in the interval  $[-1, 1]$ , are

$$x_j = \cos\left(\frac{j\pi}{n}\right), \quad 0 \leq j \leq n \quad (2.28)$$

and the weights can be computed in  $\mathcal{O}(n \log n)$  operations via the Fast Fourier Transform (FFT) (Trefethen [26], Trefethen [27]). These nodes and weights can be shifted

to an arbitrary interval  $[a, b]$  via the formulas

$$\hat{x}_j = a + \frac{b-a}{2}(1+x_j), \quad \hat{w}_j = \frac{b-a}{2}w_j. \quad (2.29)$$

This scheme exactly integrates polynomials of degree  $\leq n+1$ , but it turns out that as functions become less smooth, the accuracy of CC quadrature is about the same as that of Gaussian quadrature (Trefethen [26]).

I chose to use CC quadrature for two reasons. The problem at hand is potentially not very smooth; thus I expect that CC quadrature will yield about the same accuracy as Gaussian quadrature. Also, in anticipation of exploring possibilities of adaptive quadrature, CC is appealing because the cost of computing new weights is much lower than for Gaussian quadrature.

#### 2.2.4 Analytic Solution via Composite CC Quadrature

It is clear that (2.23) cannot be efficiently evaluated as written over the whole interval  $[0, T]$  because of cell firings. The effect is twofold: first, since  $V_k(t)$  is instantaneously reset to  $V_{reset}$  upon firing, this will affect the initial condition of the ODE to be integrated; second, the synaptic input from cell  $k$  to its postsynaptic neighbors changes. The remedy is to use composite quadrature, whereby the domain  $[0, T]$  is discretized into subintervals  $[t_j, t_{j+1}]$  of width  $\Delta t$  and (2.23) is evaluated over each subinterval.

Knowing the value of  $V_k(t_j)$ , the next value is computed as

$$V_k(t_{j+1}) = e^{-\int_{t_j}^{t_{j+1}} P_k(t) dt} \left( \int_{t_j}^{t_{j+1}} Q_k(t) e^{\int_{t_j}^t P_k(t') dt'} dt + V_k(t_j) \right). \quad (2.30)$$

The term  $e^{-\int_{t_j}^{t_{j+1}} P_k(t) dt}$  is evaluated exactly using (2.26), and the integral involving  $Q_k(t)$  is approximated using CC quadrature. Now that the voltages can be accurately computed, the next step is to accurately compute spike times.

### 2.2.5 Spike Time Identification

Small perturbations in the computed firing times of individual cells have the potential to significantly impact the network as a whole, and thus it is critical to compute spike times accurately. The algorithm that is traditionally used in the literature is interpolation. I present a more accurate algorithm that uses the Bisection and Secant Methods.

Assume for the interval  $[t_j, t_{j+1}]$  that  $V_k(t_j) < V_{th}$  and  $V_k(t_{j+1}) > V_{th}$ . Then since  $V_k(t)$  is continuous,  $V_k(t_{th}) = V_{th}$  for some point  $t_{th} \in [t_j, t_{j+1}]$ . The goal of the Spike Time Identification (STI) algorithm is to compute  $t_{th}$  as accurately as possible.

The Bisection Method is a robust, albeit slow, algorithm on  $[t_j, t_{j+1}]$ . This converges at a rate of  $(1/2)^m$ , where  $m$  is the number of iterations. However, this requires many evaluations of (2.30), which is expensive. Quadratic convergence could be achieved by using Newton's Method, but that requires evaluation of a derivative, and since convergence is local there is no guarantee that Newton's Method would

succeed.

My STI algorithm is actually a combination of two methods, with the goal of achieving robustness and high accuracy with less computational effort. Two tolerances,  $\varepsilon_b$  and  $\varepsilon_s$ , are chosen such that  $0 < \varepsilon_s < \varepsilon_b$ . The Bisection Method is used until  $|V_k(t_m) - V_{th}| \leq \varepsilon_b$ , and then the Secant Method is used until  $|V_k(t_m) - V_{th}| \leq \varepsilon_s$ . Here the role of bisection is to get  $V_k(t)$  in an area where local convergence of the Secant Method is achieved, and then superlinear convergence according to the golden ratio  $\frac{1}{2}(1 + \sqrt{5})$  is achieved. The apparent loss in order of convergence from Newton's Method to the Secant Method is offset by the fact that the Secant Method requires no derivative evaluations and just one function computation at each iteration; hence it is only as expensive as bisection.

Since function evaluations are expensive, a natural alternative is to construct an interpolating polynomial  $p(t)$  over the interval  $[t_j, t_{j+1}]$  and to use one of the three aforementioned rootfinding methods to compute the  $t_{th}$ . The advantage is that all of the expensive work is in constructing  $p(t)$ , while the rootfinding is easy. Also, since  $p'(t)$  is easily computed, Newton's Method is applicable. My implementation of this STI algorithm constructs the cubic Hermite interpolating polynomial using  $V_k(t_j)$ ,  $V'_k(t_j)$ ,  $V_k(t_{j+1})$ , and  $V'_k(t_{j+1})$ , following the formula in (Shelley and Tao [23]). I first use Newton's Method, and if it fails then I use the Bisection Method. Evaluating  $p(t)$  and  $p'(t)$  is inexpensive, so tolerances may be set very low with minimal impact on simulation speed.

I chose to develop my STI algorithm because any desired numerical accuracy can be obtained by tuning the tolerances, hence it offers the ability to compute a reliable “exact” reference solution for error measurements, something that has been lacking in the literature. Results in §3.3.1 also show that my algorithm offers better convergence than the interpolation algorithm as the number of quadrature nodes is increased.

### 2.2.6 Multi-Cell Solution via Quadrature

In network simulations it is possible for multiple firings to occur simultaneously, and each one can have an impact on the dynamics of other cells. For this reason it is necessary to have an algorithm that can handle multiple firings properly. For generality define  $\mathcal{K}$  as the set of indices of cells that fired at  $t_{th}$ . Then the ODE system for all cells is valid for all  $t \in [t_j, t_{th}]$ . We can then reset  $V_k(t_{th})$ , increment  $g_{sra,k}(t_{th})$ , and update  $\alpha_k$  for all  $k \in \mathcal{K}$ , and start these ODE’s over with these new initial data. This process of calculating the spike time and resetting the ODE’s can be continued to obtain the solution for the time interval  $[t_j, t_{j+1}] = [t_j, t_{th}] \cup [t_{th}, t_{th_2}] \cup \dots \cup [t_{th_K}, t_{j+1}]$  where there are  $K$  firing times in this interval.

This leads to the main algorithm for solving (2.10) over the interval  $[0, T]$  using composite CC quadrature.



**Algorithm 1: Exact Solution to Multi-Cell Network**

```

Initialize network, set ODE's
Compute and store default CC nodes ( $x$ ) and weights ( $w$ )
for  $j = 1$  to number of subintervals
  while (just entered subinterval  $[t_j, t_{j+1}]$ ) OR (cells fired)
    Determine  $t_{pre}$ , the start of this subinterval
    Shift CC  $x$ ,  $w$  to  $[t_{pre}, t_{j+1}]$ 
    for  $k = 1$  to  $N$  (loop over all cells)
      Compute  $V_k(t_{j+1})$ 
      if  $V_k(t_{j+1}) \geq V_{th}$ 
        Use STI algorithm to compute  $t_{th,k}$ 
      end
    Determine  $\mathcal{K}$  (identify cells that fired, if any)
    if no cells fired
      Accept  $V_k(t_{j+1})$  for all cells and go to next subinterval
    else
      Compute earliest firing time  $t_{first}$  of all cells in  $\mathcal{K}$ 
      Remove cells from  $\mathcal{K}$  that did not fire at  $t_{first}$ 
      Shift CC  $x$ ,  $w$  to the interval  $[t_{pre}, t_{first}]$ 
      for  $k \notin \mathcal{K}$ 
        Compute  $V_k(t_{first})$ 
      for  $k \in \mathcal{K}$ 
        Set  $V_k(t_{first}) = V_{reset}$ 
        Increment  $g_{sra,k}(t_{first})$ 
        Compute  $\alpha_k(t_{first})$ 
        Set firing time for  $\alpha_k$  as  $t_{sp,k} = t_{first}$ 
        Compute adjustment time  $t_{adj,k}$  for  $\alpha_k$ 
      end
    end while
  end for

```

## 2.3 Modeling Synapses with Delta Functions

The model described in §2.1 attempts to maintain realism by modeling the neurotransmitter release functions explicitly. However, this is expensive because the effect of synaptic input on a given cell must be computed via a matrix-vector product. An alternative technique is to replace (2.9) with synaptic input equations that de-

cay exponentially and experience instantaneous changes, as in (2.5), when synaptic input occurs (Rangan and Cai [21]). This leads to a faster algorithm and allows for inexpensive modeling of another piece of hippocampal anatomy, the EC/DG layer.

### 2.3.1 Modeling the EC/DG Layer

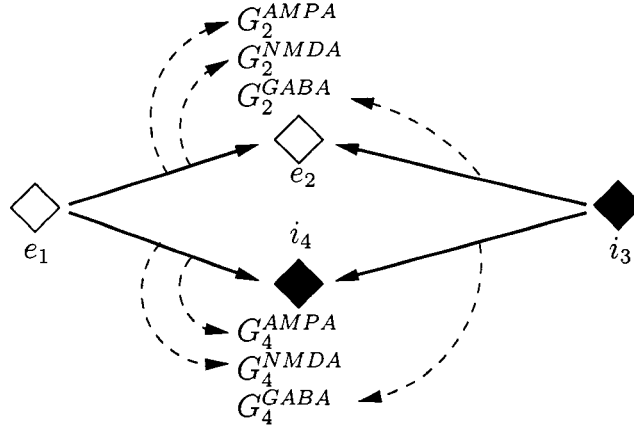
A small set of parameters governs the total behavior of the EC/DG layer, thus eliminating the need to simulate individual EC/DG cells. The only data needed from this layer are the spike times of EC/DG cells and what type of input (excitatory or inhibitory) each spike is. A Poisson process is used to generate these spike trains. Given the number of EC/DG cells,  $n_{ecdg}$ , and the rate of firing  $r$  (in Hz) of each cell, the probability of a spike occurring is given by the exponential distribution

$$P = \lambda e^{-\lambda}, \quad \lambda = \frac{r}{1000} n_{ecdg}. \quad (2.31)$$

This distribution can be used to generate the feedforward spike times  $T_j^F$  by the following formula:

$$T_{j+1}^F = T_j^F - \log(p)/\lambda, \quad (2.32)$$

where  $p$  is a uniform random number in  $[0, 1]$ . This spike is randomly designated as excitatory or inhibitory, based on the ratio  $n_{ex}/n_{ecdg}$ , where  $n_{ex}$  is the number of excitatory EC/DG cells, and the CA3 cell that receives this spike is randomly chosen.



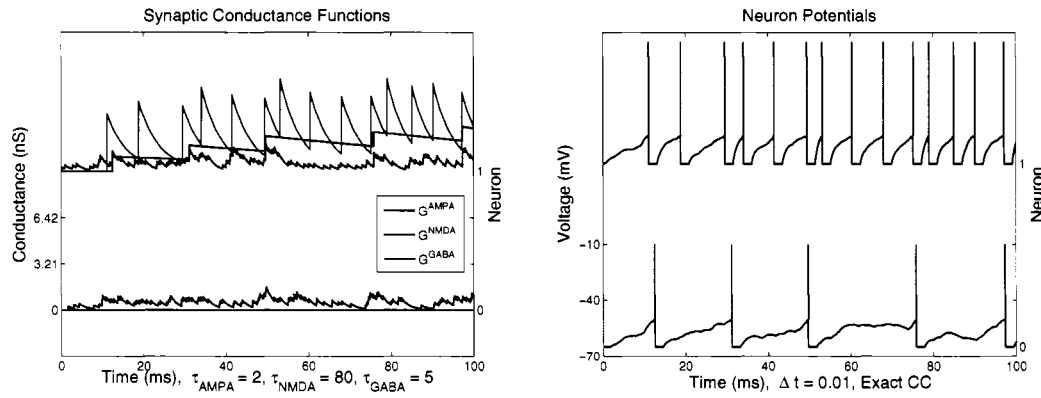
**Figure 2.5:** Diagram of synaptic connections in a 4-cell delta-synapse network. The white and black diamonds are excitatory and inhibitory cells, respectively. Thick black arrows indicate that cells  $e_1$  and  $i_3$  both connect to cells  $e_2$  and  $i_4$ . Dashed arrows indicate the receptors that are activated when the presynaptic cells release neurotransmitter (i.e., when they spike). Thus it is clear that excitatory cells influence only the AMPA and NMDA conductances, whereas inhibitory cells influence only the GABA conductance.

### 2.3.2 Delta-synapse Equations

Instead of modeling the outgoing synaptic release for each cell, this technique models the *incoming* synaptic conductances for each cell. Hippocampal cells are influenced by two main neurotransmitters, glutamate and GABA. Glutamate binds with AMPA and NMDA receptors on the postsynaptic cell to provide fast and slow excitation, while GABA binds with GABA<sub>A</sub> receptors to provide inhibition (Johnston and Amaral [16], pp. 439–442). Each neurotransmitter's effect on cell  $k$  is modeled by a conductance density  $G_k^Q(t)$ , where  $Q = \{\text{AMPA}, \text{NMDA}, \text{GABA}\}$ , so that  $Q$  is the set of receptor types. As shown in Figure 2.5, excitatory cells release only glutamate and thus they affect only AMPA and NMDA conductances, and inhibitory

cells release only GABA, and thus they affect only GABA<sub>A</sub> conductances. Hence at time  $t = T_{ij}$ , which is the  $j$ th spike time of neuron  $i$ , the synaptic input from cell  $i$  instantaneously increases  $G_k^Q(t)$  by some amount  $W_{ik}^Q$ .

In addition to recurrent CA3 connections, external synaptic inputs from the entorhinal cortex and dentate gyrus drive hippocampal cells. In this thesis, these inputs are combined into what is called the EC/DG layer. This layer of cells is a feedforward component, meaning that it receives no input, and thus it is necessary to model only the firing times of the EC/DG cells. The firing time of the  $j$ th EC/DG input of type  $Q$  to cell  $k$  is denoted  $T_{kj}^F$ . When  $t = T_{kj}^F$ , the EC/DG input instantaneously increases the corresponding  $Q$ -conductance of cell  $k$  by some amount  $F_k^Q$ .



**Figure 2.6:** a) Synaptic conductance functions in the delta-synapse model. AMPA input from 4 EC/DG cells that fire at a rate of 500 Hz stimulates both cells. Cell 2 is excitatory and synapses onto cell 1, but cell 1 is inhibitory and synapses onto itself. Notice in the left plot the fast and slow decay rates for AMPA and NMDA, and notice that although the GABA conductance is large, it also decays quickly. b) Voltages and spikes of each cell, shown here to provide context for the conductance changes observed in (a).

The instantaneous nature of these inputs is modeled by a delta function, and the

Table 2.2. Delta-synapse Conductance Function Parameters

Parameter	Value	Units	Description
$\epsilon^{\text{AMPA}}$	$E_{ex}$	mV	AMPA reversal potential
$\epsilon^{\text{NMDA}}$	$E_{ex}$	mV	NMDA reversal potential
$\epsilon^{\text{GABA}}$	$E_{inh}$	mV	GABA reversal potential
$\tau^{\text{AMPA}}$	2	ms	AMPA decay time constant
$\tau^{\text{NMDA}}$	80-100	ms	NMDA decay time constant
$\tau^{\text{GABA}}$	5	ms	GABA decay time constant
$W_{ik}^{\text{AMPA}}$	0.1-1	$\mu\text{S}/\text{mm}^2$	AMPA conductance density
$W_{ik}^{\text{NMDA}}$	0.001-0.1	$\mu\text{S}/\text{mm}^2$	NMDA conductance density
$W_{ik}^{\text{GABA}}$	0.1-4	$\mu\text{S}/\text{mm}^2$	GABA conductance density
$F_k^{\text{AMPA}}$	0.25	$\mu\text{S}/\text{mm}^2$	Feedforward AMPA conductance density

conductances decay exponentially, with associated time constants  $\tau^Q$  (see Figure 2.6).

Hence the  $Q$ -type synaptic input to cell  $k$  is

$$\frac{dG_k^Q}{dt}(t) = -\frac{G_k^Q(t)}{\tau^Q} + \sum_i^n \sum_j W_{ik}^Q \delta(t - T_{ij}) + \sum_j F_k^Q \delta(t - T_{kj}^F). \quad (2.33)$$

Note that (2.33) is a linear ODE, so using the general solution derived in §2.2.2, and

recalling that  $\int \delta(t - T) f(t) dt = f(T)$ , yields

$$G_k^Q(t) = G_k^Q(t_0) e^{-(t-t_0)/\tau^Q} + \sum_i^n \sum_{j|t_0 < T_{ij} \leq t} W_{ik}^Q e^{-(t-T_{ij})/\tau^Q} + \sum_{j|t_0 < T_{kj}^F \leq t} F_k^Q e^{-(t-T_{kj}^F)/\tau^Q}. \quad (2.34)$$

Hence with these conductances, the term  $I_{syn,k}(t)$  in (2.10) becomes

$$I_{syn,k}(t) = \sum_Q G_k^Q(t)(V_k(t) - \epsilon^Q) \quad (2.35)$$

where  $\epsilon^Q$  is the reversal potential of the  $Q$ -type conductance. I apply the techniques explained in §2.2 to solve the resulting voltage equations. Typical parameters for these conductance functions are given in Table 2.2.

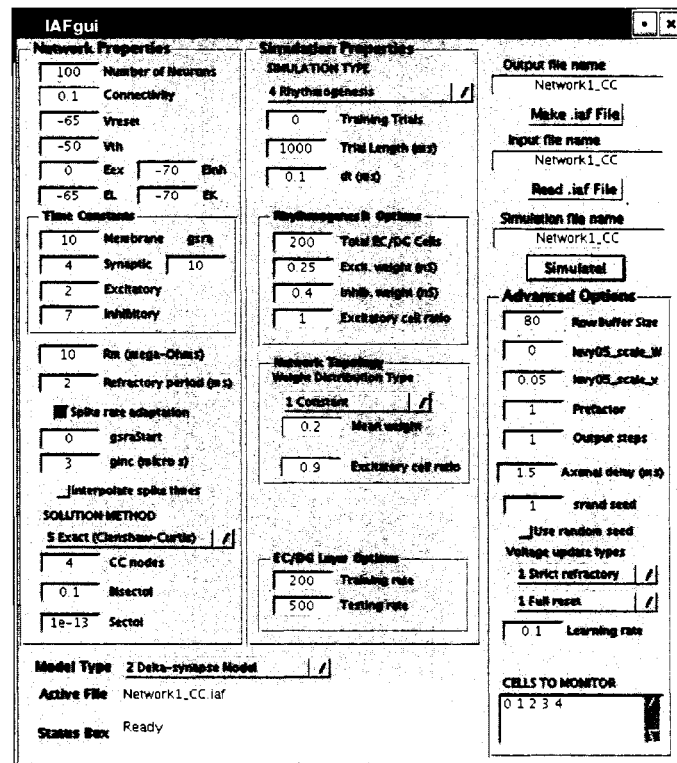
## 2.4 Simulation Platform and Visualization Tools

Efficient and accurate software is crucial for assessing viability of solution techniques and evaluating performance of network simulations, but building such robust code from scratch is tedious and error-prone. The software package developed here implements numerical methods effectively and includes two GUI's in MATLAB that allow the user to generate and simulate networks quickly and to visualize the resulting data without having to write any code. The software includes both time-stepping and quadrature methods, and it can simulate with either the standard alpha function model or the delta-synapse model.

By combining powerful numerical methods with an interface designed specifically for hippocampal modeling, this software should appeal to researchers and students who simulate hippocampal networks. This software has a direct impact on the hippocampus research group at Rice University because it finally provides a standard

simulation platform that is accessible to the group and that is tailored to its needs. The software is flexible enough that it can be extended to include model types that will be developed in the future.

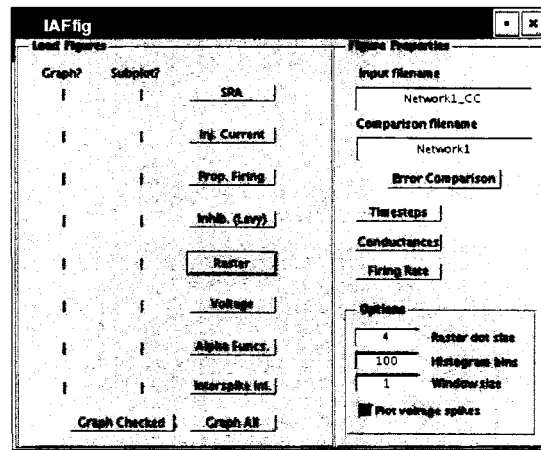
Written in C++, this code utilizes matrix and vector routines from the PETSc package (Balay et al. [5]). Though the code currently runs on a single processor, the purpose of using PETSc is to ease the task of parallelizing the code in the future.



**Figure 2.7:** The simulation GUI in MATLAB. Parameters are organized according to what type of simulation will be run and what type of solution method will be used. Output files are generated automatically and they can be used to run simulations directly from the GUI.

The simulation GUI (Figure 2.7) allows the user to set nearly all model parameters

and simulation properties, to choose from different types of stimulus currents, to determine the network topology, and to control the solution method used. Some features, such as creating custom network topologies, are not yet in the GUI, but are available as auxiliary codes that any user acquainted with MATLAB can execute. Some old features from a previous IAF model exist in the code and GUI, but they remain because future work will make them available for the IAF models in this thesis.



**Figure 2.8:** The visualization GUI in MATLAB. The data analysis and plotting codes are organized as buttons; textboxes allow the user to set certain plotting options.

The visualization GUI (Figure 2.8) contains tools which let the user plot data in an attractive and effective way in MATLAB. These tools plot various types of data, the most important of which are firing patterns, voltage trajectories, spike rate adaptation conductances, neurotransmitter release trajectories, synaptic conductances, and firing rate information. In addition, given two voltage data files, the user can generate error curves between these simulations to assess accuracy of different solution techniques



and to determine convergence of computed solutions.

It should be noted that I use three auxiliary codes to compute the initial quadrature data. Two of these codes have been borrowed and modified from Prof. Nick Trefethen (Trefethen [27], pp. 128–129). Gauss–Lobatto nodes and weights (see §3.3.4) are computed using a modified version of Greg von Winckel’s code (von Winckel [29]).

# Chapter 3

## Results

The time-stepping and quadrature solution methods are assessed for speed and convergence in this section. After establishing accuracy of the quadrature method for the alpha function and delta-synapse models, I quantify the computational complexity of Algorithm 1 with both types of STI algorithms. This complexity is compared with that of Algorithm 2, and from this I offer theoretical and computational evidence to favor choosing the Bisection / Secant STI algorithm over interpolation for the delta-synapse model. Using this model, I close with a presentation of simulations that demonstrate network rhythmogenesis.

### 3.1 Single Neuron Dynamics

The most basic measure of the performance of the solution methods for (2.10) is the error between the computed voltages and the true voltages at each  $t_j$ . The true solution will not exist in closed-form for real-world problems, but certain assumptions permit the voltages to be obtained exactly.

Assume that the network consists of a single neuron, and spike rate adaptation

and synaptic input are not present in the model. Also assume that the injected current is constant ( $I_{inj}(t) = I_0$ ). Then the exact solution for the interval  $[t_0, T]$  can be obtained from (2.30) as

$$V(t) = e^{-(t-t_0)/\tau_m} \left( - (E_L + R_m I_0) + V(t_0) \right) - V(t_0). \quad (3.1)$$

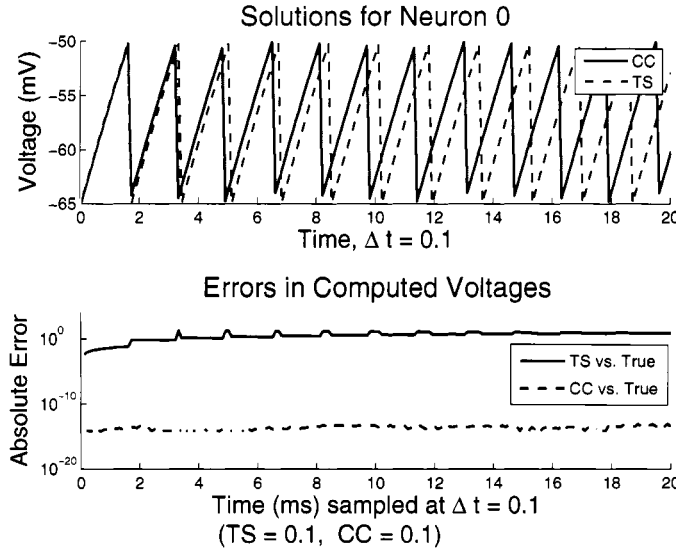
This equation is valid up until the firing time  $t_{th}$ , which can be found in closed-form:

$$t_{th} = -\log \left( \frac{V_{th} - (E_L + R_m I_0)}{V(t_0) - (E_L + R_m I_0)} \right). \quad (3.2)$$

Since  $g_{sra}$  is not active, the cell will fire at multiples of  $t_{th}$ , and hence the true solution for a driven single cell is known.

Figure 3.1 shows the computed voltage trajectories using the time-stepping method and the quadrature method. In both cases  $\Delta t = 0.1$ , but observe that in the lower plot the error is nearly machine precision for the quadrature method, whereas the time-stepping method has quickly become completely inaccurate.

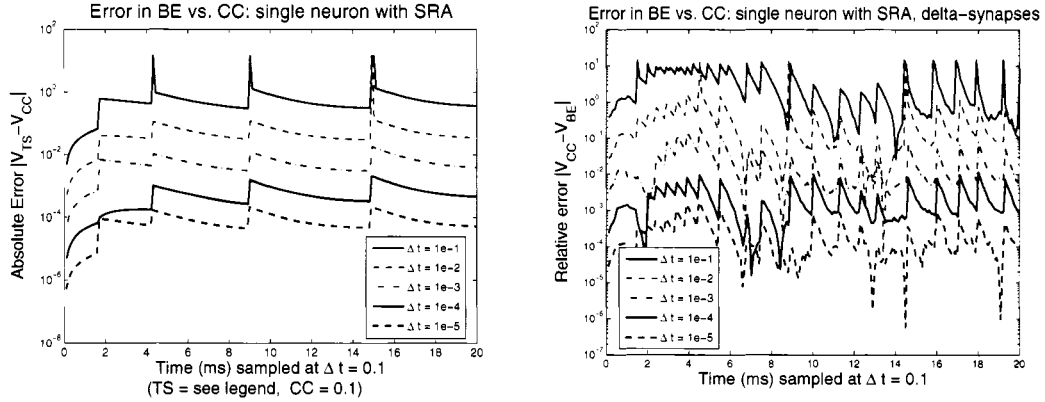
If spike rate adaptation is included in this network simulation, then it is not possible to obtain the true solution without the use of quadrature, and thus exact errors cannot be measured. However, since Backward Euler will converge to the true solution as  $\Delta t \rightarrow 0$ , the accuracy of the two solution methods in relation to each other can be obtained. Errors in voltage trajectories between time-stepping simulations for decreasing  $\Delta t$  and the CC method with fixed  $\Delta t = 0.1$  ms are shown in Figure 3.2a.



**Figure 3.1:** Single neuron voltage trajectories and error curves. a) The top panel shows the computed voltage trajectories using the Clenshaw–Curtis quadrature technique (solid blue line) and the Backward Euler time-stepping technique (dashed red line). b) The absolute error between the exact solution (as given by equations (3.1)–(3.2)) and the computed solutions. Notice that the time-stepping method loses a great deal of accuracy, whereas the quadrature method maintains near machine-precision over the whole simulation.

Clearly the quadrature method is more accurate than the time-stepping method. The simulation time for  $\Delta t = 10^{-5}$  ms was 98 seconds, while the CC method required just 0.05 seconds.

Using the delta-synapse model it is not possible to evaluate any closed-form solution exactly because the feedforward spikes are randomly generated. However, relative error measurements can still be produced, as shown in Figure 3.2b. Once again the quadrature method is highly accurate and requires just a fraction of the computation time required for Backward Euler.



**Figure 3.2:** Error curves for a single neuron with spike rate adaptation for synaptic input modeled with a) alpha functions and b) delta-synapses. The plot on the left corresponds to a cell driven by a 4 nA constant current, while the plot on the right corresponds to a cell driven by 40 EC/DG AMPA cells with firing rate 500 Hz each. The absolute errors between the solutions computed via time-stepping and quadrature are plotted for decreasing  $\Delta t$  in the time-stepping method.

### 3.2 A Simple Function: The XOR Network

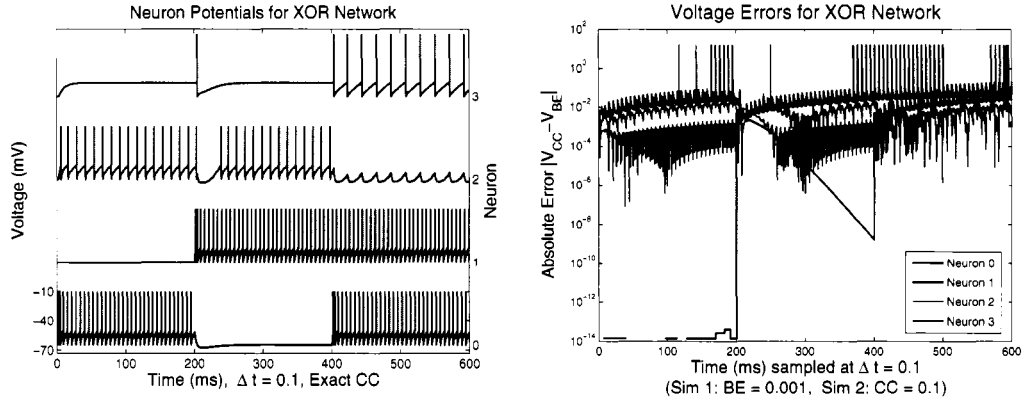
Complex cognitive functions cannot be pursued without first verifying that the model can reproduce much simpler functions. A function with both excitatory and inhibitory dynamics is the XOR (exclusive or) function. Given two input cells  $e_0$  and  $e_1$ , a third cell  $e_2$  should fire only when one of the input cells fires, but not both. This is only possible with an inhibitory cell  $i_3$  included in the model.

The network topology is given in the weight matrix

$$W = \begin{pmatrix} 0 & 0 & 1 & 0.3 \\ 0 & 0 & 1 & 0.3 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 50 & 0 \end{pmatrix}. \quad (3.3)$$

The first three rows of  $W$  correspond to  $e_1$ ,  $e_2$ , and  $e_3$ , the excitatory cells, and the

fourth row corresponds to the inhibitory cell  $i_4$ . Using  $I_{inj,k}(t) = 10$  nA, the values of the weights from cells  $e_1$  and  $e_2$  have been chosen so that when either of these cells fires alone there will be sufficient stimulation to cause  $e_3$  to fire but not  $i_4$ . A successful simulation, computed via quadrature, is displayed in the firing diagram and voltage traces of Figure 3.3. Similar results can be achieved using delta-synapses.



**Figure 3.3:** Results of a simulation that successfully reproduces the XOR behavior. For this simulation, a pulse of 10 nA current is delivered for 200 ms to cell  $e_1$ , then for the next 200 ms the same current is delivered to cell  $e_2$ , and finally for the last 200 ms both cells receive the same 10 nA current. a) The lefthand plot shows voltages and spikes for each cell. Notice that when cell  $i_4$  fires, the voltage for  $e_3$  is depressed and remains this way until the inhibitory conductance has sufficiently decayed. The time constant is  $\tau_{syn} = 5$  ms and the inhibitory weight is  $50 \mu S$ . b) The righthand plot shows error curves for the computed voltages for each cell. The errors are measured with respect to the quadrature method, and it is clear that 2–3 digit accuracy is maintained. However, the time-stepping method required 31 seconds, while the quadrature method required just 2.4 seconds.

### 3.3 Efficiency and Accuracy of the Quadrature Method

To measure the efficiency of the quadrature method is to ask how expensive the algorithm is in terms of arithmetic operations. For the purposes of this thesis, one

operation is defined as one  $+$ ,  $-$ ,  $\times$ , or  $\div$  operation. Though in reality evaluations of trigonometric or exponential functions are more expensive, they are also treated as single operations here because the point is to get a rough estimate of complexity. With the computational complexity of the quadrature method derived, numerical experiments demonstrate how my simulation software performs when key parameters are changed.

### 3.3.1 Computational Complexity: Using Alpha Functions

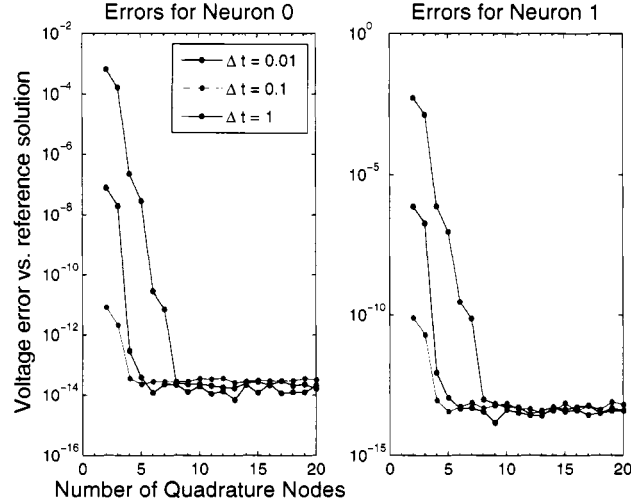
The operation count for the analytic solution for a single cell, equation (2.30), is described by two key terms:

$$V_k(t_{j+1}) = \underbrace{e^{-\int_{t_j}^{t_{j+1}} P_k(t) dt}}_{\text{outer exponential}} \left( \underbrace{\int_{t_j}^{t_{j+1}} Q_k(t) e^{\int_{t_j}^t P_k(t') dt'} dt}_{\text{quadrature}} + V_k(t_j) \right). \quad (3.4)$$

The outer exponential involves a definite integral, and (2.26) shows that this computation is composed of three terms. The first two terms require  $\mathcal{O}(1)$  operations. The third term, the contribution to  $P_k$  from synaptic input, involves a sum over the number of cells,  $N$ . However, since  $c \in [0, 1]$  is the connectivity of the network, on average each cell will have only  $cN$  synaptic inputs, and thus this term requires  $\mathcal{O}(cN)$  operations. Hence the first bracketed term of (3.4) costs  $\mathcal{O}(1) + \mathcal{O}(1) + \mathcal{O}(cN) = \mathcal{O}(cN)$  operations.

The term in parenthesis in (3.4) is where the quadrature method is used. For a single quadrature node both the exponential of  $P_k$  and  $Q_k$  at the node must be

evaluated. From the preceding paragraph's result, the exponential of  $P_k$  requires  $\mathcal{O}(cN)$  operations, and using similar reasoning the formula for  $Q_k$  in (2.17) requires  $\mathcal{O}(cN)$  operations. Using  $N_{CC}$  quadrature nodes, the total cost for the second term is  $N_{CC}\mathcal{O}(2cN)$  operations. This scaling is not detrimental, though, because results displayed in Figure 3.4 indicate that only a small  $N_{CC}$  is typically needed to obtain high accuracy.



**Figure 3.4:** Errors in computed voltages versus the number of quadrature nodes used in the alpha function model. An accurate reference solution is obtained by simulating a 2-cell, fully connected network. The cells are stimulated with sinusoidal injected current over an interval of 100 ms using  $N_{CC} = 20$  for a given  $\Delta t$ . This solution is compared to those computed by the quadrature method with a range of  $N_{CC}$  values. The resulting curves demonstrate that even for large  $\Delta t$  only a small number of quadrature nodes are needed to achieve near machine-precision accuracy.

For a network with  $N$  cells both of the above terms must be computed  $N$  times, bringing the total computational complexity of one iteration of the CC quadrature method to  $(N_{CC} + 1)\mathcal{O}(2cN^2)$  operations. A full simulation for  $T$  ms with timestep



$\Delta t$  thus requires  $\frac{T}{\Delta t}(N_{CC} + 1)\mathcal{O}(2cN^2)$  operations. This is actually an underestimate because the STI algorithm contributes a nontrivial cost.

Both of the STI algorithms must iterate to isolate the firing time of a single cell, after which the voltage for all cells from  $[t_j, t_{first}]$  must be computed. One iteration of the Bisection/Secant algorithm involves computing the voltage at the end of some interval  $[t_j, b]$ , and hence it costs  $(N_{CC} + 1)\mathcal{O}(2cN)$  operations per iteration.

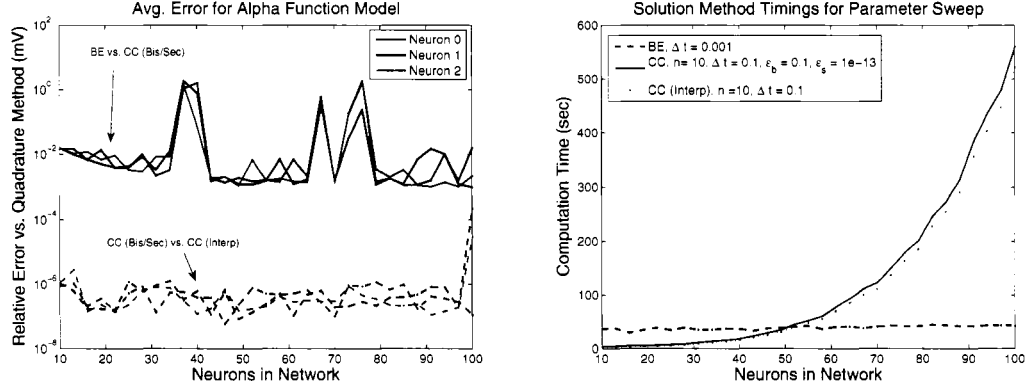
The interpolation algorithm must evaluate the derivative of  $V_k$  at the endpoints of the interval before forming the interpolation polynomial. This task requires  $\mathcal{O}(2cN)$  operations, but after that each step of the rootfinding method costs just  $\mathcal{O}(1)$  operations.

The update of all cells once the earliest firing time is found is the same for both methods:  $(N_{CC} + 1)\mathcal{O}(2cN^2)$  operations. Depending on the tolerances of each STI method, the number of iterations to reach these tolerances will vary, but on average each firing event will require the same number of iterations to isolate it. Let  $S_S$  and  $S_I$  be the average number of iterations for the Bisection/Secant and Interpolation algorithms, respectively. Assuming that there are  $F$  distinct firing times during the whole simulation then the estimate of the total cost of each STI algorithm is  $F \times [(\text{Avg. STI iterations}) \times (\text{STI cost per iteration}) + \text{Update cost}]$ . Combining this with the basic cost of the simulation yields the approximate total computational complexity, summarized in Table 3.1.

For networks of up to about 50 neurons, the quadrature method with  $\Delta t = 0.1$

Table 3.1. Complexity of Quadrature Method Simulation: Alpha Function Model

	Complexity (operations)
Total cost	$\frac{T}{\Delta t}(N_{CC} + 1)\mathcal{O}(2cN^2) + \text{STI}$
STI: Bisection/Secant	$F \times [S_S(N_{CC} + 1)\mathcal{O}(2cN) + (N_{CC} + 1)\mathcal{O}(2cN^2)]$
STI: Interpolation	$F \times [S_I\mathcal{O}(1) + \mathcal{O}(2cN) + (N_{CC} + 1)\mathcal{O}(2cN^2)]$



**Figure 3.5:** Quadrature method errors and timings versus  $N$ , the number of neurons in the network, for the alpha function model. a) Errors between the computed voltage solutions for the time-stepping method vs. quadrature method with Bisection / Secant STI (solid lines) and for the quadrature method with interpolation STI vs. Bisection / Secant STI (dashed lines). b) Computation time (in seconds) for the different solution methods.

ms,  $\varepsilon_b = 0.1$ , and  $\varepsilon_s = 10^{-13}$  outperforms the time-stepping method with  $\Delta t = 0.001$  ms both in speed and in accuracy, as evidenced in Figure 3.5. The interpolation STI algorithm cuts the computation time slightly as  $N$  increases, but this gain is negligible compared to the  $N^2$  scaling observed. Over all simulations the quadrature method is, as expected, more accurate than the time-stepping method, and the error

Table 3.2. Complexity of Quadrature Method Simulation: Delta-synapse Model

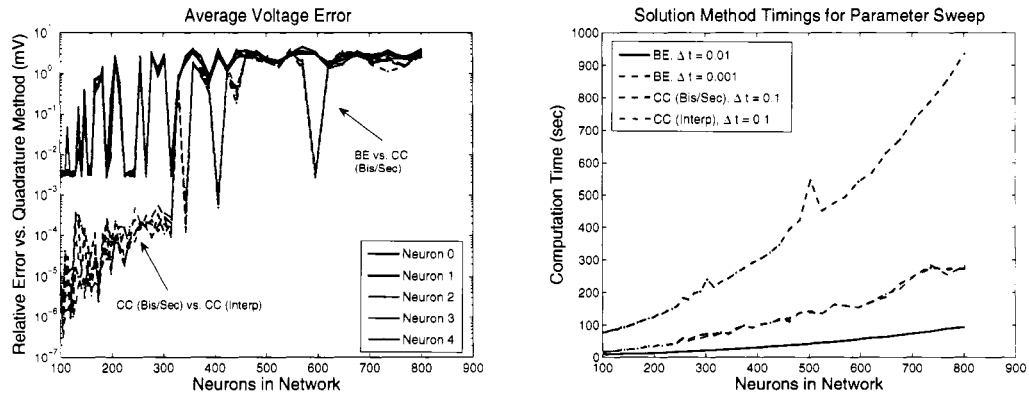
	Complexity (operations)
Total cost	$\frac{T}{\Delta t}(N_{CC} + 1)\mathcal{O}(2fN) + \text{STI}$
STI: Bisection/Secant	$F \times [S_S(N_{CC} + 1)\mathcal{O}(2f) + (N_{CC} + 1)\mathcal{O}(2fN)]$
STI: Interpolation	$F \times [S_I\mathcal{O}(1) + \mathcal{O}(2f) + (N_{CC} + 1)\mathcal{O}(2fN)]$

between solutions computed using the two different STI algorithms maintains about 6 digits of accuracy. These results demonstrate that the interpolation procedure does not save much time because it scales asymptotically as  $N^2$ , and hence it makes more sense to use a more accurate STI algorithm, namely the Bisection / Secant algorithm, for a similar computational cost.

### 3.3.2 Computational Complexity: Using Delta-synapses

For the delta-synapse model the analysis is similar, but the results are much more encouraging. The terms  $P_k(t)$  and  $Q_k(t)$  have no matrix-vector products, so this reduces the complexity by an order of magnitude. First let  $f$  be the average number of feedforward spikes per cell occurring in an interval. Following the above procedure, the computational complexity for the delta-synapse model simulation is  $\frac{T}{\Delta t}(N_{CC} + 1)\mathcal{O}(2fN)$ . Similarly, the STI algorithms also require an order of magnitude less work. As can be seen in Table 3.2, this implies that the time to simulate the delta-synapse model scales linearly with  $N$ , rather than quadratically for the alpha

function model, making this model more attractive computationally.

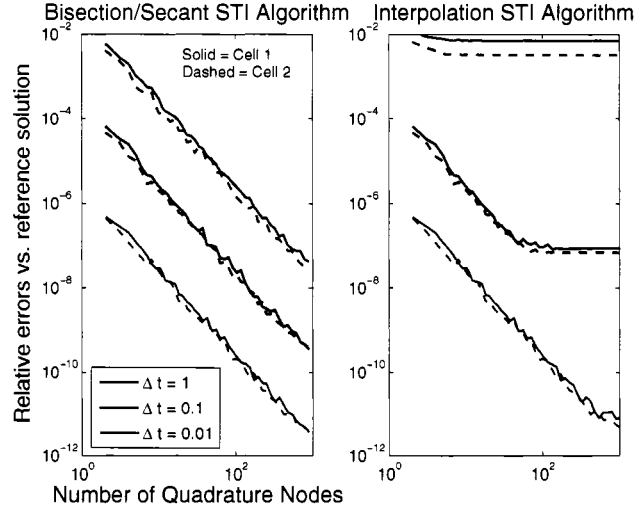


**Figure 3.6:** Quadrature method errors and timings versus  $N$ , the number of neurons in the network, for the delta-synapse model. a) Errors between the computed voltage solutions for the Backward Euler time-stepping method vs. quadrature method with Bisection / Secant STI (solid lines) and for the quadrature method with interpolation STI vs. Bisection / Secant STI (dashed lines). b) Computation time (in seconds) for the different solution methods. Simulations were run for 1000 ms. Networks consisted of 50% excitatory cells and 50% inhibitory cells in order to prevent “seizure”, where nearly all cells fired in a period of just a few milliseconds. Though cells received sparse EC/DG input, the cells that did spike connected to enough other cells that the recurrent connections led to rapid synaptic input across the network.

As evidenced in Figure 3.6, the quadrature method with the delta-synapse model is very efficient at simulating large networks, and in practice the time to run a simulation with either STI algorithm is about the same. Hence the Bisection / Secant algorithm is again preferred because it is more accurate. Additionally, the size of networks that can be simulated increases by an order of magnitude.

The drawback to the delta-synapse model is that the delta functions introduce discontinuities that make it more difficult for the quadrature method to accurately approximate the inner integral of (2.23). A similar experiment as that used for the alpha function model is performed, but the results in Figure 3.7 show that as  $N_{CC}$

increases, the convergence rate is exponential, and indeed many nodes are required in order to achieve near-machine precision relative accuracy for a given  $\Delta t$ . When the interpolation STI algorithm is used, the computed solutions are less accurate (as expected) but the accuracy also stagnates for  $N_{CC}$  large enough, which limits the performance of the interpolation scheme.



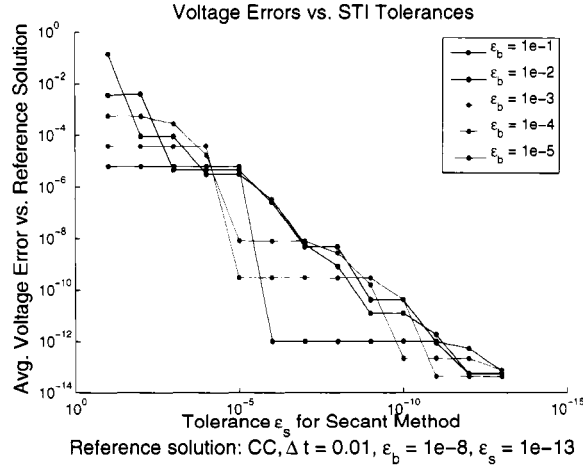
**Figure 3.7:** Relative errors in computed voltages versus the number of quadrature nodes used. As in Figure 3.4, an accurate reference solution is obtained by simulating a 2-cell, fully connected network for 100 ms at a given  $\Delta t$ , but this time the cells are stimulated with spike trains from a 4-cell EC/DG layer with input rate 500 Hz. This solution is compared to those computed by the quadrature method with a range of  $N_{CC}$  values and with the two STI algorithms. In this case, the errors in the computed solutions decrease logarithmically with  $N_{CC}$ . Stagnation occurs for the interpolation algorithm once  $N_{CC}$  is large enough for a given timestep, but the Bisection / Secant algorithm continues to converge as more nodes are used.

The implication is that, for small networks, in order to get the most accurate solutions via quadrature, the computational complexity of the delta-synapse simulation may scale poorly, thus making it as time-consuming as, or slower than, the

alpha function model simulation. For large networks this scaling should become less significant.

### 3.3.3 Effect of STI Tolerances

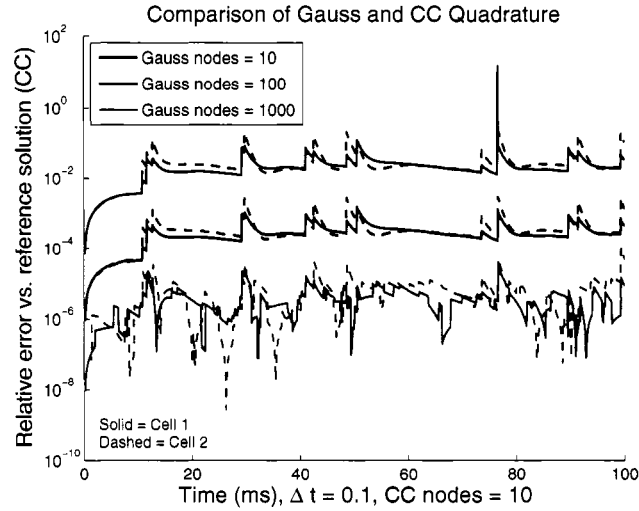
Having established the efficiency and accuracy of the quadrature method as the number of nodes and number of neurons are varied, it is important to quantify how the tolerances affect the Bisection / Secant STI algorithm's convergence. Consider the single neurons that were used to assess accuracy in Figure 3.2. A reference solution is computed with  $\Delta t = 0.01$  ms,  $\varepsilon_b = 10^{-8}$ , and  $\varepsilon_s = 10^{-13}$ . This reference solution is compared with solutions computed by varying  $\varepsilon_s$  for fixed  $\varepsilon_b$ , as shown in Figure 3.8. Ultimately, convergence to any desired tolerance  $\varepsilon_s$  is achieved, and thus  $\varepsilon_b$  can be large, thus saving computation time.



**Figure 3.8:** Errors in the computed voltages of a single neuron with SRA turned on (see Figure 3.2 for parameters). Flat regions on these curves indicate that the desired tolerance has already been reached; it does not imply any negative stagnation.

### 3.3.4 Clenshaw–Curtis Quadrature vs. Gaussian Quadrature

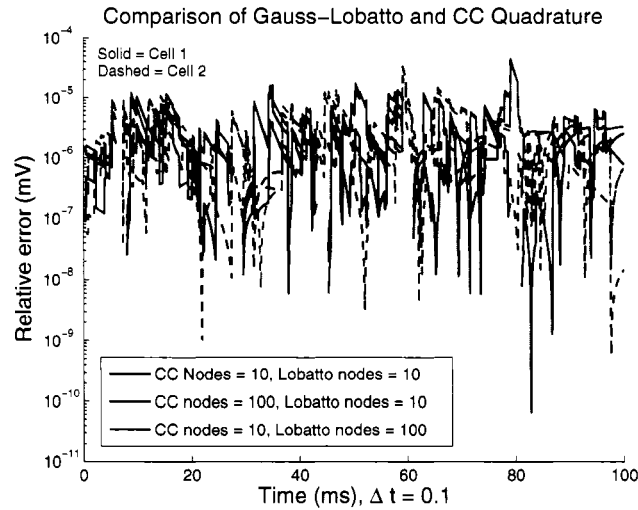
While quadrature methods can increase the accuracy of computed voltage solutions, some quadrature methods are more appropriate than others for solving the IAF equations. Specifically, the choice of integration nodes  $x_j$  impacts the accuracy of the computed solutions.



**Figure 3.9:** Relative errors in computed voltages for Gaussian and Clenshaw Curtis quadrature schemes. Using CC quadrature with  $N_{CC} = 10$ , a reference solution is obtained by simulating a 2-cell, fully connected network for 100 ms with  $\Delta t = 0.1$  and with stimulation from a 4-cell EC/DG layer with input rate 500 Hz. This solution is compared to those computed using Gaussian quadrature for various numbers of nodes. The errors between these solutions decrease as Gaussian quadrature uses more nodes.

Typically, Gaussian quadrature has been used to compute the approximate solution to (2.10), but when compared to CC quadrature it actually performs worse for this problem (see Figure 3.9). This result is counterintuitive, since Gaussian quadrature will exactly integrate polynomials of higher degree than will CC quadrature.

Examining the nodes used in these schemes provides the remedy. In Gaussian quadrature, the nodes lie in the interval  $(a, b)$ ; for CC quadrature, they lie in  $[a, b]$ , with the extreme nodes being  $x_0 = a$  and  $x_n = b$ . It turns out that forcing the extreme nodes in Gaussian quadrature to be the endpoints of the integration interval and optimally choosing the remaining  $n - 2$  nodes, as in Gauss–Lobatto quadrature, is more appropriate than standard Gaussian quadrature for some problems (Davis and Rabinowitz [11], pp. 104–105; Golub [13]). Using Gauss–Lobatto quadrature significantly reduces the discrepancy observed in Figure 3.9, so that 5–7 digits of agreement are shown, as in Figure 3.10.

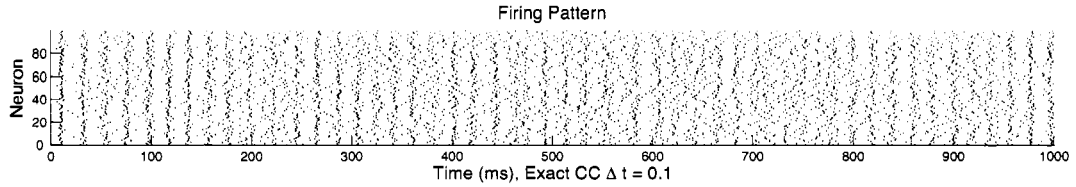


**Figure 3.10:** Relative errors in computed voltages for Gaussian and Clenshaw–Curtis quadrature schemes. The same network as in Figure 3.9 is simulated for  $\Delta t = 0.1$ , but this time CC and Gauss–Lobatto quadrature schemes are compared for different numbers of nodes. Notice that the errors are basically the same (5–7 digits of agreement) no matter if the same number of nodes is used or if one quadrature method uses more nodes than the other (see the legend).



### 3.4 Network Rhythmogenesis

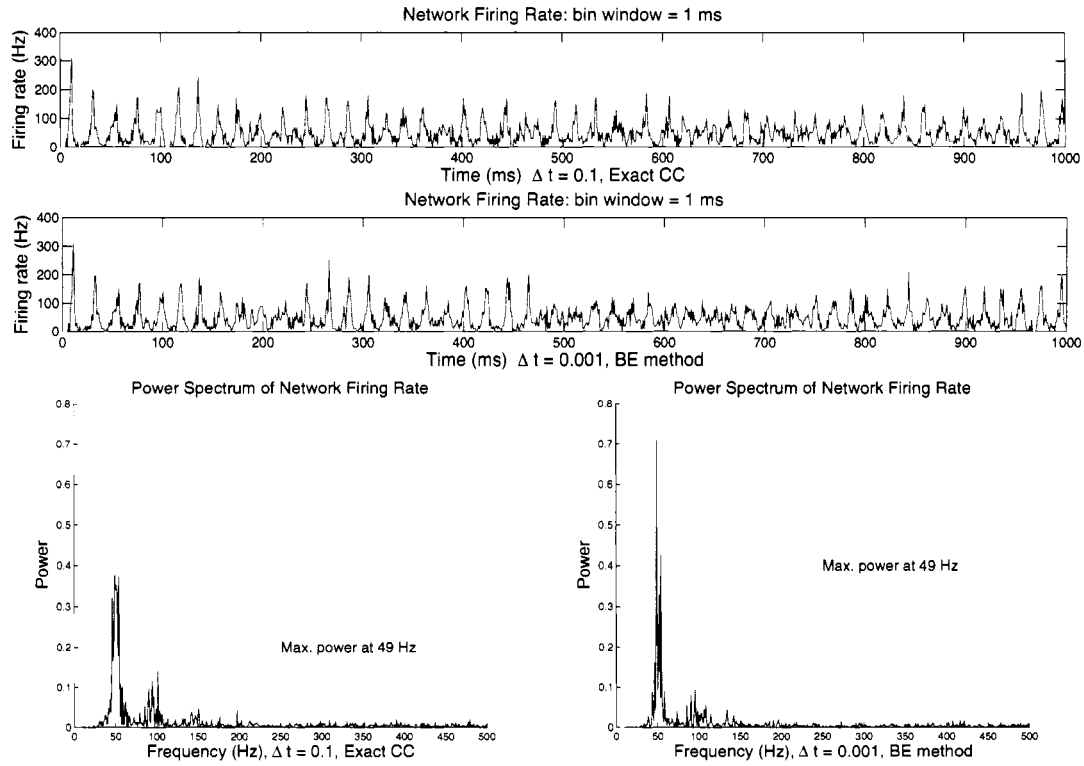
Simulations of large networks (usually more than a thousand cells) with inhibition have shown that populations of neurons as a whole can fire in oscillatory patterns, while single cells fire sparsely (Traub and Miles [25], pp. 164–5). This phenomenon, known as network rhythmogenesis, has been produced in a variety of integrate-and-fire models with alpha function synaptic inputs (Pinsky and Rinzel [20], Brunel and Wang [7]). I present evidence that the quadrature method with the delta-synapse model is an accurate technique for simulating large networks. Possible applications could include using it to efficiently simulate rhythmogenesis once the proper conductance parameter values are known.



**Figure 3.11:** a) Firing diagram of a 100-cell network computed using the quadrature method. Cells are randomly stimulated from a 200-cell EC/DG layer in which each EC/DG cell fires at 500 Hz. The population oscillations are observed as sparse bands of firing that occupy about 10–20 ms time windows. These firings are synchronized, meaning that most cells fire at regular intervals. The feedforward weights are  $F^{AMPA} = 0.2$ , while synaptic weights are  $W^{AMPA} = 0.2$ ,  $W^{NMDA} = 0.002$ ,  $W^{GABA} = 0.8$  (all have units  $\mu\text{S}/\text{mm}^2$ ).

Consider a network of 90 excitatory cells and 10 inhibitory cells, each with 10% connectivity, and whose synapses all have strengths as given in Figure 3.11. A refractory period of  $\tau_{\text{ref}} = 2$  ms is imposed after a cell spikes. The EC/DG layer is

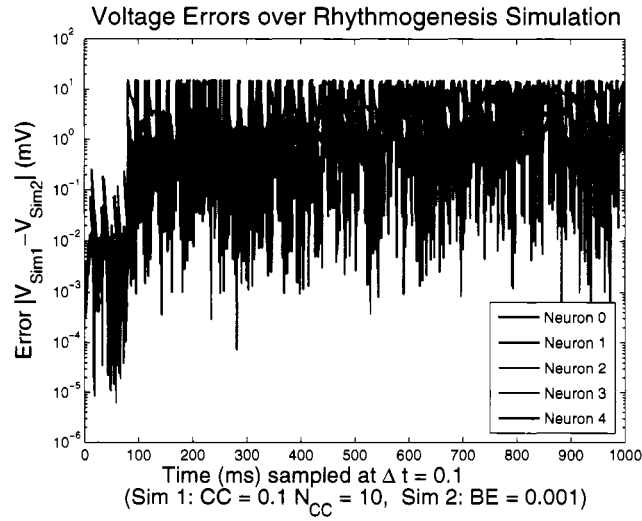
comprised of 200 AMPA-type cells which fire at a rate of 500 Hz. When the EC/DG layer drives this network, synchronized population spiking results (Figure 3.11), akin to that reported by Pinsky and Rinzel for their two-compartment model (Pinsky and Rinzel [20]). However, unlike the PR model, the IAF model requires inhibition to observe network oscillations, and the cells do not demonstrate bursting.



**Figure 3.12:** The top two plots show the firing rate of the network over time for the quadrature and time-stepping methods. The general spiking pattern is captured well, and indeed the power spectra (bottom two plots) associated with the firing rates from each solution method have the same maximum power at 49 Hz, interpreted as the frequency of oscillation of the population.

The simulation is run for both solution techniques, using  $\Delta t = 0.001$  ms for Backward Euler time-stepping and  $\Delta t = 0.1$  ms with  $N_{CC} = 10$  for quadrature. The

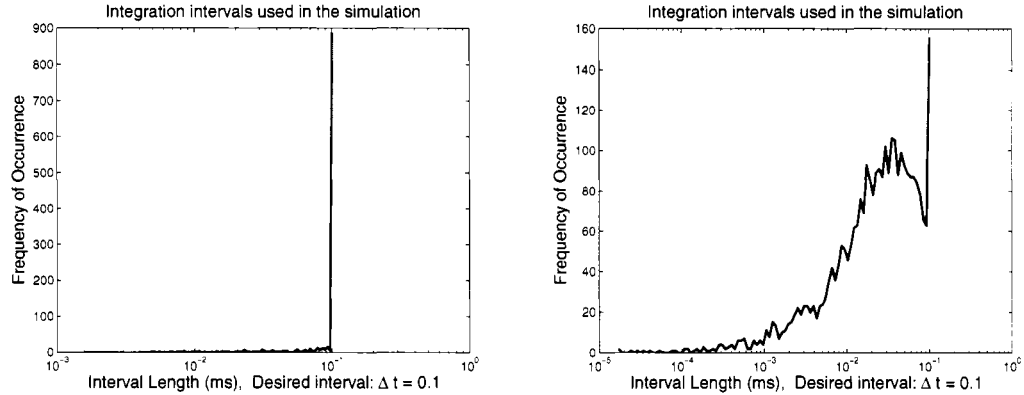
results show that for statistical measures of the network firing, the solution techniques agree. The computed firing rates show that the qualitative behavior of the network as a whole is captured well, and the power spectrum of these rates identify the same oscillatory frequency of 49 Hz (see Figure 3.12).



**Figure 3.13:** Errors in the computed voltages of 5 neurons for the 100-cell oscillatory network. The voltages of five neurons are monitored over the course of the simulation. For the first 150 ms there is 2–3 digit accuracy between the time-stepping and quadrature methods, but then the accuracy decays. However, as evidenced in Figure 3.12, the qualitative behavior of the network is the same despite differing individual voltage trajectories.

Results for the accuracy of individual neurons, however, do not show agreement over large time intervals. The errors in the computed voltages for 5 neurons are shown in Figure 3.13. The accuracy decays over the first 150 ms from 2–3 digits to 0–2 digits for the rest of the simulation, with brief spots of slightly higher accuracy. Such decay suggests that one method is producing more accurate results, but future work will be necessary to assess which one is more accurate.

I hypothesize that the differences result from the manner in which EC/DG spikes are taken into account in the quadrature method. In contrast to Rangan and Cai's approach, which computes the effect of the EC/DG spikes individually (Rangan and Cai [21]), my approach evaluates over subintervals of length  $\Delta t$  and updates conductances with all the EC/DG spikes that fired during this interval. Although this approach introduces error, it speeds up simulations because the algorithm actually takes timesteps of desired length  $\Delta t$  instead of much smaller intervals between EC/DG spikes (see Figure 3.14). Therefore, using my approach, the results from convergence analysis indicate that when EC/DG spikes are evaluated individually the true time step can be significantly smaller than  $\Delta t$ .



**Figure 3.14:** Effective timesteps used in quadrature methods. These two plots show the distributions of timesteps over the course of a simulation with 500 Hz EC/DG input and a desired timestep  $\Delta t = 0.1$  ms. Note the  $x$ - and  $y$ -axis ranges in both plots. a) My approach yields a sharp distribution of timesteps when the EC/DG spikes are included at the end of the interval. b) When the interval of integration is adjusted to integrate between successive EC/DG spikes, the actual timesteps used in the simulation must become smaller. Although (b) will compute voltages more accurately, it also increases simulation time.

### 3.5 Summary of Results

The quadrature method generally outperforms the time-stepping method in resolving accurate voltage trajectories. Using the delta-synapse model instead of the alpha function model reduces the computational complexity of the quadrature method from  $\mathcal{O}(N^2)$  to  $\mathcal{O}(N)$ , thus facilitating the simulation of larger networks. Also, for networks that are not highly active, there is little practical reason to choose interpolation over the Bisection / Secant algorithm for STI for the delta-synapse model.

However, discontinuities inherent in the delta-synapses demand that, for the most accurate solutions, the number of quadrature nodes be very large. The delta-synapse model maintains statistical measures of accuracy, but the question is still open as to whether voltage trajectories for long simulations can be resolved more accurately than when computed via time-stepping. In addition, it is important to use a quadrature rule that takes the endpoints of the integration interval as nodes, otherwise accuracy may be lost.

The IAF neuron models and the numerical methods in this thesis are available in the software package developed here, making it possible to simulate IAF networks in an efficient, accurate, and practical manner.

# Chapter 4

## Future Work

### Synaptic Modification

A natural extension of the ideas in this thesis is to use the quadrature techniques to perform simulations that attempt to mimic cognitive functions. The hippocampus is a critical anatomical structure for behaviors such as temporal compression and trace conditioning, both of which involve memory formation (August and Levy [3], Moyer et al. [19]). These behaviors have been reproduced in simpler McCulloch–Pitts neuronal networks (Levy et al. [17]), and so it is natural to presume that behaviors that are reproducible in less realistic models ought to be reproducible in more realistic models. In order to test this hypothesis the IAF model must have some mechanism for synaptic modification (“learning”). Implementing this feature in the model and in the software will allow for exploration of cognitive functions with confidence in the numerical results.

### Application to Cluster Cutting

The simulation software developed in this thesis can generate test data that may be used to evaluate the performance of cluster cutting algorithms used in hippocampal

recording analysis. Extracellular recordings in the rat hippocampus have shown that certain cells fire when the rat’s head points in a certain direction (Yoganarasimha et al. [30]). The recorded data is obtained by inserting a tetrode, which measures the field potential at four locations, into the hippocampus of a live rat and taking measurements during behavioral experiments. The challenge is to isolate individual cell firings using the recorded data, a process called “cluster cutting”.

Cluster cutting algorithms are in constant development, and evaluating the effectiveness of these algorithms requires many data sets (Schmitzer-Torbert et al. [22]). Instead of using experiments, which are expensive and time-consuming to obtain, using the simulation software in this thesis quickly provides voltage data that can be used to generate surrogate data sets. The advantage of these data sets is that they do not contain noise from other cells. Hence they can be used to determine how accurate the cluster cutting algorithms are when given perfect data.

## **Model Reduction**

Although the hippocampus does not appear to have any spatial topology that can be used to accelerate simulations, the network dynamics may be reproducible in a lower dimensional system. Rhythmogenesis, for example, is a property of the network, not of individual neurons, and hence model reduction techniques may be able to produce the same response without the need for constructing large networks. This is a new but active area of research that has proven fruitful for other brain regions, like the visual cortex (Cai et al. [10]), so it is reasonable to hypothesize that

model reduction could be put to use for the hippocampus.

## **Software Development**

The software platform that I developed currently supports the models and methods used in this thesis, but other models and methods can be incorporated as well. The hippocampus research group at Rice University has worked on models of varying granularity, and it would be advantageous to have all these models available in one common software package.



# Chapter 5

## Conclusion

Quadrature methods are accurate techniques to solve the integrate-and-fire model equations, limited primarily by the size of the network and how active the network is. The efficiency of the quadrature method depends upon how synaptic input is modeled. Using delta-synapses to model synaptic conductances reduces the order of complexity of the quadrature method from  $\mathcal{O}(N)$  from  $\mathcal{O}(N^2)$ , but the number of quadrature nodes required to obtain a desired accuracy scales exponentially for this model.

Network models that use alpha functions to model neurotransmitter release are faster and more accurate than Backward Euler time-stepping for networks of about 50 neurons, while delta-synapse models are faster for up to 1000 neurons. Simulation speed is also governed by the activity of the network. Networks with low rates of firing can be efficiently simulated with quadrature, whereas networks prone to seizure can slow this technique considerably. Interpolation procedures for spike time identification offer little to no advantage for significantly speeding up simulations unless the network activity levels are high. Hence using a more accurate scheme, such as the Bisection /

Secant STI algorithm proposed here, is more practical.

Integrate-and-fire models are deceptively simple, but even after 100 years there is no definitively superior method for solving them. The best approach is to have an assortment of numerical techniques and to know when to apply them to obtain the best performance. This thesis not only identifies the size of networks for which quadrature methods are the preferred solution methods, but additionally introduces simulation software that implements these methods in a stable, accurate, and efficient manner.

## Bibliography

- [1] L. F. Abbott. Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain Research Bulletin*, 50:303–304, 1999.
- [2] G. A. Ascoli, L. Hunter, J. L. Krichmar, J. L. Olds, and S. L. Senft. Computational neuroanatomy of the hippocampus. Accessed March 5, 2007. <http://www.krasnow.gmu.edu/L-Neuron/index.html>.
- [3] D. August and W. Levy. Temporal sequence compression by an integrate-and-fire model of hippocampal area CA3. *Journal of Computational Neuroscience*, 6:71–90, 1999.
- [4] D. A. August. *Sequence Learning by an Integrate-and-Fire Neural Network Model of Hippocampal Area CA3*. PhD thesis, University of Virginia, May 1997.
- [5] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith. PETSc home page. Accessed July 26, 2006. <http://www-unix.mcs.anl.gov/petsc/petsc-as/>.
- [6] A. Bragin, G. Jandó, Z. Nádasdy, J. Hetke, K. Wise, and G. Buzsáki. Gamma (40–100 Hz) oscillation in the hippocampus of the behaving rat. *The Journal of Neuroscience*, 15:47–60, 1995.
- [7] N. Brunel and X.-J. Wang. What determines the frequency of fast network oscillations with irregular neural discharges? I. synaptic dynamics and excitation-inhibition balance. *Journal of Neurophysiology*, 90:415–430, 2003.
- [8] G. Buzsáki, Z. Horváth, R. Urioste, J. Hetke, and K. Wise. High-frequency network oscillation in the hippocampus. *Science*, 256(5059):1025–1027, 1992.
- [9] B. Byrne. Division of neuroscience. Accessed March 27, 2007. <http://www.neuroscience.bham.ac.uk/neurophysiology/research/hippocampus.htm>.
- [10] D. Cai, A. V. Rangan, and D. W. McLaughlin. Neuronal information encoding and reduction of dimension in network dynamics. *SIAM News*, 40(2), March 2007.
- [11] P. J. Davis and P. Rabinowitz. *Methods of Numerical Integration*. Academic Press, Inc., 2nd edition, 1984.
- [12] P. Dayan and L. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. The MIT Press, 2001.

- [13] G. H. Golub. Some modified matrix eigenvalue problems. *SIAM Review*, 15(2): 318–334, April 1973.
- [14] D. Hansel, G. Mato, C. Meunier, and L. Neltner. On numerical simulations of integrate-and-fire neural networks. *Neural Computation*, 10:467–483, 1998.
- [15] A. Hodgkin and A. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117: 500–544, 1952.
- [16] D. Johnston and D. G. Amaral. *The Synaptic Organization of the Brain*, chapter 11, pages 417–458. Oxford University Press, 4th edition, 1998.
- [17] W. B. Levy, A. B. Hocking, and X. Wu. Interpreting hippocampal function as recoding and forecasting. *Neural Networks*, 2005.
- [18] Y.-H. Liu and X.-J. Wang. Spike-frequency adaptation of a generalized leaky integrate-and-fire model neuron. *Journal of Computational Neuroscience*, 10: 25–45, 2001.
- [19] J. R. Moyer, R. A. Deyo, and J. F. Disterhoft. Hippocampectomy disrupts trace eye-blink conditioning in rabbits. *Behavioral Neuroscience*, 104(2):243–252, 1990.
- [20] P. F. Pinsky and J. Rinzel. Intrinsic and network rhythmogenesis in a reduced Traub model for CA3 neurons. *Journal of Computational Neuroscience*, 1:39–60, 1994.
- [21] A. V. Rangan and D. Cai. Fast numerical methods for simulating large-scale integrate-and-fire neuronal networks. *Journal of Computational Neuroscience*, 22:81–100, 2007.
- [22] N. Schmitzer-Torbert, J. Jackson, D. Henze, K. Harris, and A. D. Redish. Quantitative measures of cluster quality for use in extracellular recordings. *Neuroscience*, 131:1–11, 2005.
- [23] M. J. Shelley and L. Tao. Efficient and accurate time-stepping schemes for integrate-and-fire neuronal networks. *Journal of Computational Neuroscience*, 11:111–119, 2001.
- [24] E. Süli and D. Mayers. *An Introduction to Numerical Analysis*. Cambridge University Press, 2003.
- [25] R. D. Traub and R. Miles. *Neuronal Networks of the Hippocampus*. Cambridge University Press, 1991.
- [26] L. N. Trefethen. Is Gauss quadrature better than Clenshaw–Curtis? *SIAM Review*, 2007.

- [27] L. N. Trefethen. *Spectral Methods in MATLAB*. SIAM, 2000.
- [28] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, 1997.
- [29] G. von Winckel. `lglnodes.m`. Accessed online April 3, 2007. <http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=4775&objectType=File>, 2004.
- [30] D. Yoganarasimha, X. Yu, and J. J. Knierim. Head direction cell representations maintain internal coherence during conflicting proximal and distal cue rotations: Comparison with hippocampal place cells. *The Journal of Neuroscience*, 26(2): 622–631, January 2006.