Characterization of an advanced neuron model

Christopher Echanique
University of Central Florida
CHARACTERIZATION OF AN ADVANCED NEURON MODEL

by

CHRISTOPHER A. ECHANIQUE

A thesis submitted in partial fulfillment of the requirements for the Honors in the Major Program in Computer Engineering in the College of Engineering and Computer Sciences and in The Burnett Honors College at the University of Central Florida Orlando, Florida

Summer Term 2012

Thesis Chair: Dr. Aman Behal
ABSTRACT

This thesis focuses on an adaptive quadratic spiking model of a motoneuron that is both versatile in its ability to represent a range of experimentally observed neuronal firing patterns as well as computationally efficient for large network simulation. The objective of research is to fit membrane voltage data to the model using a parameter estimation approach involving simulated annealing. By manipulating the system dynamics of the model, a realizable model with linear parameterization (LP) can be obtained to simplify the estimation process. With a persistently excited current input applied to the model, simulated annealing is used to efficiently determine the best model parameters that minimize the square error function between the membrane voltage reference data and data generated by the LP model. Results obtained through simulation of this approach show feasibility to predict a range of different neuron firing patterns.
ACKNOWLEDGMENTS

I express sincere thanks and gratitude to my committee members, who have been gracious enough to facilitate this project with their wisdom and experience. Special thanks go to my thesis chair, Dr. Aman Behal, for his guidance and support. Thanks also to my graduate mentor, Lingfei Zhi, without whom I might have never gained the knowledge and experience to program and implement the complex neuron model presented.

Partial support for this work was provided by the National Science Foundation Scholarships in Science, Technology, Engineering, and Mathematics (S-STEM) program under Award No. 0806931 as well as the National Institute of Neurological Disorders and Stroke grant (#R15NS062402). Any opinions, findings, and conclusions and recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or the National Institute of Neurological Disorders. Partial support for this work was also provided by the UCF Office of Research and Commercialization.
# TABLE OF CONTENTS

INTRODUCTION .................................................................................................................. 1

LITERATURE REVIEW ............................................................................................................. 3

Model ........................................................................................................................................ 5

Parameters ................................................................................................................................ 5

LP Model .................................................................................................................................. 6

Simulated Annealing ............................................................................................................... 8

EXPERIMENTAL DESIGN ....................................................................................................... 10

Identification ............................................................................................................................. 10

Optimization .............................................................................................................................. 13

Validation ................................................................................................................................. 15

RESULTS .................................................................................................................................. 17

Regular Spiking Neurons ........................................................................................................ 17

Intrinsically Bursting Neurons ................................................................................................. 19

Chattering Neurons ................................................................................................................ 21

Rapidly Adapting Neurons ..................................................................................................... 23

DISCUSSION ............................................................................................................................ 26

CONCLUSION .......................................................................................................................... 28

APPENDIX ............................................................................................................................... 30

MATLAB Implemented Code ................................................................................................ 30

REFERENCES .......................................................................................................................... 41
INTRODUCTION

The generation of action potentials from voltage dependent ionic channels in neurons has been a widely researched phenomenon. A fundamental issue in computational neuroscience is to characterize the relationship between the neural output recording and the input current to the cell [1]. Over the past century, dozens of models have been introduced to describe this neural relationship. After an extensive survey of such models, the adaptive quadratic spiking model of a neuron proposed by Eugene Izhikevich [3] has been chosen as the focus of research. This model is both computationally efficient for large network simulation and versatile in its ability to represent a range of experimentally observed neuronal firing patterns.

One of the limitations reported in the literature is the ability to systematically and efficiently fit membrane voltage data to neuron models. Identification of large scale neuronal networks using such models is a major stepping stone in understanding the details of important biological systems and can eventually result in novel tools for the early detection of neuromuscular diseases. By employing a parameter estimation process using a set of reference data, an optimal set of model parameters can be found that characterize this given reference data to a particular type of neuron spiking behavior. After a survey of various parameter estimation techniques, simulated annealing has been chosen as the most effective and efficient optimization algorithm to find a global solution for nonlinear systems dynamics, such as those in the Izhikevich model. Therefore it has been implemented into the estimation process to find the most optimal model parameters, which are associated with spiking and sub-threshold
dynamics of the system as well as the threshold and reset parameters of the neuronal spiking
model. The presented model and estimation method have been employed in the data
computing environments MATLAB and SIMULINK. These methods have been tested using
membrane voltage data from the model itself for validation.

The organization of this paper is as follows: In section 2 we present a literature review
addressing some of the issues and limitations of current research on the subject. In section 3
we define the Izhikevich model and parameters, as well as identify the parameter estimation
approach, simulated annealing, and define the function and its parameters. In Section 4, we
describe the experiments and the datasets used to assess the performance of the simulated
annealing parameter estimation method. Results of the parameter estimation approach are
presented in Section 5. Section 6 is an assessment the performance of this method. In Section
7, we summarize our findings and provide directions for future research.
LITERATURE REVIEW

Over the past century, dozens of models have been introduced to characterize the relationship between the neural output recording and the input current to the cell. Among the most widely known is the Hodgkin-Huxley model presented in 1952, a Nobel Prize winning model to describe the initiation and propagation of action potentials in neurons, particularly giant squid axons [2]. This four-dimensional dynamical model, both versatile and accurate when reproducing most types of neuronal behavior, has been the foundation of many subsequent models for its biological plausibility. However, it has been deemed computationally inefficient for large scale networks due to the hundreds of parameters associated with it.

Due to the growing interesting in modeling larger scale neural networks as opposed to single neurons, many scientists are not concerned with biological plausibility and cannot afford the costly computations associated with conductance-based Hodgkin-Huxley type models. At times, a simple model that can reproduce most of the neurocomputational features of a neuron is sufficient [3]. One such model is the notable leaky integrate and fire model [4] defined by a linear differential equation and reset criteria for state variables when the membrane potential crosses a threshold. Though not technically a spiking model (when the threshold is reached, it is only ‘said’ to fire a spike), this simple model is efficient to compute, leaving it a popular choice for spiking network simulations and to create adaptations from. Yet its major drawback lies in its limitation to produce only a few types of firing patterns of neurons, categorizing it as a polar opposite to the Hodgkin-Huxley model on the neuronal model spectrum.
Recently neuron models have been extensively researched and modified to bridge this gap and achieve both versatility and computational efficiency. One such model has been proposed by Izhikevich [5] and is the primary focus of our research. A recovery variable intended to account for the activation of K+ ionic currents and inactivation of Na+ ionic currents is introduced to this new adaptation of the integrate-and-fire model. The resulting adaptive quadratic spiking model can exhibit characteristics of all known types of cortical neurons. And with a comparatively low number of operations to simulate a short interval of data [6], this model proves to be a prime contender for versatility and computational efficiency among neuronal models.

Although this spiking model and its application have been discussed in great length, a systematic technique to accurately estimate parameters of experimental data for reproduction and analysis is still needed. There are two approaches to fitting model parameters to data [7]. One can hand tune these model parameters to fit the data and produce the desired behavior, as in [8], where an adaptive exponential integrate-and-fire model was tuned to fit a Hodgkin-Huxley based model. While this method may yield good results, it relies heavily on the expertise of the researcher and is labor-intensive when trying to fit different types of data. Thus the second more practical approach involving automatic parameter estimation processes is necessary.
DEFINING MODELS, PARAMETERS, AND OPTIMIZATION

Model

A simple adaptive quadratic spiking model [6] can be described by the following state equations below

\[ C \dot{v} = k(v - v_r)(v - v_t) - u + I \]  

(1)

\[ \dot{u} = a[b(v - v_r) - u] \]  

(2)

with post-spike resetting criteria

\[ \text{if } v \geq V_p, \text{ then } \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \]  

(3)

Here \( v \) and \( u \) are dimensionless state variables, representing the membrane potential and membrane recovery, respectively. The input variable \( i \) corresponds to the injected and/or synaptic current that affect the system dynamics. The resetting criteria ensure that the state variables are reset appropriately after the spike reaches \( V_p \).

Parameters

The unknown parameters that need to be estimated are defined by the vector

\[ \theta_0 = (C, k, v_r, v_t, a, b, c, d) \]  

(4)

where \( C \) is the membrane capacitance, \( v_r \) represents the voltage resting potential, \( v_t \) represents the instantaneous threshold potential, \( k \) denotes the spike initiation and subthreshold
dynamics of the system, \(a\) is the recovery time constant, \(b\) is the level of subthreshold adaptation, and \(c\) and \(d\) are the post spiking and spike adaptation of the recovery variable, respectively.

**LP Model**

A linearly parameterized (LP) realizable model can be obtained from Izhikevich’s simple model with a manipulation of the system dynamics [9]. To do this, the resetting discontinuities given in (3) must first be integrated into the state equations (1) and (2). Since the membrane potential is reset to \(c\) once it has reached \(V_p\), this can be rewritten into the model in the form of a step input with a jump size of \(c - V_p\) as follows

\[
v \rightarrow v + (c - V_p)s(t - t_{sj})
\]

where \(s(t-t_{sj})\) denotes a unit step at the \(j^{th}\) spike at time \(t_{sj}\) [9]. Since this is valid for all spiking instants, a summation of the time derivative of (3) can be incorporated into the model state equation (1) as follows

\[
C\dot{v} = k(v - v_r)(v - v_l) - u + I + (c - V_p) \sum_j \delta(t - t_{sj})
\]

The same concept can be applied to the reset discontinuity in \(u\), which is rewritten as a step input with a jump size of \(d\).

\[
u \rightarrow u + ds(t - t_{sj})
\]
Incorporating the time derivative of this with a summation over all spike times into the state equation, we have

\[ \dot{u} = a[b(v - v_r) - u] + d \sum_j \delta(t - t_{sj}) \quad (8) \]

The newly formed state equations (6) and (8) are now integrated with the post-spike reset criteria. In order to obtain a linearly parameterized model from the nonlinear dynamics of the state equations, the Laplace transformation of both equations (6) and (8) is required. To avoid dependency on the derivatives of the input and output, a low pass filter is applied [9] in the following form.

\[ \frac{1}{A} = \frac{1}{s^2 + \beta_1 s + \beta_0} \quad (9) \]

After substituting the Laplace transform of (8) into (6) and applying the aforementioned filter, the following expression is obtained

\[ V = \frac{k(s+a)L(V^2)}{A} + \frac{1}{A} + \frac{k(v_f + v_t)(s+a) + s(\beta_1 s + \beta_0 - ab \delta)}{A} \]

\[ + \frac{(c-v_p)(s+a) - d}{A} \sum_j \exp(-st_{sj}) + \frac{(s+a)v(0)}{A} + \frac{k_v v_f + ab v_r}{c} - \frac{1}{A} \]

where \( V \) and \( I \) represent the Laplace transform of the \( v \) and \( i \) respectively, \( L(\cdot) \) is the Laplace operator, and \( A \) is the low pass filter. Since the last three terms of the expression do not persist beyond an initial transient, they can be disregarded from future analysis [9]. This linearly
parameterized realizable model can be expressed in the form of the product of two vectors as follows

\[ v = W(v, i, t_{sj})\theta(\theta_0) \tag{11} \]

where \( W(v, i, t_{sj}) \) is a realizable regression vector given as

\[ W = \left[ \frac{s}{A} L(v^2), \frac{1}{A} L(v^2), \frac{s}{A} V^3, \frac{1}{sA}, \frac{s}{A} \frac{1}{A} s \frac{1}{A} \frac{1}{A} \frac{1}{A} \frac{1}{A} \sum \exp(-st_{sj}), \frac{1}{A} \sum \exp(-st_{sj}) \right] \tag{12} \]

and \( \theta \) is an unknown parameter vector defined as function of \( \theta_0 \) as follows

\[
\theta = \begin{bmatrix}
\frac{k}{c}, \frac{ka}{c}, -\frac{k}{c} (v_r + v_t) + \beta_1 - a, -\frac{ka}{c} (v_r + v_t) + \beta_0 - \frac{ab}{c}, \\
\frac{kv_r}{c}, a - abv_r, \frac{1}{c}, \frac{a}{c}, c - V_p, a(c - V_p) - \frac{d^T}{c}
\end{bmatrix} \tag{13}
\]

**Simulated Annealing**

Annealing is a process used in material sciences to improve the properties of a material by heating and cooling it [11]. Inspired by this natural phenomenon, simulated annealing uses this process while performing a parameter search method. Initially, the large degrees of freedom are used to explore a solution space. Then, a cooling process begins to reduce the chances of escaping an optimal solution.

Using this method, a search temperature \( T \) and a candidate solution \( x \) are defined. With each iteration, the solution \( x \) may be replaced with a more optimal solution \( y \) based on probability. This is calculated using a Boltzmann-Gibbs distribution [11]:
where $f$ is the error function, $T$ represents the temperature, and $c$ is a positive constant. The way the temperature changes with each iteration can be defined in several ways to alter the rate of cooling and obtain the most optimal solution.

The function to be minimized using this simulated annealing algorithm is defined by the error function below:

$$E = \sqrt{(\hat{\theta} - \nu)^2} = \sqrt{W\theta - \nu)^2}$$

where $E$ represents the error between the reference data $\nu$ and the product of the realizable regression vector $W$ times the unknown parameter vector $\theta$. Given upper and lower bounds for each parameter in $\theta_0$, simulated annealing uses an iterative stochastic search procedure to select values within the parameter search space, attempting to find the global minimum.
EXPERIMENTAL DESIGN

Two different types of reference data are used for parameter estimation. The first type is membrane voltage data obtained from the Izhikevich model used to test the validity of the approach [9]. Then a noisy version of this model-based membrane voltage is used in order to test the method’s robustness. These types of reference data are applied to parameter estimation approach, which can be broken down into three stages: Identification, Optimization, and Validation. The following sections present pseudocode and provide explanations of the steps to each stage. The actual code implemented in MATLAB can be found in the Appendix.

Identification

The parameter estimation process begins with identification. Here, the applied current, membrane potential, and spike times are all identified, and the regression vector W is realized.

IDENTIFICATION
===============
1: Generate Reference Data
   1a: Injected Current
   1b: Membrane Voltage
   1c: Spike Instants and Voltage Peak (Vp)
2: Obtain W Regression Vector
3: Remove Initial Transient

Step 1: Generate Reference Data

Based on the LP model derived in (11), three inputs are required to obtain the W regression vector: current, membrane potential, and spike instants.
Sub-Step 1a: Injected Current

A prerequisite for the parameters of LP model to converge to their true values is that the system must be persistently excited. Since up to two parameters of the model can be estimated with a reference containing one sinusoid [9], four sinusoidal frequencies are necessary for the eight parameters of $\theta_0$. The current $i$ applied to the system can be defined in the following form:

$$i = I_1 \sin(\omega_1 t) + I_2 \sin(\omega_2 t) + I_3 \sin(\omega_3 t) + I_4 \sin(\omega_4 t) + s$$

(15)

where each of the four sets of $I$ and $\omega$ represent different amplitudes and frequencies of sinusoids respectively and $s$ denotes a constant step. The values for these sinusoids are selected to induce sufficient spike trains for estimation and depend on the type of neuron used.

Sub-Step 1b: Membrane Voltage

As mentioned before, two types of membrane voltage reference data are used. The first is obtained directly from the Izhikevich model itself. Here, parameters in theta (4) are chosen and current (15) is injected into the system to generate the membrane voltage data, which is implemented in MATLAB and Simulink environments. Such parameters are selected to replicate the behaviors of four types of neurons: regular spiking, intrinsically bursting, chattering, and rapidly adapting neurons [3]. This is to demonstrate the capability of the estimation approach to generate results for a wide range of neuron types.
Next, this same approach is taken to obtain the voltage data from the model. Four neurons types are still used in the estimation process. However, noise is added to this reference data by passing it through a filter with a signal-to-noise ratio of 40 dB. This is used in order to test the robustness of the approach.

*Sub-Step 1c: Spike Instants and Voltage Peak (V_p)*

After the membrane voltage is generated, the last prerequisites for the W vector are the voltage peak (V_p) and spike instants. Since the Izhikevich model requires V_p as an input, the voltage peak is known for any data generated from the model itself. Otherwise, V_p can be acquired simply by finding the maximum value of the given voltage data. Once found, V_p is then used to determine the instants at which spikes occur. An algorithm is employed which iterates through the data, forming an increasing step function at times when the membrane voltages reaches V_p. The MATLAB code for this can be seen in the Appendix as well.

*Step 2: Obtain W Regression Vector*

Once the current, voltage, and spike instants are generated, the realizable regression vector W is ready to be obtained. An implementation of (12) can be realized in the Simulink environment to produce the nine components of the regression vector.

*Step 3: Remove Initial Transient*

As noted previously, some of the terms acquired in the LP model (10) were dropped since they do not persist beyond an initial transient. To simplify the process and because the
goal of this parameter estimation approach is to replicate general spiking patterns and not initial neuronal behavior, these terms were not implemented. However their lack of presence in the derived LP model causes discrepancies at the beginning of the membrane voltage trains. To compensate for this, the reference data is clipped several milliseconds at the start in order to avoid complications during the estimation process.

**Optimization**

The second stage of the process involves utilizing the simulated annealing algorithm to optimize the error function and obtain the estimated parameters of $\theta_0$ (4). The parameters of the simulated annealing function, such as the objective function, boundary constraints, initial point, temperature, etc., must first be defined before optimization can occur. The Optimization Toolbox provided in the MATLAB environment conveniently offers a graphical user interface where these parameters can be entered. The steps of this stage are listed below:

<table>
<thead>
<tr>
<th>OPTIMIZATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>=============</td>
</tr>
<tr>
<td>1: Define Objective Function</td>
</tr>
<tr>
<td>2: Select Theta Boundary Constraints</td>
</tr>
<tr>
<td>&gt; Identify the resting membrane potential</td>
</tr>
<tr>
<td>3: Run Simulated Annealing</td>
</tr>
</tbody>
</table>
Step 1: Define Objective Function

Simulated annealing takes in an objective function \( f(x_1...x_n) \) attempting to search for values of \( x_1...x_n \) that minimize the function’s output. In this case, the objective function is defined as the error between the reference data \( v \) and the product of the regression vector \( W \) and the unknown parameter set \( \theta \) \((13)\). The MATLAB implementation of this function is given in the Appendix.

Step 2: Identify Theta Boundary Constraints

To prevent simulated annealing from selecting parameters in an infinite search space, upper and lower boundary constraints can be applied to each parameter being estimated. Since the parameters of the Izhikevich model represent meaningful and some easily attainable components of the membrane voltage it produces, boundaries can be assigned to a parameter with prior knowledge of its functionality. For example, parameters \( C, k, \) and \( \alpha \) in \((4)\) all must be greater than zero for the model to produce a valid output \([3]\). Some parameters, such as the resting membrane potential \( v_r \) and the post-spike reset \( c \), can be even roughly estimated by analyzing the behavior of the system. The resting membrane potential is simply the value at which the voltage converges to in the absence of current in the system. To find this, membrane voltage data with the same parameters as the original reference data being used for estimation is regenerated with zero current injected. It can be seen that the voltage will rapidly converge to a value after roughly a few seconds of data. This value is used as an estimate of the actual value for \( v_r \). Likewise, an estimate of the post-spike reset \( c \) can be obtained by finding the
average value at which the voltage falls to after reaching the voltage peak $V_p$. Both of these parameters, $v_r$ and $c$, demonstrate properties of the system that are precise and stable. Therefore, tighter boundaries can be applied to them once a close estimate is found.

**Step 3: Run Simulated Annealing**

Once the parameters of the simulated annealing function are defined, optimization can be executed to explore the parameter search space stochastically and find the eight parameters in $\theta_0$ that provide the most optimal solution with minimum error in the objective function. These estimated parameters are then used for validation, the last stage of the process.

**Validation**

The final stage of the process is validation. Here, the set of parameters estimated in the previous stage along with constant step or sinusoidal current injections is applied to the Izhikevich model to compare against the reference system.

```
VALIDATION
==========
1: Generate Voltage with Estimated Parameters
   > Step Current
   > Apply to Model
2: Compare Target vs. Estimated Data
```
Step 1: Generate Voltage with Estimated Parameters

After the optimization stage is complete, the estimated set of parameters $\hat{\theta}$ is ready to be applied to the model. The efficacy of the system identification is validated by injecting either a constant step or sinusoidal current into both the reference system (a) and the Izhikevich model with $\hat{\theta}$ (b) [9]. The injected current, whether step or sinusoids, is chosen so as to produce sufficient spiking for comparison.

Step 2: Compare Target vs. Predicted Data

The goal of system identification is to reproduce the spiking behavior of the reference system. Because of this, comparisons drawn between the resulting spiking patterns (a) and (b) are both quantitative and qualitative. A quantitative emphasis is placed on the percentage error of spikes between target and predicted data. While this error can change based on the strength of the injected current, it gives a relatively strong indicator of the accuracy of the system identification. Also, for the reference data that was generated directly from the Izhikevich model, comparisons can be made by finding the error between each parameter used for reference and the parameters that have been estimated. Qualitative comparisons also provide a strong validation of the predicted data. Should the reference data exhibit distinct neuronal characteristics (i.e. bursting, rapid adaptation, chattering, etc.) these must also be evident in the prediction.
RESULTS

Simulation results are presented for each type of reference data used: the Izhikevich model reference data and the same Izhikevich model data with noise added (signal to noise ratio of 40 dB). Since the noisy reference data for each of the four neurons uses the same parameters and injected current as the original model data, they are grouped into the section with the corresponding neuron. All reference data is generated with a time of two seconds at a sampling rate of .02 points per second. Because of the stochastic nature of the simulated annealing process, simulations are run ten times at a maximum of 100,000 function evaluations for each neuron and the parameter set resulting in the least amount of error from the error function is presented.

Four sets of model parameters are used to demonstrate the feasibility of this approach to produce results for a wide range of neuron types, which include: (1) regular spiking, (2) intrinsically bursting, (3) chattering, and (4) rapidly adapting neurons.

Regular Spiking Neurons

Regular spiking neurons are neurons that can generate spikes with arbitrarily low frequency, depending on the strength of the applied current [5]. They can be characterized by the following parameter set:

$$\theta_0 = (100, .7, -60, -40, .03, -2, -50, 100)$$

For reference data generation, the following parameters are used for the applied current:
\[ I_1 = 30 \quad I_2 = 100 \quad I_3 = 70 \quad I_4 = 80 \]
\[ \omega_1 = 0.05 \quad \omega_2 = 0.3 \quad \omega_3 = 0.2 \quad \omega_4 = 0.5 \]

where the unit of current is expressed in \text{pA} and the frequencies are in \text{rad/ms}. A step of 100 pA is also added to this sinusoidal current to increase the amount of spikes produced.

\[ \theta_0 \quad \hat{\theta} \quad \text{Error (%)} \]
\begin{tabular}{|c|c|c|}
\hline
C & 100 & 98.4428 & 1.56 \% \\
\hline
k & 0.7 & 0.6898 & 1.46 \% \\
\hline
\text{Vr} & -60 & -59.9651 & 0.06 \% \\
\hline
\text{Vt} & -40 & -40.3468 & 0.87 \% \\
\hline
a & 0.03 & 0.0214 & 28.73 \% \\
\hline
b & -2 & -1.1282 & 43.59 \% \\
\hline
c & -50 & -49.9319 & 0.14 \% \\
\hline
d & 100 & 65.4626 & 34.54 \% \\
\hline
\end{tabular}

\textbf{Figure 1} – Voltage traces of prediction and target data for a regular spiking neuron are shown in the graph. Target parameters \( \theta_0 \) and estimated parameters \( \hat{\theta} \) are given in the table beside, along with the percentage error.

The estimated parameters \( \hat{\theta} \) obtained through optimization are given in the table in Figure 1. The graph displays the spike trains achieved during validation with a step current of 100 pA. Here the prediction demonstrates almost identical spiking behavior as the target data, with a 0\% error in the number of spikes for one second of data. It is important to note the percentage error between the target and estimated parameters. Estimated parameters \( C, k, v_r, v_o, \) and \( c \) fall exceptionally close to their target values (less than 2\% error), while parameters \( a, b, \) and \( d \) exhibit high error.
The same reference data and applied current is used for the next set of results. However white Gaussian noise is added to the system with a signal-to-noise ratio of 40 dB. Both optimization and validation are performed as before and the results are given in Figure 2.

![Graph](image)

**Figure 2** – Voltage traces of prediction and target data for a regular spiking neuron with white Gaussian noise are presented in the graph. The table beside lists the target parameters $\theta_0$ and estimated parameters $\hat{\theta}$ along with the percentage error.

<table>
<thead>
<tr>
<th></th>
<th>$\theta_0$</th>
<th>$\hat{\theta}$</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>100</td>
<td>99.0348</td>
<td>0.97 %</td>
</tr>
<tr>
<td>k</td>
<td>0.7</td>
<td>0.6967</td>
<td>0.47 %</td>
</tr>
<tr>
<td>Vr</td>
<td>-60</td>
<td>-60.0841</td>
<td>0.14 %</td>
</tr>
<tr>
<td>Vt</td>
<td>-40</td>
<td>-39.4714</td>
<td>1.32 %</td>
</tr>
<tr>
<td>a</td>
<td>0.03</td>
<td>0.0406</td>
<td>35.18 %</td>
</tr>
<tr>
<td>b</td>
<td>-2</td>
<td>-1.5432</td>
<td>22.84 %</td>
</tr>
<tr>
<td>c</td>
<td>-50</td>
<td>-49.9307</td>
<td>0.17 %</td>
</tr>
<tr>
<td>d</td>
<td>100</td>
<td>103.2008</td>
<td>3.20 %</td>
</tr>
</tbody>
</table>

Again, the predicted data generated by the estimated parameters accurately replicates the spiking behavior of the reference system, even in the presence of noise. The prediction overshot the target spike frequency with a spiking error of 8.33%. Also most of the parameters estimated fell close to their respective target values, again with the exception of $a$ and $b$.

**Intrinsically Bursting Neurons**

Intrinsically bursting neurons are characterized by an initial burst of high frequency spikes followed by a low frequency tonic spiking [5]. These typically have a lower firing frequency than the regular spiking neurons, and also have higher rheobase current, or the
threshold current that produces an action potential. To generate this neuron type, the following parameter set is used:

\[ \theta_0 = (150, 1.2, -75, -45, .03, -2, -50, 100) \]

Since the rheobase current is higher for this type of neuron, the step of the current is increased to 400 pA and the amplitudes and frequencies are given below:

\[
\begin{align*}
I_1 &= 30 \\
I_2 &= 100 \\
I_3 &= 70 \\
I_4 &= 80 \\
\omega_1 &= 0.05 \\
\omega_2 &= 0.3 \\
\omega_3 &= 0.2 \\
\omega_4 &= 0.5
\end{align*}
\]

**Figure 3** – Voltage traces of prediction and target data for an intrinsically bursting neuron are shown in the graph. Target parameters \( \theta_0 \) and estimated parameters \( \hat{\theta} \) are given in the table beside, along with the percentage error.

In Figure 3, the graph displays the spike trains achieved during validation, this time with a step current of 400 pA. The characteristics of intrinsically bursting neurons, including initial bursting, are successfully replicated in the predication with a spike error of 0%. Estimated values for parameters \( C, k, v_r, v_t, \) and \( c \) (all associated with the state variable \( v \)) have low error while parameters \( a, b, \) and \( d \) are significantly off from their true values.
Again, the same reference data and applied current is used for generate reference data where white Gaussian noise with a signal-to-noise ratio of 40 dB is added. Optimization and validation results are given in Figure 4 below.

\[
\begin{array}{|c|c|c|}
\hline
\theta_0 & \hat{\theta} & \text{Error (\%)} \\
\hline
C & 150 & 150.5001 & 0.33 \% \\
k & 1.2 & 1.2009 & 0.08 \% \\
V_r & -75 & -75.1813 & 0.24 \% \\
V_t & -45 & -45.0007 & 0.00 \% \\
a & 0.01 & 0.0077 & 23.38 \% \\
b & 5 & 5.3085 & 6.17 \% \\
c & -56 & -56.0158 & 0.03 \% \\
d & 130 & 135.0967 & 3.92 \% \\
\hline
\end{array}
\]

**Figure 4** – Voltage traces of prediction and target data for an intrinsically bursting neuron with white Gaussian noise are presented. The table beside lists the target parameters $\theta_0$ and estimated parameters $\hat{\theta}$ along with the percentage error.

The predicted data shown in the graph Figure 4 still replicates the spiking characteristics of the reference system, even in the presence of noise. However, the prediction is slightly lower than the target spike frequency with a spiking error of 20%. Most of the parameters estimated fall close to their respective target values, but the large discrepancy in parameter $a$ associated with the adaptation may have caused the significant error in spike frequency.

**Chattering Neurons**

Chattering neurons, also known as fast rhythmic bursting, exhibit high frequency bursts of action potentials followed by a relatively short interspike interval [3]. The model parameters
defining this neuron type are as follows:

\[ \theta_0 = (50, 1.5, -60, -40, .03, 1, -40, 150) \]

The injected current is set to 200 pA and the amplitudes and frequencies are given below:

I_1 = 30 \quad I_2 = 100 \quad I_3 = 70 \quad I_4 = 80
\omega_1 = 0.05 \quad \omega_2 = 0.3 \quad \omega_3 = 0.2 \quad \omega_4 = 0.5

Figure 5 – Voltage traces of prediction and target data for a chattering neuron are shown in the graph. Target parameters \( \theta_0 \) and estimated parameters \( \hat{\theta} \) are given in the table beside, along with the percentage error.

The validation results shown in the graph in Figure 5 demonstrate the fast rhythmic bursting behavior of chattering neurons. While the estimated parameter values fell much closer to the target than the results for the previous neurons, the spike frequency in the prediction is slightly lower than the target, with a spike error of 9.1%.

After applying white Gaussian noise with a signal-to-noise ratio of 40 dB to this same reference data using the same current, the following results are obtained which are given in Figure 6 on the next page.
Figure 6 – Voltage traces of prediction and target data for a chattering neuron with white Gaussian noise are presented in the graph. The table beside lists the target parameters $\theta_0$ and estimated parameters $\hat{\theta}$ along with the percentage error.

Here the predicted data in Figure 6 still replicates the chattering characteristics of the reference system. The prediction is almost perfectly in sync with the spiking patterns of the target reference data with each of the 22 spikes represented in the one second interval and a 0% spike error.

Rapidly Adapting Neurons

Rapidly adapting neurons are characterized by quick adaptation to injected current, which eventually inhibits action potentials. These have a much higher firing frequency than most neurons along with very low rheobase current and shorter interspike intervals. Rapidly adapting neurons can be represented by the following parameter set:

$$\theta_0 = (1, 0.04, -82.65, -42.34, .02, .2, -65, -.5)$$
Since the rheobase current is much lower for this type of neuron, a step of 18 pA is added to the injected current, along with the following amplitudes and frequencies:

\[
\begin{align*}
I_1 &= 2 & I_2 &= 7 & I_3 &= 4.5 & I_4 &= 8 \\
\omega_1 &= 0.5 & \omega_2 &= 2.25 & \omega_3 &= 2 & \omega_4 &= 2.5
\end{align*}
\]

<table>
<thead>
<tr>
<th>(\theta_0)</th>
<th>(\hat{\theta})</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1</td>
<td>0.9987</td>
</tr>
<tr>
<td>k</td>
<td>0.04</td>
<td>0.0397</td>
</tr>
<tr>
<td>Vr</td>
<td>-82.65</td>
<td>-82.7332</td>
</tr>
<tr>
<td>Vt</td>
<td>-42.34</td>
<td>-42.5943</td>
</tr>
<tr>
<td>a</td>
<td>0.02</td>
<td>0.0213</td>
</tr>
<tr>
<td>b</td>
<td>0.2</td>
<td>0.1849</td>
</tr>
<tr>
<td>c</td>
<td>-65</td>
<td>-64.9515</td>
</tr>
<tr>
<td>d</td>
<td>-0.5</td>
<td>-0.4785</td>
</tr>
</tbody>
</table>

Figure 7 – Voltage traces of prediction and target data for a rapidly adaptive neuron are shown in the graph. Target parameters \(\theta_0\) and estimated parameters \(\hat{\theta}\) are given in the table beside, along with the percentage error.

For the validation stage, a constant step current of 18 pA is used. The prediction here accurately replicates the adapting neuronal behavior of the reference system, with high frequency initial spikes and eventual spike inhibition. With a spike error of 7.8%, the number of spikes in the target is slightly underrepresented by the prediction. It is important to note that although the percentage error is relatively low for all parameter values, parameters \(a, b,\) and \(d\) show considerably high error from their true values when compared to that of the other five.

Once again, white Gaussian noise is applied to the same set of reference data applied current is used for generate reference data and the results are given in Figure 8 on the next page.
Figure 8 – Voltage traces of prediction and target data for a rapidly adapting neuron with white Gaussian noise are presented in the graph. The table beside lists the target parameters $\theta_0$ and estimated parameters $\hat{\theta}$ along with the percentage error.

The predicted data in Figure 8 exhibits the rapidly adapting behavior of the neuron and almost exactly matches up with the target for each spike. The error in the number of spikes is 0%, once again despite the discrepancies in the parameters values for $a$, $b$, and $d$. The results for these neurons are further discussed in detail in the next section.
DISCUSSION

The results for the prediction of reference data from the Izhikevich model indicate a good prediction for replicating the behaviors of the four types of neurons presented and demonstrate robustness of the approach under the presence of noise. These results validate the manipulation of the model equations into the linearly parameterized model (11) as well as the efficacy of the parameter estimation approach. The effect of parameters $a$, $b$, and $d$ on the firing rate of the neuron merits more discussion. High error in these estimated parameters associated with the adaptation state equation (2) have been consistent throughout each simulation, though the parameters still produce valid results. At first glance, this may suggest a lack of dependency on these parameters towards the output. However, with all other parameters held constant, slight changes to the values to $a$, $b$, or $d$ can drastically affect the firing rate of the output. One explanation that may attribute to the error is the possibility that there are multiple solutions that produce valid output. Since the goal of the approach is to minimize the error between the LP model and reference data, the simulated annealing algorithm may be selecting alternate values of $a$, $b$, and $d$ that result in the least error and thus a valid prediction. Another important observation to note is the significance of the state variable $u$ in the adaption equation. This variable is designed to offer adaptive characteristics of neurons for the model to replicate. While its presence in the model is essential for biological plausibility, the variable itself is immeasurable and therefore lacks biological meaning. With the goal of utilizing this approach for experimental data in mind, the importance of the exact values
of these parameters associated with the immeasurable $u$ variable can be understated, so long as they result in a suitable prediction.

Another point that merits more discussion is that the simulated annealing parameters selected can dramatically affect the convergence of the error to zero. The initial temperature chosen greatly contributes to this convergence. Higher temperatures allow for a more broad exploration of the parameter search space while lower temperatures attempt to explore a local minimum. Initially, a temperature of 100,000 was used on all parameters during simulation. Though at times this high temperature resulted in good predictions, the parameter search was much more stochastic and did not fully localize to a minimum. Often times good results were difficult to reproduce given the same set of data, suggesting that the approach depended more on luck to achieve such results. Instead, lower temperatures, based on a scale of 1000, were assigned to each individual parameter and were adjusted based on the importance of the parameter to vary during simulation (i.e. $v_r$ and $c$ will have lower temperature values since they are predetermined before the start of simulation). The product is a slightly slower yet more systematic approach to allow for the convergence of the error to zero. This temperature setting was utilized in the Results section for its consistency when reproducing good results. With that being said, the temperature values selected were based on trial and error. Choosing the appropriate temperatures for each parameter can considerably improve the efficacy of the simulated annealing method and thus requires more attention and research.
CONCLUSION

In this thesis, a simulated annealing approach has been proposed to estimate the parameters in the Izhikevich quadratic spiking neuron model. Results obtained from this approach conclude that the model can be represented by a manipulation of its equations into a linearly parameterized model, allowing for model parameter estimation based on injected current and membrane potential. Several simulations were run on four different types of neurons to validate the approach on its ability to replicate a wide range of neuronal characteristics as well as its robustness to noise. Future work will focus on employing this parameter estimation approach on membrane voltage data from detailed ion-channel based Hodgkin-Huxley type model as well as in vitro data collected from embryonic rat motoneurons to validate the approach with biological systems.
APPENDIX
APPENDIX

MATLAB Implemented Code

Main.m

% Main file to generate quadratic model data and run parameter estimation

%% Generate Model Data

clearwrk = 0;
% Clear screen and workspace data
if (clearwrk)
    clc; clear;
end

noise = 0;
neuron = 0;
savewrk = 0;

% Set step size, amplitude, and frequency for data_i
step_size = 100;
I = [30, 100, 70, 80]
omega = [.05 .3 .2 .25];

% Assign parameter values to theta
theta = [100, .7, -60, -40, .03, -2, -50, 100]; %Regular Spiking
% theta = [150, 1.2, -75, -45, .01, 5, -56, 130]; %Intrinsically Bursting
% theta = [50, 1.5, -60, -40, .03, 1, -40, 150]; %Chattering
% theta = [.04, 5, 140, 1, .02, .2, -65, -0.5]; %Rapid Adapting
% theta = convert_theta(theta,-82.6556)
% Set the values for V0(starting point), Vp(voltage peak), and run time
global Vp
V0=-70; Vp = 35;

%% Run simulink model to get Vr
 t_start = 0; t_max = 0.02; t_run = 1000;
C = theta(1); k = theta(2); vr = theta(3); vt = theta(4);
a = theta(5); b = theta(6); c = theta(7); d = theta(8);

k1 = k/C; k2 = -k1*(vr+vt); k3 = k1*vr*vt; k4 = 1/C; k5 = k4; k6 = vr;

cur = i_sin(I*0, omega, t_run, t_max, 0);
% Run Izhikevich model implemented in Simulink
sim('s_izhikevich');

vr_est = vv.signals.values(end)
clearvars -except V0 Vp noise neuron savewrk step_size I omega theta vr_est

%% Run simulink model again to get data_v

t_start = 0; t_max = 0.02; t_run = 2000;
cur = i_sin(I, omega, t_run, t_max, step_size);

C = theta(1); k = theta(2); vr = theta(3); vt = theta(4);
a = theta(5); b = theta(6); c = theta(7); d = theta(8);

k1 = k/C; k2 = -k1*(vr+vt); k3 = k1*vr*vt; k4 = 1/C; k5 = k4; k6 = vr;
% Run Izhikevich model implemented in Simulink
sim('s_izhikevich');

clc

% Assign data for parameter estimation
data_v = [vv.time,vv.signals.values]; %Membrane potential (will be modified)
if(exist('noise','var')&&noise) %Adds 40dB noise
    data_v(:,2) = awgn(data_v(:,2),40);
end
model_v = data_v; %Membrane potential (second copy)
data_i = [current.time,current.signals.values]; %Current
data_im = spiketime(data_v); %Spike times of membrane potential
data_u = [uuu.time,uuu.signals.values];

% Saves current to file for NEURON use
if(exist('neuron', 'var')&&neuron)
    data_i_nano = [data_i(:,1) data_i(:,2).*0.01];
    save('C:\cortex\data_i.txt', 'data_i_nano', '-ascii', '-tabs')
end

% Plot all data
p(data_v);%, data_i, data_im);

clear k1 k2 k3 k4 k5 k6 k c vr vt a b c d nn current tout neuron clearwrk cur

%% Obtain W Realizable Regression Vector
% Define filter components
global alpha_0 alpha_1
alpha_0 = 1 ; alpha_1 = 2 ;
% Run simulink implementation of Linearly Parameterized (LP) model
sim('s_lpmodel')
clc

%% Clip data_v and V to remove initial transient
global W v time
x = [1/theta(1) theta(2:8)];
theta_0 = [x(1)*x(2), x(1)*x(2)*x(5), -x(1)*x(2)*(x(3)+x(4)) + alpha_1 - x(5), -x(1)*x(2)*x(5)*(x(3) + x(4)) + alpha_0 - x(1)*x(5)*x(6), x(1)*x(2)*x(3)*x(4)*x(5) + x(1)*x(3)*x(5)*x(6), x(1), x(1)*x(5), x(7) - Vp, x(5)*(x(7) - Vp) - x(1)*x(8)];
W = [WW.signals.values(:,2), WW.signals.values(:,3), WW.signals.values(:,4), WW.signals.values(:,5), WW.signals.values(:,6), WW.signals.values(:,7), WW.signals.values(:,8) , WW.signals.values(:,9), WW.signals.values(:,10)];
v = [WW.time w*theta_0];
data_v = [WW.time WW.signals.values(:,1)];
p(V, data_v);

t_new = 20; %in milliseconds
data_v_new = select_data(data_v, t_new, t_run, t_max);
w = select_data(w, t_new, t_run, t_max);
v = data_v_new(:,2);
time = data_v_new(:,1);
wtheta = w*theta_0;

plot(data_v_new(:,1), v, data_v_new(:,1), w*theta_0)

clear t_new

%% Adjust w*theta to match v and reduce error
e = wtheta - v;
plot(time, e)
mean(e);
global avg_error
avg_error = e;
avg_error = zeros(size(w(:,1)));
e_new = wtheta - v - avg_error;
e'*e
e_new'*e_new

%clear e_new e

%% Define Lower and Upper bounds and Starting Point
% Sets the starting point to zero
x0 = [0 0 0 0 0 0 0 0];
% Defines ±1% boundary for vr and ±100% boundary for all other parameters
p = 1.0;
p_vr = .01;
neg = 1-p;
pos = 1+p;
low = zeros(1,8);
upp = zeros(1,8);
for k = 1:8
  if(x(k)<0)
    if(k==3)
      low(k) = vr_est*(1+p_vr);
      upp(k) = vr_est*(1-p_vr);
    elseif(k==3)
      low(k) = vr_est*(1-p_vr);
      upp(k) = vr_est*(1+p_vr);
    else
      low(k) = x(k)*pos;
      upp(k) = x(k)*neg;
    end
  end
end
if(k==3)
    low(k) = vr_est*(1-p_vr);
    upp(k) = vr_est*(1+p_vr);
else
    low(k) = x(k)*neg;
    upp(k) = x(k)*pos;
end
end
clear neg pos k p_vr

%% Run Optimization Toolbox

optimtool

% Here the toolbox will open and simulated annealing parameters must be
% chosen for optimization. The following values used for all simulations:
% Objective function: @myfun
% Start point: x0
% Bounds:    Lower: low         Upper: upp
% Max iterations: Default (400)
% Max function eval: 30000
% Time limit: Default (Inf)
% Function tolerance: Default (1e-6)
% Unboundedness threshold: Default (1e-20)
% Stall iterations: inf (or high value, 100000000)
% Annealing function: Fast Annealing
% Reannealing interval: Default (100)
% Temperature update function: Exponential
% Initial temperature: 200000
%% Compare Model Vs Estimated
% Once simulated annealing is complete, results must be exported to the
% workspace as optimresults.
if(exist('optimresults', 'var'))

% Generates target data using step current
step_size = 100;
I = zeros(1,4);
t_run = 1000;

C = theta(1); k = theta(2); vr = theta(3); vt = theta(4);
a = theta(5); b = theta(6); c = theta(7); d = theta(8);

k1 = k/C; k2 = -k1*(vr+vt); k3 = k1*vr*vt; k4 = 1/C; k5 = k4; k6 = vr;
cur = i_sin(I, omega, t_run, t_max, step_size);
sim('s_izhikevich');
model_v = [vv.time,vv.signals.values];  %Membrane potential (second copy)
data_im = spiketime(model_v);    %Spike times of membrane potential
data_u = [uuu.time,uuu.signals.values];
% Defines estimated parameters
theta_est = optimresults.x;
theta_est = [1/theta_est(1) theta_est(2:end)];

% Obtains percentage error between parameters
percent_error = zeros(1,8);
for k = 1:8
    percent_error(k) = abs((theta(k) - theta_est(k))/theta(k))*100;
end
clear k
\begin{verbatim}
C = theta_est(1); k = theta_est(2); vr = theta_est(3); vt = theta_est(4);
a = theta_est(5); b = theta_est(6); c = theta_est(7); d = theta_est(8);

k1 = k/C; k2 = -k1*(vr+vt); k3 = k1*vr*vt; k4 = 1/C; k5 = k4; k6 = vr;

% Run simulink model
sim('s_izhikevich');

% Assign data for comparison
v_est = [vv.time,vv.signals.values];
im_est = spiketime(v_est);
u_est = [uuu.time,uuu.signals.values];
clc
% Prints out results
theta'
theta_est'
percent_error'
im_est(end,2)
data_im(end,2)
spike_error = abs(im_est(end,2)-data_im(end,2))/data_im(end,2)*100;
fprintf('Step Size = %d pA \n', step_size);
fprintf('Spiking Error = %.2f', spike_error);
disp('%')

%Plot data to compare
subplot(2,1,1); plot(model_v(:,1), model_v(:,2));
if(I(1)==0)
    title(sprintf('Step Input = %d pA', step_size));
end
\end{verbatim}
legend('Target');
ylabel('[mV]');
subplot(2,1,2); plot(v_est(:,1), v_est(:,2));
legend('Prediction');
ylabel('[mV]');
xlabel('[ms]');

% plot(model_v(:,1), model_v(:,2), v_est(:,1), v_est(:,2))
% ylabel('Voltage (mV)');
% xlabel('Time (ms)');
% legend(sprintf('Target: %d sps', data_im(end,2)), sprintf('Prediction: %d sps', im_est(end,2)));
% title('Neuron');
% plot(data_u(:,1), data_u(:,2), u_est(:,1), u_est(:,2))
clear k1 k2 k3 k4 k5 k6 k c vr vt a b c d nn current tout
end

%% Save Result
if (savewrk)
    clearvars -except vr_est step_size I omega theta vp V0 t_start t_run
t_max p x0 theta_est optimresults options low upp percent_error spike_error
    noisepower
    run save_result
end

objfun.m

function F = myfun(x)
global alpha_0 alpha_1 w v vp avg_error;
theta_e = [x(1)*x(2), x(1)*x(2)*x(5), -x(1)*x(2)*(x(3)+x(4)) + alpha_1 - x(5),...
-x(1)*x(2)*x(5)*(x(3) + x(4)) + alpha_0 - x(1)*x(5)*x(6),...
- x(1)*x(2)*x(3)*x(4)*x(5) + x(1)*x(3)*x(5)*x(6), x(1), x(1)*x(5),...
- x(7) - Vp, x(5)*(x(7) - Vp) - x(1)*x(8)]';

e = W*theta_e - v;
F = e'*e;

spiketime.m

% Returns a running total number of spikes that occurred at each instant
% given a set of membrane potential data
function [data_im] = spiketime(data_v)

%%
n=0; temp1=0; temp2=0;
data_im = data_v(:,1);
data_im(1,2)=0;

for i=2:length(data_v(:,1));
    if data_v(i-1,2) - data_v(i,2) >= 30
        n = n+1;
        temp2=data_v(i,1);
        if (temp2-temp1) < 20
            data_v(i,1)
        end
        temp1=temp2;
    end
    data_im(i,2) = n;
end

clear n temp1 temp2 i
**select_data.m**

% Clips a set of data by selecting new start and end points
function [data_clipped]=select_data(data, t_beg, t_end, t_max)
    interval = (t_beg/t_max:t_end/t_max);
    data_clipped = data(interval,:);
    clear t_end t_beg data_i_s data_v_s

**p.m**

function [] = p(data1, data2, data3)
    %p Easy plot
    % plots 2D data with an easier function call
    if nargin == 1
        plot(data1(:,1), data1(:,2));
    elseif nargin == 2
        plot(data1(:,1), data1(:,2), data2(:,1), data2(:,2));
    elseif nargin == 3
        plot(data1(:,1), data1(:,2), data2(:,1), data2(:,2), data3(:,1), data3(:,2));
        %axis([0 x_max -80 60]);
    elseif nargin ==
        plot(data1(:,1), data1(:,2), data2(:,1), data2(:,2) + offset);
    end
end
REFERENCES


