Estimating Parameters of a Multi-Class Izhikevich Neuron Model to Investigate the Mechanisms of Deep Brain Stimulation

A Thesis

Submitted to

Temple University Graduate Board

In Partial Fulfillment

Of the Requirements for the Degree

MASTER OF SCIENCE

In ELECTRICAL ENGINEERING

Christopher Tufts

April 30 2013

Thesis Approval(s):

Iyad Obeid, PhD., Thesis Adviser, Electrical and Computer Engineering

Joseph Picone, PhD., Electrical and Computer Engineering

Li Bai, PhD., Electrical and Computer Engineering

Abstract

The aim of the research is to provide a computationally efficient neural network model for the study of deep brain stimulation efficacy in the treatment of Parkinson’s disease. An Izhikevich neuron model was used to accomplish this task and four classes of neurons were modeled. The parameters of each class were estimated using a genetic algorithm with a fitness function based on spike frequency as a function of input current. After computing the optimal parameters the neurons were interconnected to form the network model. The Izhikevich network model was simulated, but was unable to reproduce the individual neuron characteristics seen in a biological network.

Table of Contents

2. Introduction 4

2.1. Motivation 4

2.2. Research Objectives 5

2.3. Organization of Thesis 5

3. Background 6

3.1. Deep Brain Stimulation 6

3.2. Neural Models 8

3.2.1. Hodgkin Huxley Model 8

3.2.2. Izhikevich Model 11

4. Methods 17

4.1. Phase Plane Analysis 17

4.2. Parameter Estimation 19

4.2.1. Fitness Functions 20

4.2.2. Parameter Search Algorithm 24

5. Simulation 26

5.1. Introduction 26

5.2. Multi-Class Neuron Model 27

5.2.1. Hodgkin Huxley Network Model 27

5.2.2. Izhikevich Network Model 28

5.3. Parameter Estimation 29

5.3.1. Testing and Selection of Fitness Function 29

5.3.2. Genetic Algorithm and Estimation of Izhikevich Parameters 33

5.3.3. TC Parameter Estimation 33

5.3.4. STN Parameter Estimation 37

5.3.5. GPe Parameter Estimation 40

5.4. Multi-Class Model Testing and Results 44

5.5. Discussion of Network Model Results 46

5.6. Computational speed/resource comparison 48

6. Conclusion 50

7. References 52

8. APPENDIX 57

8.1. GPi (GPe) Neuron 57

8.2. STN 59

8.3. TC 59

# Introduction

## Motivation

The goal of the proposed research is to provide a computationally efficient model to allow for the study of deep brain stimulation (DBS) for the treatment of Parkinson’s disease (PD). PD affects tens of millions of people worldwide and the frequency and economic burden of the condition are set to increase as the elderly population grows. PD is a neurodegenerative disorder characterized by tremors of the limbs and impaired muscular movements. [1]

DBS is a therapeutic strategy used to reduce symptoms associated with PD such as tremors and decreased locomotion. [2] However, the mechanisms behind DBS remain unclear and a topic of debate. [3] One method of investigation is to use computational models comprising several areas of the brain close to the stimulation site.

Modeling the stimulation site requires simulating multiple classes of neurons. DBS not only affects the neurons at the stimulation site, but also all the neurons in the surrounding network connected to the stimulation site. To study the dynamics of the network, a biologically plausible model must be used to simulate each neuron. However one tradeoff of in implementing biologically plausible neural models, such as the Hodgkin Huxley model, is high computational complexity for high accuracy. The complexity severely limits the size of the networks and the length of the simulations, which in turn limits their predictive or explanatory power. There are other models, such as the *Integrate and Fire* model, which offer a low level of computational complexity, but lack biophysically realistic results.

The Izhikevich neuronal model is capable of offering biologically plausible results with low computational complexity. Creating a large-scale multi-class network using Izhikevich neurons would allow the long and short-term effects of DBS to be studied. This could provide great insight to some of the underlying mechanisms of DBS.

## Research Objectives

The primary objective of the proposed research is to develop a computationally efficient and scalable neural network to study the effects of DBS using the Izhikevich modeled neurons. The neural network will consist of four different types of neurons: sub-thalamic nucleus (STN) neurons, thalamic cortical relay (TC) neurons, globus pallidus interna (GPi) neurons, and globus pallidus externa (GPe) neurons. Each neuron will be modeled using the Izhikevich neuron model. A principal objective of this work will be to find the parameters of each neuron using an error function based on the frequency-current (f-I) curve and a genetic algorithm to search the solution space. The test data for the parameter estimation is derived from biologically plausible Hodgkin Huxley (HH) neuron models of each individual class of neuron. After the parameters have been estimated for each type of neuron, the Izhikevich neurons will be connected in a network. The final outcome of the project will be a network of Izhikevich neurons capable of mimicking a network of HH neurons. The model will be designed to replicate the HH neural network published by Rubin and Terman in 2004. [4] Finally, the computational savings of the Izhikevich model over the HH model will be calculated for comparison.

## Organization of Thesis

The thesis is organized in the following manner. A comprehensive background is provided covering all aspects of the research. The background begins with an overview of the motivation for the research: DBS. The next background component will cover two types of neuron models: the HH model and the Izhikevich model. The next component is the methods section that will provide information on fitness functions and the genetic algorithm used for parameter estimations. Following the methods section, the results of the research will be discussed. The final section will consist of conclusions based on the research.

# Background

The overall goal of this research is to provide a simulation tool to study the mechanisms behind DBS efficacy. The mechanisms behind DBS remain unclear and the benefits of the treatment have primarily been established more or less empirically. [5] In previous studies, local field potentials (LFP) have been recorded during deep brain stimulation as a measure of synchronized neuronal activity. However no direct causal link between LFPs recorded during DBS and corresponding motor symptoms has been demonstrated. [6][7]

Ideally the optimal method for studying DBS would be to make electrical recordings from neurons throughout the brain in response to DBS stimuli. Since this is practically impossible, simulation is the next best tool to study DBS. [8] Numerical simulation is currently the only valid option to offer insight into the activity in the neuronal areas affected during treatment. The use of simulation may help uncover the network dynamics occurring as a result of DBS. If the network interactions during treatment are known, optimal DBS parameters could be calculated empirically instead of being determined ad hoc. A valid measure of DBS efficacy could allow for closed loop tuning of DBS allowing patients with PD a higher level of care.

## Deep Brain Stimulation

DBS is therapeutic treatment for a variety of neurological issues including Parkinson’s disease and Dystonia. PD and Dystonia are motor control disorders characterized by uncontrolled muscle tremors or spasms. They are known to be caused by disease in one of the several structures of the deep brain responsible for feed-forward motor control. For example, PD is caused by death of dopaminergic neurons in the substantia nigra, which is part of the deep brain feedback loop that modulates motor control. [1][9] DBS electrodes may be placed bilaterally or a single electrode may be implanted unilaterally depending on the treatment selected. [10] Each implanted stimulation electrode has a single stimulation site. A pulse generator implanted subcutaneously in the subclavicular area drives the electrode. [11] Figure 1 shows a typical DBS implantation. [12] The stimulating electrode can be implanted in the STN, the GPi, or the ventral intermediate (VIM) nucleus. The frequency of stimulation ranges from 130 Hz to 185 Hz and is manually adjusted during the implantation procedure. [13]

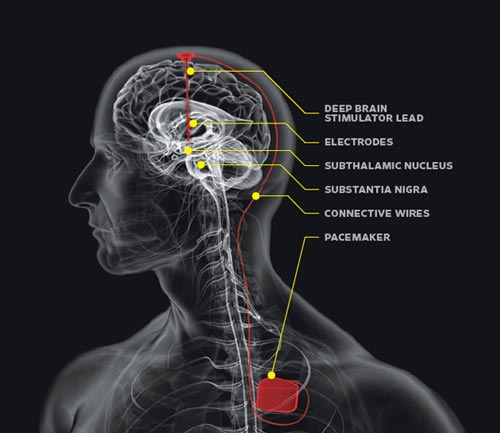


Figure - DBS Implant Hardware

The mechanisms behind DBS’s efficacy are largely unknown. For example, it is not known whether DBS acts to enhance or suppress neuronal activity within a given brain structure, which areas and which neurons within these areas are acted upon by DBS, or how the geometry and orientation of the neurons modulate the effect of the electric field generated by DBS. [4] Increased neuronal activity in the GPi and STN is thought to account for the motor dysfunction in PD.[11] One possible mechanism underlying DBS is that it suppresses neuronal activity. This belief is held because DBS has a similar outcome to ablative surgeries. [14] Another possible mechanism for the efficacy of DBS is the increase in activity in the GPi causing downstream effects in other neurons. [4]

## Neural Models

### Hodgkin Huxley Model

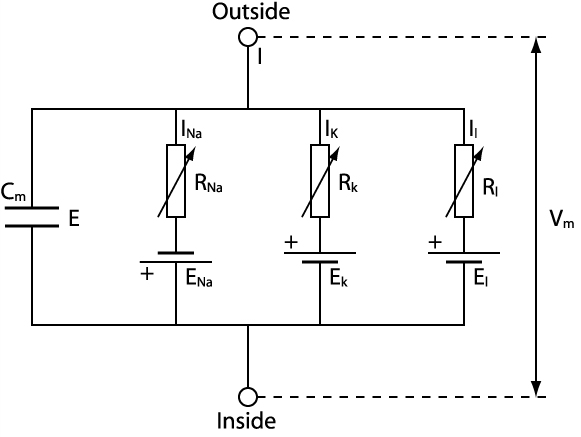


Figure - Hodgkin-Huxley Circuit Representation

The Hodgkin Huxley model simulates the membrane voltage of an electrically active neuron using a network of capacitors and resistors. Current can move across the membrane by charging the membrane (Cm) or by movement of ions through resistances in parallel with the capacitance. The ionic channels present in this model are the Potassium (Ik) channel, the Sodium (I­Na) channel, and the leakage channel (IL). The leakage channel represents the channel that chloride and other ions may travel across. [15]. The membrane channel impedances are voltage dependent and are modeled using first-order dynamics. Each Hodgkin-Huxley neuron requires four simultaneous differential equations to solve. Although accurate, HH models are not computationally efficient, especially when models scale to hundreds or thousands of interconnected neurons. Although HH was originally created to model membrane dynamics of squid axons, the general model style has since been used to model dozens of cell types. The term “HH model” refers to a membrane model with parallel capacitive and resistive elements. Each resistive element models one type of voltage gated ion channel, and includes a series constant “activation” voltage that models the Nernst potential. Each ion channel is controlled by voltage-dependent “gating variables” which are modeled with first-order dynamics. Gating variable parameters must be carefully matched to biological measurements in order for the model to produce biophysically relevant simulations. Depending on the sophistication of the cell type being modeled, the number of parallel differential equations per neuron is a minimum of five.

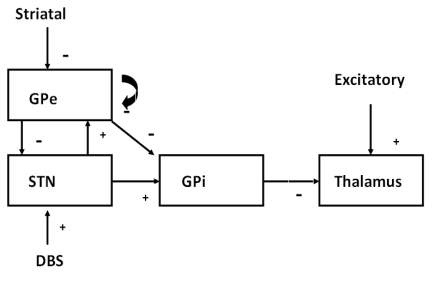


Figure - Network Architecture

The goal of the research is to build Izhikevich neuron models that accurately mimic the corresponding HH models. The Izhikevich model has eight parameters, which must be optimized to match the HH models. A genetic algorithm with an f-I based fitness function will be used to determine the optimum parameters. The neural network architecture for the proposed work can be seen in Figure 3. The HH mathematical descriptions of each of the four classes of neurons are outlined in the Appendix. The network has been modeled previously using HH neurons in research by Rubin and Terman. [4], [16] The Izhikevich network model will allow future work in which network models can be simulated that are an order of magnitude larger.

The network architecture outlined in Figure 3 shows the basic structures affected by DBS when applied at the STN. The “+” signs represent excitatory input currents and the “-“ signs represent inhibitory input currents. Each TC neuron receives excitatory input from the sensory motor cortex and inhibitory input from eight GPi neurons. Each GPi neuron receives inhibitory input from two GPe neurons and excitatory input from one STN neuron. Each STN neuron receives inhibitory input from two GPe neurons. The STN neurons also receive a DBS input current. The GPe neurons receive excitatory input from three STN neurons and inhibitory input from two other GPe neurons.

Although increased activity in the GPi is associated with PD, the model sets out to show that high frequency DBS will further increase GPi activity resulting in restoration of thalamic relay capabilities. The signals from the STN and GPe interact with the GPi currents to generate patterns of GPi activity consistent with experimental data. The TC cell acts a relay for signals from the sensory motor cortex. In the case of Parkinson’s disease, these signals are not relayed correctly or at all. However if DBS is applied at the STN, the STN becomes increasingly active. The increased activity in the STN results in increased activity in the GPi, inducing the neurons in the GPi to fire tonically at high frequency. This tonic high frequency firing results in restoration of the relay capability of the TC neurons. [4]

### Izhikevich Model

#### Model Description

The HH model is capable of producing all the firing patterns exhibited by real biological neurons. The primary issue with the HH model is the model’s computational complexity and extreme sensitivity to time step size during simulation. The HH sensitivity to time step was uncovered during preliminary work while replicating Rubin and Terman’s network model. Time steps as small as 1e-6 milliseconds with a fourth order Runge Kutta method were needed to ensure the model’s operation. Therefore the HH model is not well suited to large-scale neural model simulations. In Figure 4 a comparison of run-time is shown for the network model using three different algorithms: 1) HH with adaptive Runge Kutta, 2) HH using 4th order Runge Kutta and a time step of 0.01mS, and 3) Izhikevich using a forward Euler algorithm with a fixed time step of 0.01mS. The x-axis displays the scale of the model. This effectively means the original population of the Rubin and Terman model was scaled by this factor. For instance at a scale of one, the original model consisting of two TC, 16 GPe, 16 GPi, and 16 STN neurons is created and run. At a scale of two, the model will run 4 TC, 32 GPe, 32 GPi, and 32 STN neurons, and so on. The network was simulated to 100mS.

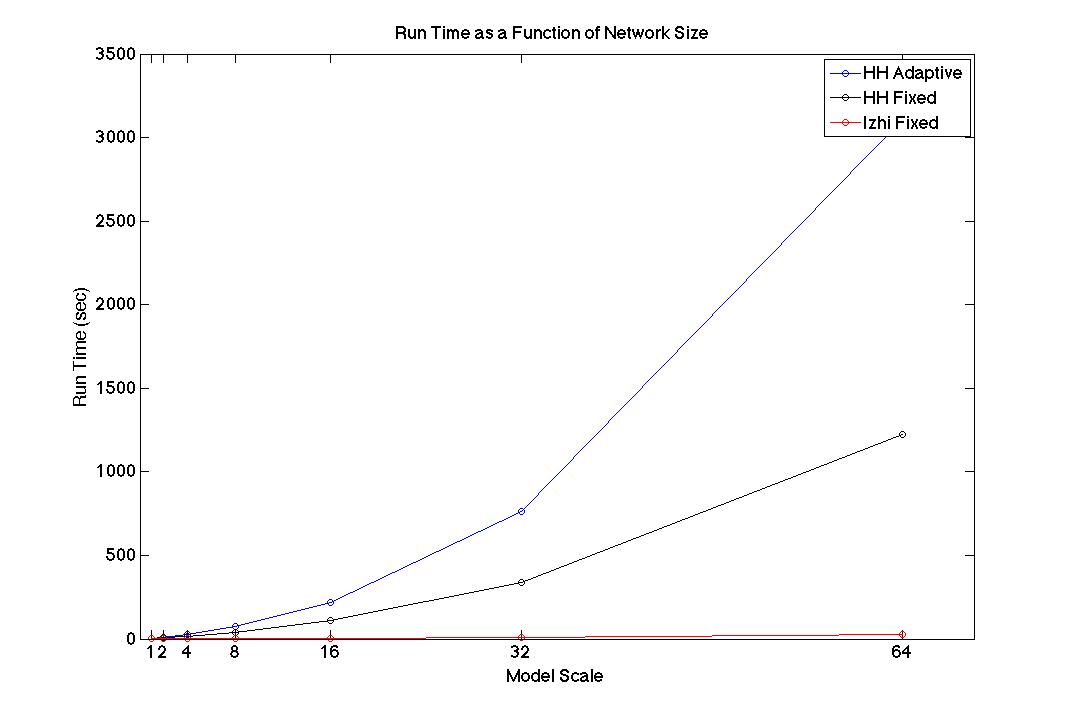


Figure 4 - Run time as a function of network size

A computationally simple model, known as the Izhikevich model, is capable of producing rich firing patterns exhibited in biological neurons. The Izhikevich model offers an alternative to the HH model and allows large-scale models to be simulated. [17] Whereas HH strives to create a biologically plausible model of each substructure in the neuron membrane, the Izhikevich model strives to create a simplified model in which many details are compressed into just two equations; with proper parameterization, the end result can be a biologically plausible cell model. In addition to the lower computational complexity, a lower order solution method can be used for simulations with an Izhikevich model. For example, a simple first order forward Euler method with a time step of 0.01ms can be used to solve the Izhikevich equations.

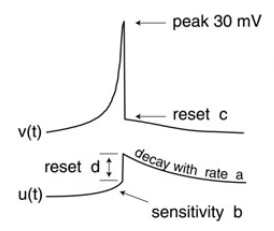


Figure 5 - Izhikevich spike with primary parameters

The Izhikevich model has just two differential equations governing the membrane potential. These are membrane voltage *v*, and the recovery current, *u* as seen in Equation 1, Equation 2, and Equation 3. The variable *C* represents the membrane capacitance, *vr* is the resting potential, *k* describes the upstroke shape of an action potential, and *vt* is the instantaneous threshold potential. The recovery time constant is *a*.

Equation 4

Equation 5

Equation 6

The variable *b* determines whether the recovery variable, *u*, is a resonant or an amplifying current. If *b* is greater than zero the recovery current will be a resonant current, if *b* is greater than zero the recovery current will be an amplifying current. Resonant currents cause the neuron membrane voltage to sag in response to hyperpolarized pulses of current, peak in response to depolarized subthreshold pulses, and produce rebound, or post-inhibitory, spikes. Amplifying currents allow the neuron to act as a quadratic integrate and fire neuron. The quadratic integrate and fire neuron is not capable of bursting or postinhibitory spikes. However the model is capable of exhibiting class one excitability, or the ability to encode the input current to the neuron.

If a spike is detected the membrane potential is reset to *c* and the recovery current is altered. The variable *d* represents the total amount of outward minus inward currents activated during the spike and affecting the after-spike behavior. [18]

The Izhikevich model is capable of a variety of biologically realistic neuron spiking patterns. A summary of the patterns can be seen in Figure 5. Class two excitability differs from class one excitability in that it does not encode the input to the neuron. Regardless of excitatory input amplitude a class two neuron will maintain a constant spiking frequency. We can also see a variety of responses to depolarization and hyperpolarization. Many neurons exhibit a combination of the behaviors shown in Figure 5. Different firing patterns may require alterations to parameters in the model. It is not unusual to have several vectors of parameters for different conditions based on membrane potential and input current for a single neuron model.

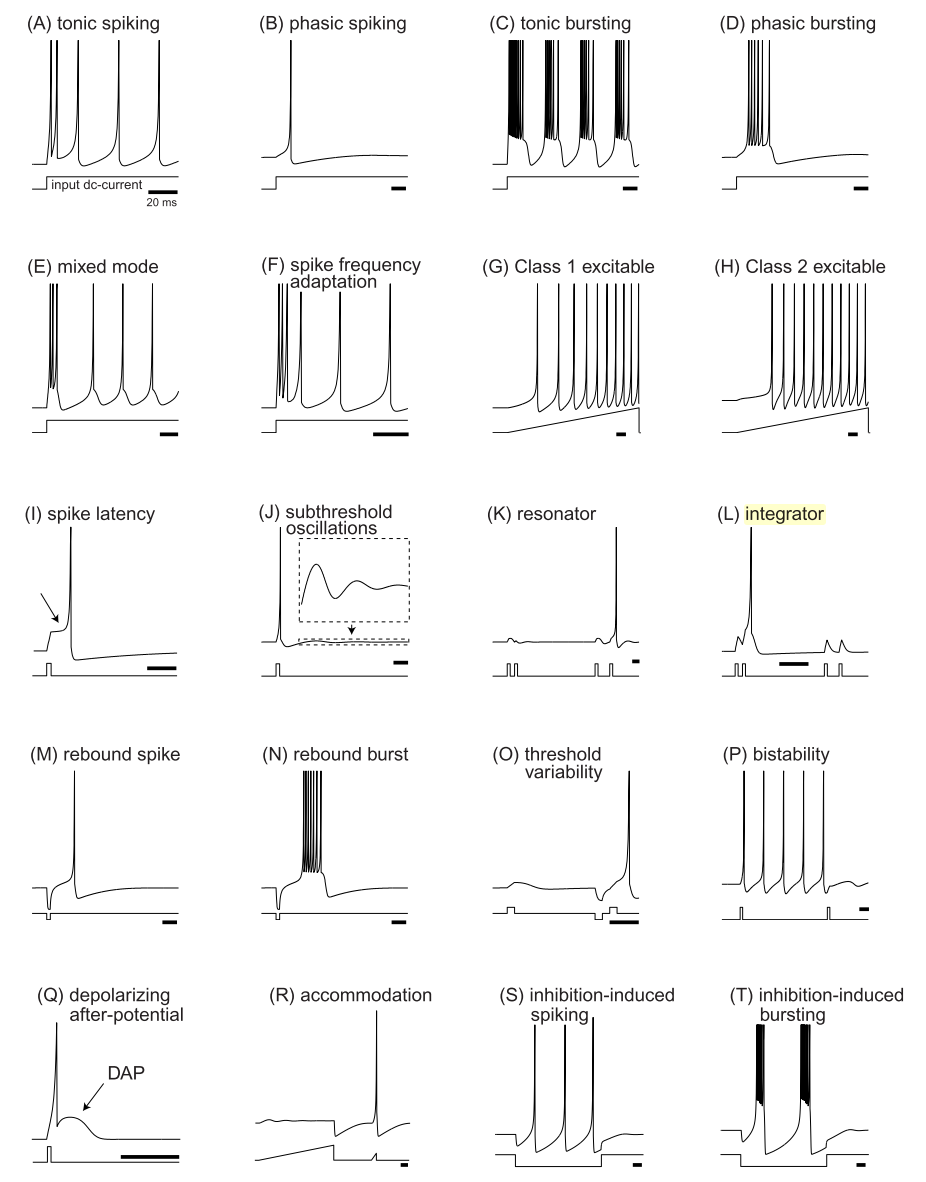
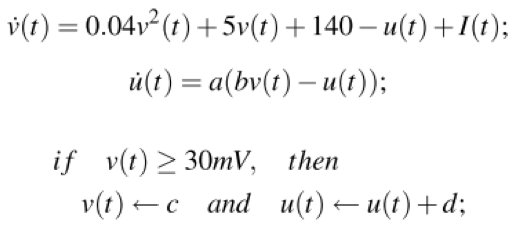


Figure 5 - Summary of neurocomputational properties for the Izhikevich neuron [18]

#### Determination of Parameters for Izhikevich

In previous studies estimation of the parameters for the Izhikevich model has been carried out using several different methods. Izhikevich states, “To fine-tune the model, we use recordings of real neurons. We consider the tuning successful when the quantitative difference between simulated and recorded responses is smaller than the difference between the responses of two “sister” neurons recorded in the same slice.” [18] However throughout his text he offers no estimation procedure, which implies manual tuning of the neuron’s parameters dependent on desired neuron behavior.

In previous work by Dur-e-Ahmad, several parameters were determined via biological data and the rest of the parameters were manually tuned based on spike frequency adaptation characteristics. [19] The parameters *vt, k, C, c, and vr* were estimated based on biological recordings. However the parameters *a, b,* and *d* were manually tuned based on spike frequency adaptation characteristics.



Equation 7 - Simplified Izhikevich model

Inter-spike interval along with firing frequency of a real neuron have been used to train parameters for the Izhikevich neuron in a previous studies by Kumar et al and Michmizos. [20] [21] In the previous works the simplified version of the Izhikevich model was used. This model is seen in Equation 7. Therefore only 4 different parameters were trained: a, b, d, and I. Parameter “c” was set to the resting potential of the neuron.

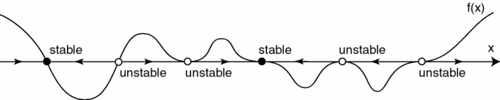
# Methods

## Phase Plane Analysis

Phase plane analysis of a system is the geometrical representation of system dynamics. The analysis is carried out using phase portraits, which depict the equilibria and trajectories of the system’s state/phase space. In the case of a one-dimensional system, the phase portrait consists of a phase line, trajectories, and equilibria as seen in Figure 6. [22] The phase line is based on a state variable, *x*, and its derivative represented by *f(x)*. The trajectories are represented by the arrows on the x-axis and the equilibria are represented by the circles.

The black circles represent stable equilibria and the white circles are unstable equilibria. Equilibria occur where *f(x)* is equal to zero. In Figure 6 we can see that unstable equilibria have trajectories which point away from the equilibrium at all times. The stable equilibrium act as attractors and have trajectories pointing into the equilibrium. Using the trajectories and equilibria we can easily determine the dynamics of a system with the initial value of the state variable *x*.

Figure 6 - Phase Line



The Izhikevich neuron is often analyzed using a two-dimensional phase portrait. The dynamics of the Izhikevich neuron can be described using the state variables *v* and *u*. In Figure 7 below we can see an example of a phase portrait for an Izhikevich neuron. [18]

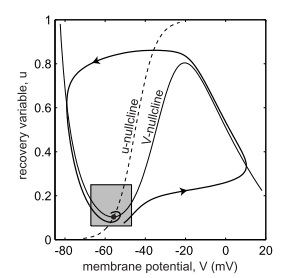


Figure 7 - Izhikevich Neuron Phase Portrait

In Figure 7, the trajectory of the system is shown by the directional line. [18] The *u* nullcline is represented by the dashed line and is defined as all points where the derivative of *u* with respect to time is equal to zero. Similarly, the *v* nullcline is defined by all points in the system where the derivative of *v* with respect to time is equal to zero. The nullclines of each state variable are important because each intersection of the nullclines represents an equilibrium point.

The trajectory shown in Figure 7 is calculated based on a set of initial conditions (*v(0),u(0)*). The direction of the trajectory line is indicated by the value of (*dv(t)/dt, du(t)/dt*) for *t=0* to a preset stopping time. The trajectory line is the value of *v(t), u(t)* at each point of the time range being calculated. Therefore we can tell the trajectory of the membrane potential and recovery variable if we have the initial conditions.

## Parameter Estimation

The following section outlines several error functions, also know as fitness functions, used along side the genetic algorithm to determine parameters for the Izhikevich neurons. Phase plane trajectory density (PPTD), root mean square (RMS) error based on membrane potential, and the f-I curve based error function were all examined, implemented and tested. The f-I curve based error function was found to be the only method capable of training an Izhikevich neuron.

The parameter estimation method for the research was a genetic algorithm (see 4.2.2) using an f-I curve sum of squares error algorithm as a fitness function. The parameter estimation began with an initial guess at the parameters to be used in the Izhikevich model. The spiking frequency of the Izhikevich model at each input current setting was compared to the training data provided by an HH model used in the previous works of Rubin and Terman. [4] The difference between the two data sets will be found using a sum of squares error function based on spike frequency. The set of parameters that result in the lowest error value will be passed onto the next generation of the genetic algorithm. All other members of the population will consist of new parameter “guesses”.



Figure 8 - Flowchart outlining the basic operation of the genetic algorithm

### Fitness Functions

#### Point-To-Point Membrane Potential RMS

Equation 8 - RMS

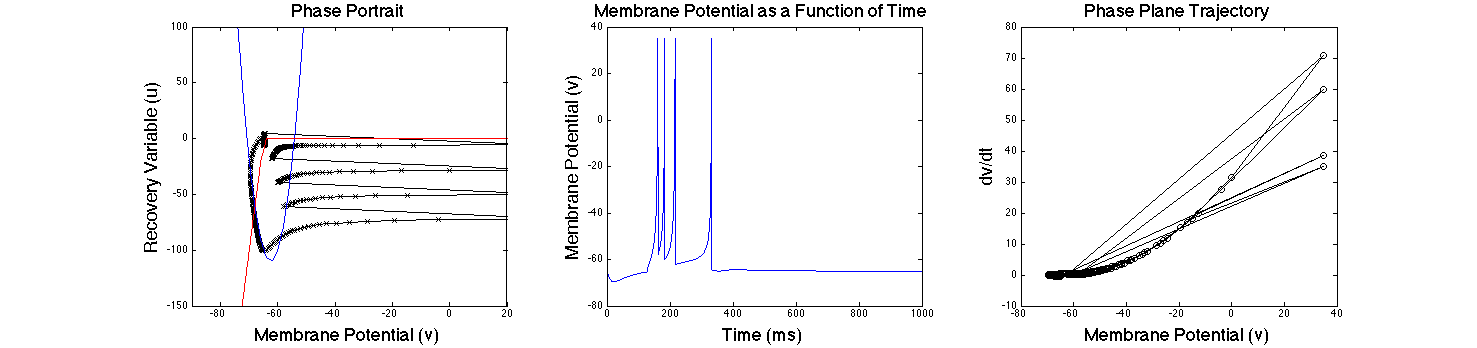
One potential error function to be used in conjunction with the genetic algorithm is the point-to-point RMS error function using the membrane potential of a neuron. This algorithm can be seen in Equation 8. The membrane potential of the training data is compared with the membrane potential generated using the Izhikevich model and returns a measure of error.

#### Phase-Plane Trajectory Density

Equation 9 - PPTD

The PPTD error function is displayed in Equation 9. The basic premise behind the PPTD method is to plot the phase plane of the model and to compare it to the phase plane of the training data. To obtain the phase plane, the voltage is plotted against the change in voltage over time. Next the phase plane is divided into a grid with each square of the grid representing a value of *i* and a value of *j* from Equation 9. The number of data points in each square of the phase plane plot are summed. Essentially a 2-D histogram of the phase plane is calculated. The values of the histograms make up the *dataij* and the *modelij* terms. *N* represents the total number of points and is used to normalize the values.

Figure 10 - From left to right: a) Phase portrait, b) membrane potential as a function of time, c) phase plane trajectory



As previously mentioned, the phase portrait for the Izhikevich neuron is based on two state variables: the recovery variable, *u* and the membrane potential, *v*. Figure 10 shows the two-dimensional phase portrait, the membrane potential as a function of time, and the one-dimensional phase portrait; all three plots depict the same event. In Figure 10a a phase portrait is displayed with the *v* nullcline represented by the blue line, the *u*-nullcline represented by the red line, and the trajectory of the neuron shown in black.

In the middle graph we see the neuron is firing a burst of rebound spikes. The neuron used in this simulation was given an inhibitory input for the first 125ms of the simulation, and then the input current was removed. The neuron fires multiple post-inhibitory spikes. The one-dimensional phase portrait, to the right, shows the derivative of the membrane potential plotted against the membrane potential at each point of the simulation.

In both the left and right graphs we can see the trajectory is repetitive. The graph in the middle confirms the repetitive trajectory as can be seen by the multiple spikes. Each time a spike occurs the neuron is reset to its resting potential. We can see on the far left the value of *u* is driven to a large negative value due to the inhibitory current and a positive *b* value. Once the inhibition is removed the neuron fires spikes at a decreasing frequency until the recovery variable becomes zero and the neuron returns to an equilibrium state. On the far left we see the derivative of the voltage is very large for the first spike, then decreases along the trajectory of each spike that follows.

PPTD utilizes the one-dimensional phase portrait for comparison between training data and model data. Although the two-dimensional phase portrait provides more information, it is not applicable in this case because our training data (from the Hodgkin-Huxley model) only contains membrane potential. In contrast, the two-dimensional phase portrait can be utilized when validating the parameters in the model to ensure the model displays the same dynamics as the original Rubin and Terman model.

The PPTD method is less sensitive to phase shifts than other point-to-point methods while still allowing for the same degree of accuracy. [23], [24] Error due to phase shift is demonstrated in Figure 11. PPTD avoids this error by discretizing the error function removing the need for matching spike times.

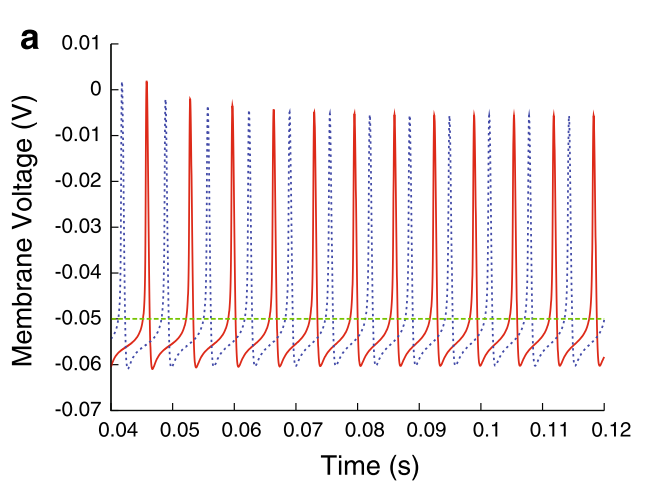


Figure – Phase shift example

#### Time Based Methods

Time-based error functions offer another measure of error and can be used in training a neural model. The two primary measures used are inter-spike interval (ISI) and spike times. Comparison of ISI or spike times using a least mean squares error function can provide a fitness function for the genetic algorithm. The use of spike times is not ideal in this case because it is susceptible to phase shift errors.

#### F-I Curve

Equation 10 - F-I Error Function

A comparison of frequency-current (f-I) characteristics of a neuron can be used to train the model and determine parameters. The f-I characteristics are determined by detecting spikes and calculating the frequency at which a neuron fires in response to a given input current. The f-I values from the training data are compared with f-I values of the model using the normalized sum of squares algorithm seen in Equation 10. [25]

### Parameter Search Algorithm

The parameter space for each neuron class was searched using a genetic algorithm (GA). The purpose of the GA is to search the solution space to identify the best hypothesis which is defined as the hypothesis that optimizes the predefined fitness function.[26] In the case of this project, the fitness function is normalized sum of squares based on the f-I curve.

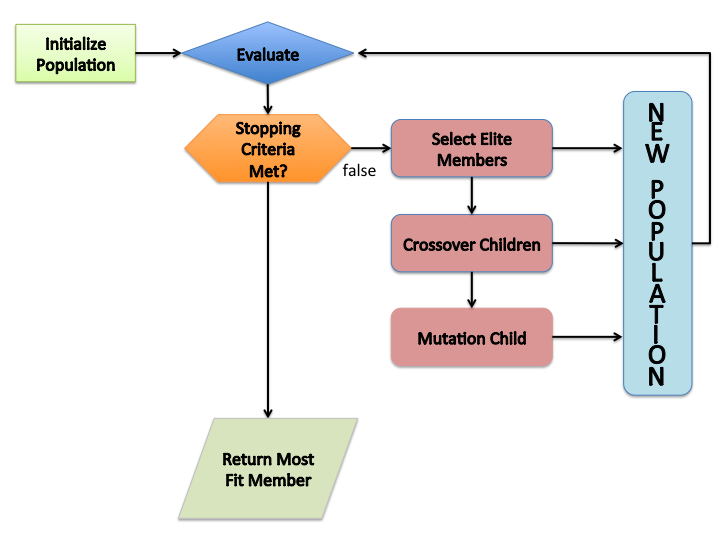
The algorithm creates an initial population at random. Each member of the population is a vector containing an estimated value for each of the parameters to be optimized. At each iteration of the algorithm, the current population is used to create the next population. The creation of the new population is accomplished by first computing the current population’s fitness function. Based on the results of each member of the populations fitness score, several members will be selected to become parents. The members selected will have the best fitness score. There will also be a few members of the population with lower fitness scores selected and labeled as *elite*. The *elite* members of the population pass on to the next generation without modification. However the parents will produce children by one of two methods, mutation or crossover.. Mutation adds a random vector from a Gaussian distribution to the parent to create the mutated child. Crossover children are created by combining pairs of parents in the population. At each coordinate of the child vector, one gene, or entry, is selected at the same coordinate from one of the two parents and passed on to the child. [27], [28] The current population is then replaced with the elite members and the children produced by the parents.

The breakdown of the population for the GA is shown in Table 1. Each member of the population has its own individual vector of parameters consisting of a value for each of the following Izhikevich parameters: *a, be, d, vr, vt, bi, Iapp, c, k*.

Table 1 - Breakdown of the genetic algorithm population

|  |  |  |  |
| --- | --- | --- | --- |
| Population Size | Elite Members | Crossovers | Mutated |
| 100 | 2 | 80 | 18 |

The primary stopping criteria will be a function tolerance of 1e-6. The function tolerance is essentially the change in the average value of the fitness function for the population between the current generation and the previous generation. A limit on the maximum number of generations will also be imposed. If the function tolerance has not been reached prior to 100 generations, the algorithm will be stopped.



**Figure 13 - Genetic Algorithm Flow Chart**

# Simulation

## Introduction

Several components were necessary to carry out the research. First the HH based network model by Rubin and Terman was replicated using C++. [4] The network model consists of 4 different neuron classes, which inherit from a parent class known as *Neuron*. This network model serves as the framework for generating training and test data. The C++ model was duplicated and modified to use Izhikevich neuron equations in place of HH equations. Therefore the network structure for both models is identical. These two network models will serve as the base for comparison of the completed network structure after training takes place.

The training process is carried out using MATLAB. The built-in GA calls on a fitness function. The fitness function consists of 2 components: 1) an internal function for simulating an Izhikevich neuron based on the parameters provided by the genetic algorithm, 2) the f-I curve SSE function. Upon determination of the optimal parameters for each type of neuron, the parameters are plugged into the C++ Izhikevich network model.

## Multi-Class Neuron Model

### Hodgkin Huxley Network Model

The primary focus of the research is to develop a multi-class neuron model using Izhikevich neurons. The multi-class neuron model was originally developed by Rubin and Terman [4], and has neurons from the STN, GPi, GPe, and TC. The original model uses HH models of each type of neuron. The model is capable of replicating a network of Parkinsonian neurons under the effects of DBS.

Equation 11

Equation 12

The model created in Rubin & Terman 2004 was originally written in XPP-aut. XPP-aut (X-window phase plane plus auto) is a powerful tool for solving differential equations, boundary value problems, and other mathematical equations. The first step of the research was to convert the model from XPPaut into C++. Although XPPaut is a powerful tool for computation, it is fairly slow. Other than runtime consideration, the model was converted to allow a larger degree of freedom in regards to data export, computational methods, and any additional modifications that may take place in future works. Four separate C++ class files were created; one for each type of neuron. Sixteen STN, sixteen GPi, sixteen GPe, and two TC neurons were instantiated and interconnected. The interconnection between neurons was specified as follows: each STN neuron received inhibitory input from two GPe neurons, each GPe neuron received excitatory input from three STN neurons and inhibitory input from two other GPe neurons, each GPi neuron received excitatory input from one STN neuron, and each thalamic neuron received inhibitory input from eight GPi neurons.

Each neuron receives synaptic input based on Equation 11 and Equation 12. The synaptic input current, , represents the input current from neuron α onto neuron β. The maximal synaptic conductance is represented by and the synaptic reversal potential is represented as . The synaptic strength is represented as and is defined by Equation 12. The Heaviside step function is denoted as , and and control the synaptic time courses.

All calculations were carried out using an adaptive step Runge Kutta algorithm from the GNU scientific library. [29] All results of the C++ model were validated against the XPPaut model. The C++ model was used to generate training and test data to train and test the Izhikevich model parameters.

### Izhikevich Network Model

The next step in the work was to modify the C++ model described in the previous section. The model described in 5.2.1 used HH neuron models for each type of neuron. Each of the HH models had to be replaced with Izhikevich neuron models. This provided a framework to test accuracy of an Izhikevich neural network model.

## Parameter Estimation

### Testing and Selection of Fitness Function

During the initial stages of research, PPTD was found to be incapable of training an Izhikevich model based on biological or HH data. The PPTD method was implemented as the fitness function for the GA. Initial tests had to be conducted to determine the optimal number of bins for the PPTD algorithm. The tests were carried out during iterative training of an STN neuron. The genetic algorithm had a population of 100 members, a maximum number of generations of 100, and a functional tolerance of 1e-6. The genetic algorithm was run 5 times for each individual number of bins. The number of bins for the PPTD algorithm tested was from 20x20 bins to 400x400 bins at increments of 20 bins. The STN neuron was trained with only one current profile and the current was kept constant at zero. The spike frequency of the training data was 3Hz. The parameter ranges tested for the STN neuron are shown in Table 2.

Table 2 - Parameter range for STN training during PPTD tests

|  |  |  |
| --- | --- | --- |
| Parameter | Lower Bound | Upper Bound |
| a | 0.001 | 0.2 |
| b | -5 | 20 |
| d | 0 | 200 |
| v­r | -80mV | -55mV |
| vt | -50mV | -25mV |
| k | 0.001 | 2.0 |
| I\_app | 0 | 500pA |

Every trial of the genetic algorithm generated parameters for the STN neuron that produced a firing frequency of zero and matched the resting potential of the training data. The results were due to Izhikevich model’s lack of a biologically realistic downstroke after a spike occurs. The Izhikevich model has an instantaneous reset when a spike occurs. Therefore when an Izhikevich neuron spikes, the downstroke is incapable of matching the shape of an HH neuron downstroke, which by definition takes finite time. The typical downstroke of a biological neuron is in the range of 2-5mS, which corresponds to 200 to 500 samples. In this case the derivative of the membrane potential with respect to time is drastically different between the HH and Izhikevich neurons. The difference in the derivative results in a large error value and therefore drives the genetic algorithm toward the replicating the resting potential of the neuron.

An example outlining the issues with PPTD are shown in Figure 14. A STN neuron is modeled using a manually tuned Izhikevich model (red) and a HH model (blue). In Figure 14a the voltage trace of both neurons can be seen. Both have a firing frequency of approximately 3Hz and similar spike shape, but if we look below in Figure 14b we can see the phase plane for the two different types of neurons are drastically different. The only overlap seen is in the resting state where the *dv/dt* is close to zero and the membrane potential is at the resting potential of 70mV. We can also notice the downstroke of the HH model, in between 0 and -100 dv/dt and -70 to -60mV, is a gradual process with each circle representing a sample from the HH data. However if we look at the Izhikevich neuron, there is only one point with a negative derivative, this is the instantaneous downstroke of the neuron after a spike.

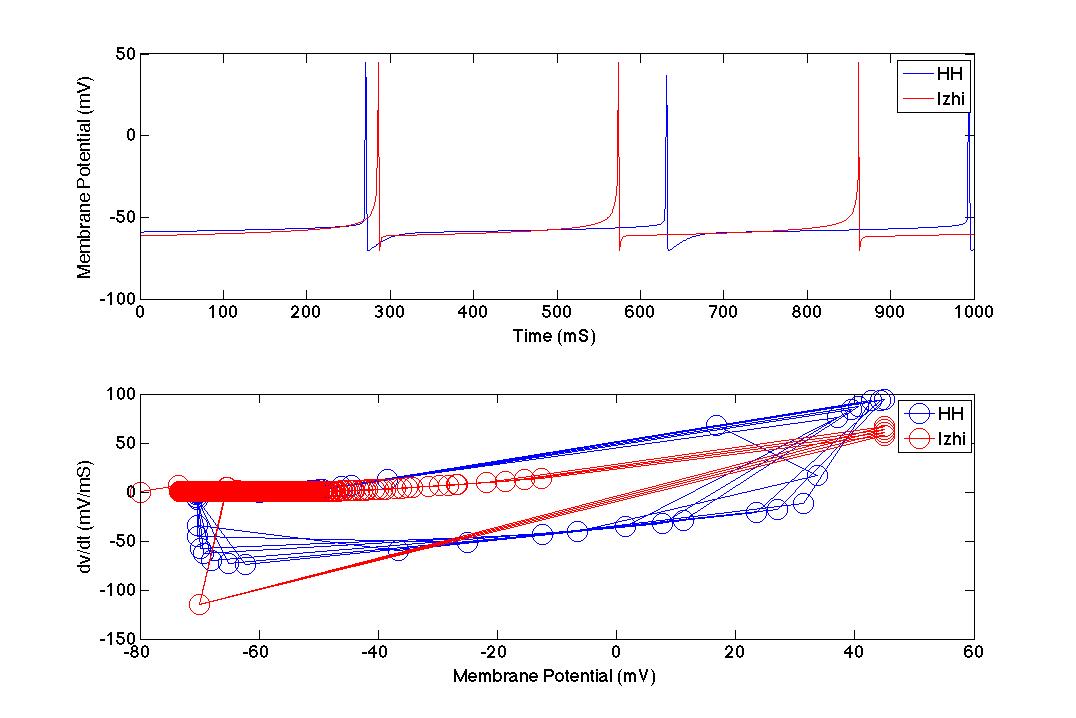


Figure 14 - Comparison of HH and Izhikevich phase plane a) membrane potential as a function of time, b) phase plane

After determining the PPTD algorithm was not suitable for training an Izhikevich neuron, the point-to-point membrane potential RMS method was implemented as the fitness function of the genetic algorithm. The same conditions as used during the PPTD tests were used to validate the point-to-point method. The only difference was the number of bins tested. When implementing the point-to-point method, the algorithm was discretized to prevent phase shift error and is shown in Equation 13. The point-to-point analysis only requires a one-dimensional histogram, so the number of bins was also one-dimensional. The range of the number of bins used in the algorithm ranged from 20 to 400 in increments of 20. Similar to the PPTD method, the point-to-point error function suffered from amplified error in the region of the spike downstroke. Training of the Izhikevich neuron parameters resulted in parameters that would be capable of only reproducing the resting potential of the HH training data. This was caused by the increased error during each spike due to the difference in spiking downstroke shape and membrane potential value.

Equation 13 - Discretized point-to-point error function

The PPTD and point-to-point membrane potential error functions were both ruled out as optimal methods for training the Izhikevich model. The above-mentioned methods are based on the membrane potential and therefore provide an accurate representation of spike shape when used to tune a HH neuron model. However both methods were found to be less than satisfactory for training an Izhikevich neuron.

The f-I curve based fitness function was chosen to carry out the training of the Izhikevich model. The Izhikevich neuron is reset to a resting voltage each time a spike is detected. A simple spike counter was implemented to record the number of spikes in a predetermined period. The number of spikes in the period is compared with the number of spikes in the training data in the same time window. The ISI method could also be used, but would additionally require ISI calculations for each member of the population at each generation of the genetic algorithm. Therefore the f-I model was the optimal choice for the fitness function.

### Genetic Algorithm and Estimation of Izhikevich Parameters

Determining the optimal parameters for the STN, GPe, and TC Izhikevich neurons was the next step in the research. The GPi neurons have the same characteristics as the GPe neurons with the exception of a constant applied current. Therefore the GPe and GPi neurons only required one set of parameters to be estimated. Determination of the optimal parameters was carried out in MATLAB. The genetic algorithm supplied in the global optimization toolbox was used. The fitness function used by the GA was an SSE function, which calculates error based on the firing rate of a neuron as a function of the input current the neuron receives.

During each iteration of the GA an estimated set of parameters for the Izhikevich neuron, along with a given input current, will be used to generate membrane potential data. Spikes detected during data generation will be stored. The number of spikes during the one-second training simulation will be compared to one second of the training data using the fitness function previously described.

The training and test input current values for each type of neuron was based on the range of input current values each neuron would see when connected in the network. These ranges were determined from the previous works of Rubin and Terman. [4]

### TC Parameter Estimation

The parameter estimation for the TC neurons was carried out using eight different f-I data values. Six of the f-I values were based on constant excitatory input. The TC cell fires at a constant frequency when receiving excitatory input. The other two values were based on the specified post-inhibitory characteristics of a TC cell.

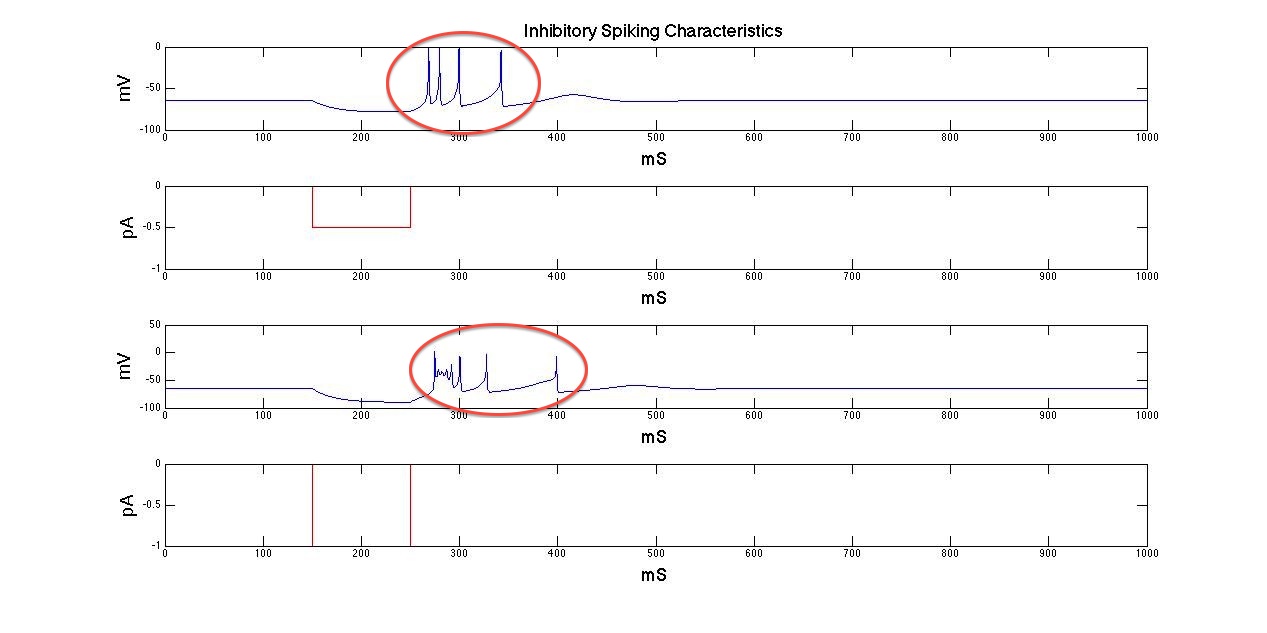


Figure 15 - Inhibitory spiking characteristics of a TC neuron (Training Data)

The training data is generated using the Rubin and Terman HH model in C++. Each f-I training data trial consists of 1000mS of membrane potential data, 1000mS of input current data, and the spike frequency of the trial. The input current profile consists of the input current value at each time step. All Izhikevich neurons, including the TC neuron, are trained using a time step of 0.01mS.

Table 3 - Training data and settings (TC)

|  |  |  |
| --- | --- | --- |
| Firing Count | Length of Trial | Input Profile |
| 4 | 1000mS | -0.5pA for 100ms, 0pA 100mS-1000mS |
| 4 | 1000mS | -1.0pA for 100ms, 0pA 100mS-1000mS |
| 0 | 1000mS | 0pA – constant |
| 62 | 1000mS | 2pA – constant |
| 103 | 1000mS | 4pA – constant |
| 133 | 1000mS | 6pA – constant |
| 155 | 1000mS | 8pA – constant |
| 174 | 1000mS | 10pA – constant |
| 4 | 500mS | -0.5pA for 100ms, 0pA 100mS-1000mS |
| 4 | 500mS | -1.0pA for 100ms, 0pA 100mS-1000mS |

Each generation of the GA produce estimations of the Izhikevich parameters. Using the estimated parameters and the input current from the training set, ten different trials are carried out. The training data settings and values can be seen in Table 3.

Table 4 - Estimated Parameters (TC)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | be | vr (mv) | vt (mV) | Iapp (pA) | d | k | c (mV) | bi |
| 0.009 | -1.276 | -51.048 | -43.289 | 3.826 | 15.160 | 2.000 | -68.991 | 49.997 |

The reason for running 2 separate trials with the inhibitory data is to provide an accurate estimation of post-inhibitory spiking behavior. The first set of trials using the inhibitory data counts the total number of spikes in the 1000mS trial. However, the second set of trials only looks at the first 500mS. The first 500mS is the only place the post-inhibitory spikes should occur as can be seen by the circled areas in Figure 15. Therefore if the number of spikes occurring during the 500mS window is not equal to the total number of spikes during the 1000mS trial, the estimated parameters are not capable of replicating the post-inhibitory spiking characteristics of the training data and should be ruled out.

Table 5 - Test Settings and results (TC)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Firing Count (HH) | Firing Count (Izhi) | Length of Trial | Input Profile | SSE value |
| 5 | 0 | 1000mS | -0.75pA (0-100ms) 0pA (900-1000mS) |
| 1 | 0 | 1000mS | -0.25pA (0-100ms) 0pA (900-1000mS) |
| 35 | 33 | 1000mS | 1pA – constant | 176.38 (test set) |
| 84 | 86 | 1000mS | 3pA – constant |
| 119 | 123 | 1000mS | 5pA – constant |
| 145 | 155 | 1000mS | 7pA – constant |  |
| 165 | 184 | 1000mS | 9pA – constant |
| 182 | 212 | 1000mS | 11pA – constant |

One hundred individual trials of the genetic algorithm were carried out using the same settings, such as upper and lower bounds, for each trial. The algorithm was run with a population size of 100 and produced 100 generations of estimates. The parameters estimated by the genetic algorithm can be seen Table 4. Table 5 shows the f-I data sets and conditions used during the testing phase. Once the parameters for the TC neuron had been determined, the neuron was tested and compared to the TC test data.

Figure 16 displays the results for each individual test set. We can easily see the learned parameters were incapable of replicating the post-inhibitory spiking behavior seen in the two inhibitory test sets. However we can see the parameters were capable of reproducing the spiking frequency trend present with all the excitatory test sets.

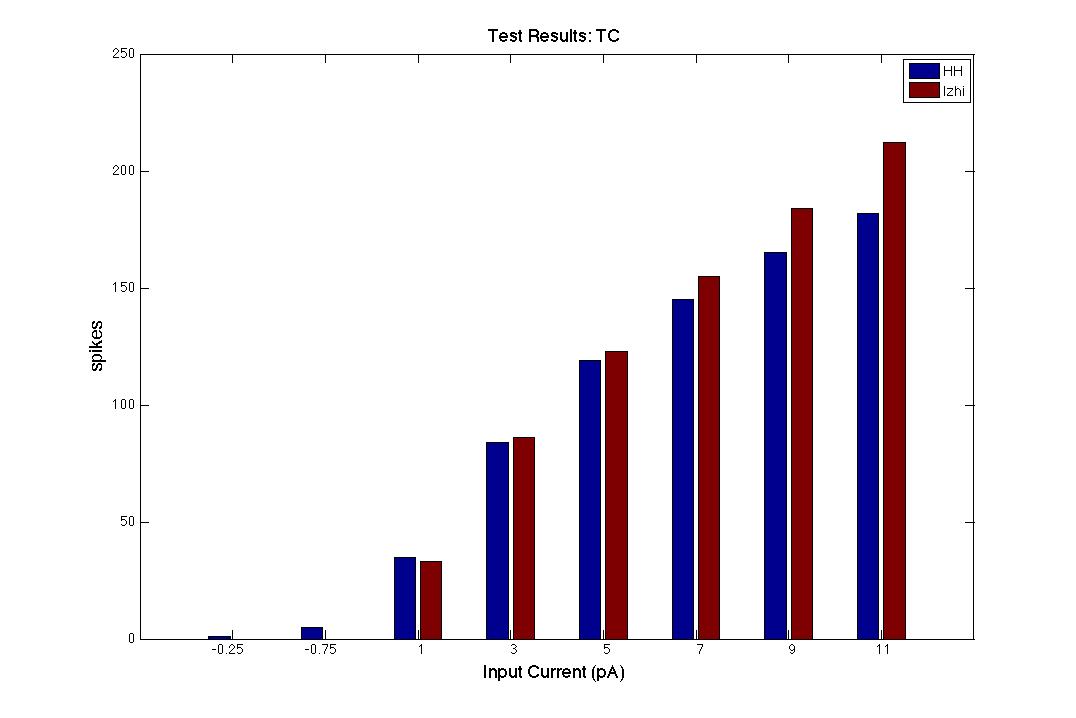


Figure 16 - Number of spikes for each f-I test (TC)

### STN Parameter Estimation

The parameters for the STN neuron were determined using a training set consisting of 6 excitatory f-I series and 2 inhibitory f-I series. The input currents were selected based on the normal input range of the neuron as shown in Rubin and Terman’s work. [4] Windowing was not necessary during the inhibitory training phase as was used with during the TC neuron training. This was due to the fact that the inhibitory current was present for the first 500mS of the 1000mS run, which inhibited any spiking. Therefore the only possible spiking that would occur during the inhibitory f-I training series was post-inhibitory.

Table 6 - Training data and settings (STN)

|  |  |  |
| --- | --- | --- |
| Firing Count | Length of Trial | Input Profile |
| 12 | 1000mS | -30pA 500mS, 0pA 500mS |
| 13 | 1000mS | -50pA 500mS, 0pA 500mS |
| 3 | 1000mS | 0pA – constant |
| 14 | 1000mS | 10pA – constant |
| 28 | 1000mS | 20pA – constant |
| 41 | 1000mS | 30pA – constant |
| 54 | 1000mS | 40pA – constant |
| 66 | 1000mS | 50pA – constant |

Table 7 - Estimated parameters (STN)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | be | vr (mv) | vt (mV) | Iapp (pA) | d | k | c (mV) | bi |
| 0.005 | 0.405 | -58.093 | -21.000 | 0.7824 | 150.000 | 0.519 | -55.000 | 67.2683 |

The parameters for the STN neuron were determined after one hundred trials of the genetic algorithm were run. The genetic algorithm used a population size of 100 and a maximum of 100 generations could be reached if the minimum functional tolerance of 1e-6 did not occur first. The set of parameters exhibiting the minimum error value during the training trials were used in the testing phase and network model. The learned parameters can be seen in Table 7.

Table 8 - Test settings and results (STN)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Firing Count (HH) | Firing Count (Izhi) | Length of Trial | Input Profile | SSE Error Value |
| 3 | 4 | 1000mS | -20pA 500mS, 0pA 500mS |
| 13 | 8 | 1000mS | -40pA 500mS, 0pA 500mS |
| 6 | 4 | 1000mS | 5pA – constant | 18.375 (test set) |
| 21 | 20 | 1000mS | 15pA – constant |
| 35 | 34 | 1000mS | 25pA – constant |
| 47 | 50 | 1000mS | 35pA – constant |  |
| 60 | 65 | 1000mS | 45pA – constant |
| 72 | 81 | 1000mS | 55pA – constant |

After determining the parameters for the Izhikevich neuron, the STN neuron was tested. The test set contained 8 different f-I data series to determine the error of the estimated parameters. As with the training data, the test set consisted of two inhibitory f-I series and six f-I excitatory series. The test settings, spiking frequencies, and error values can be seen in Table 8, Figure 17.

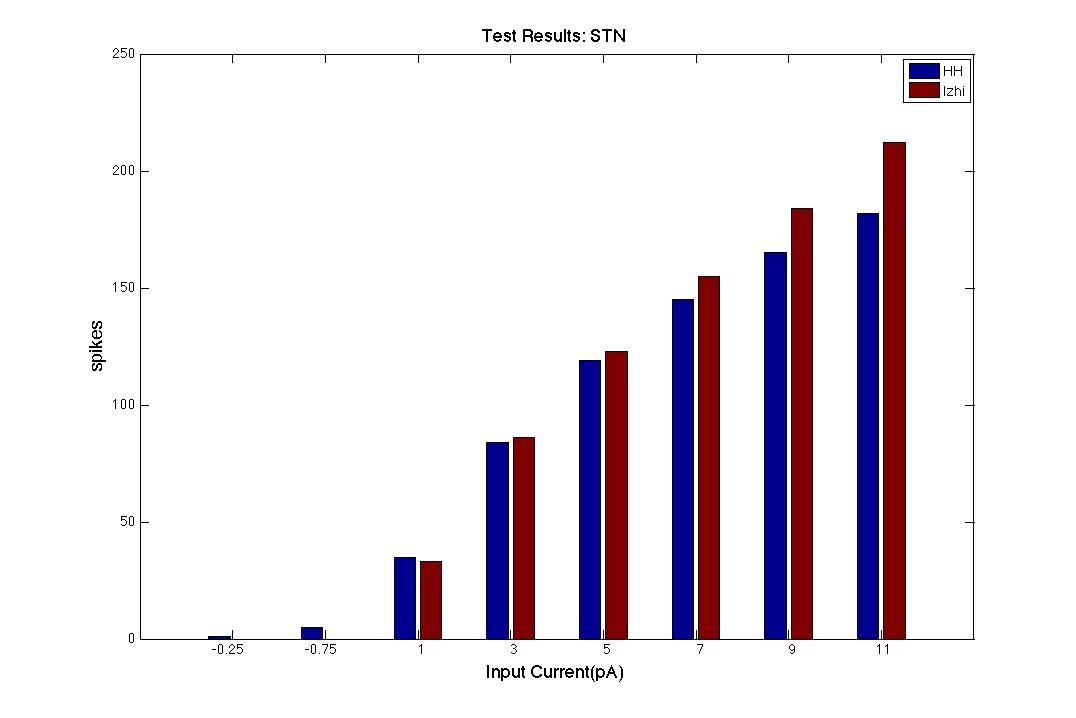


Figure 17 - Test results (STN)

### GPe Parameter Estimation

Table 9 - Training settings (GPe)

|  |  |  |
| --- | --- | --- |
| Firing Count | Length of Trial | Input Profile |
| 0 | 1000mS | -0.5pA constant |
| 0 | 1000mS | -1.0pA constant |
| 13 | 1000mS | 0pA – constant |
| 44 | 1000mS | 1pA – constant |
| 57 | 1000mS | 2pA – constant |
| 67 | 1000mS | 3pA – constant |
| 76 | 1000mS | 4pA – constant |
| 85 | 1000mS | 5pA – constant |
| 13 | 300mS | 0pA -constant |

Parameter estimation for the GPe/GPi neuron was carried out using 8 different f-I data values. The training data set contains two inhibitory f-I sets and six excitatory f-I sets. The estimated parameters are tested twice for the zero input f-I training set. The first test checks the number of spikes for the complete trial time of 1000mS and the second test checks the number of spikes in a 300ms window. The spiking pattern of the GPe neuron with zero input current can be seen circled in Figure 18. The training conditions are outlined in Table 9.

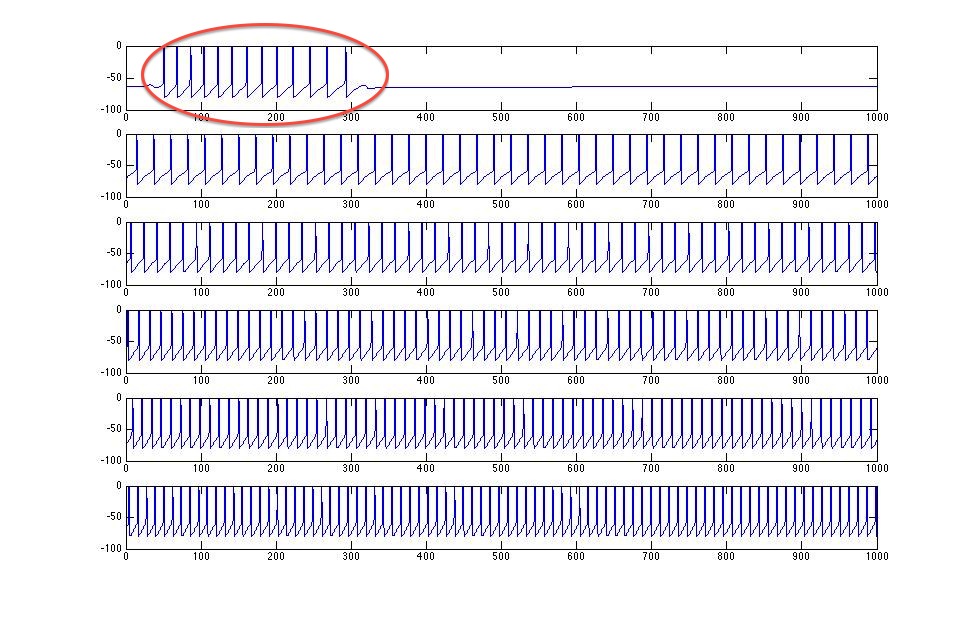


Figure 18 - GPe/GPi Training Data (top to bottom 0pA, 1pA, 2pA, 3pA, 4pA, 5pA input)

Table 10 - Estimated parameters (GPe)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | be | vr (mv) | vt (mV) | Iapp (pA) | d | k | c (mV) | bi |
| 0.02 | 0.17 | -48.44 | -25.20 | 116.48 | 42.01 | 0.68 | -69.75 | 50 |

Table 11 - Test settings and results (GPe)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Firing Count (HH) | Firing Count (Izhi) | Length of Trial | Input Profile | SSE Value |
| 0 | 0 | 1000mS | -0.75pA constant |
| 11 | 7 | 1000mS | -0.25pA constant |
| 28 | 29 | 1000mS | 0.5pA – constant | 17.35 (test set) |
| 50 | 46 | 1000mS | 1.5pA – constant |
| 62 | 59 | 1000mS | 2.5pA – constant |
| 72 | 72 | 1000mS | 3.5pA – constant |  |
| 81 | 85 | 1000mS | 4.5pA – constant |
| 88 | 97 | 1000mS | 5.5pA – constant |

One hundred trials of the genetic algorithm were run with a population size of 100 for 100 generations and a minimum functional tolerance value of 1e-6. The optimal values for the GPe/GPi neuron are listed inTable 10. The parameters determined during the training phase were then tested using the test conditions outlined in Table 11. The test data produced via the HH model is shown in Figure 19. The following input currents were used during testing: -0.75, -0.25, 0.5, 1.5, 2.5, 3.5, 4.5, 5.5 pA. The input current stayed constant during all simulations for the 1000mS duration.

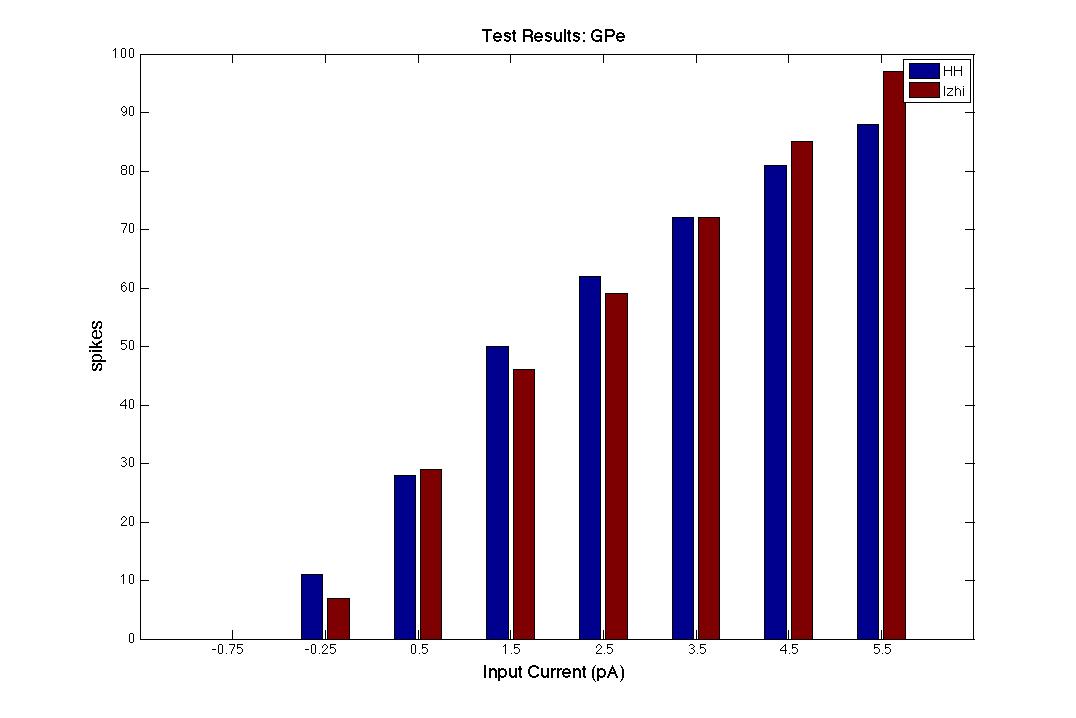


Figure 19 - Number of spikes for each f-I test set (GPe)

In Figure 19 the results of each individual test can be viewed. The GPe model was capable of mimicking the firing frequency with less than 10% error for all test settings with the exception of f-I data set using the inhibitory input of -0.25pA. However if we view the spike frequency comparisons inFigure 19, we can see the GPe neuron followed the same firing frequency trend as the test data.

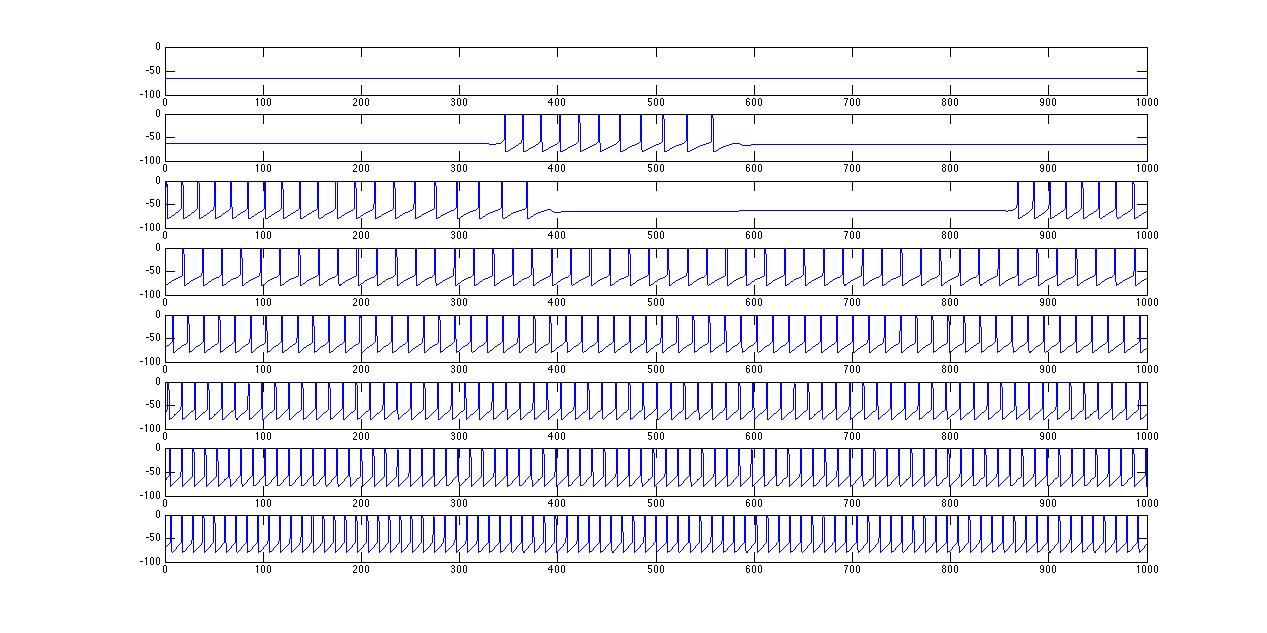


Figure 20 – GPe test data (HH) [input current from top to bottom: -0.75, -0.25, 1, 3, 5, 7, 9pA]

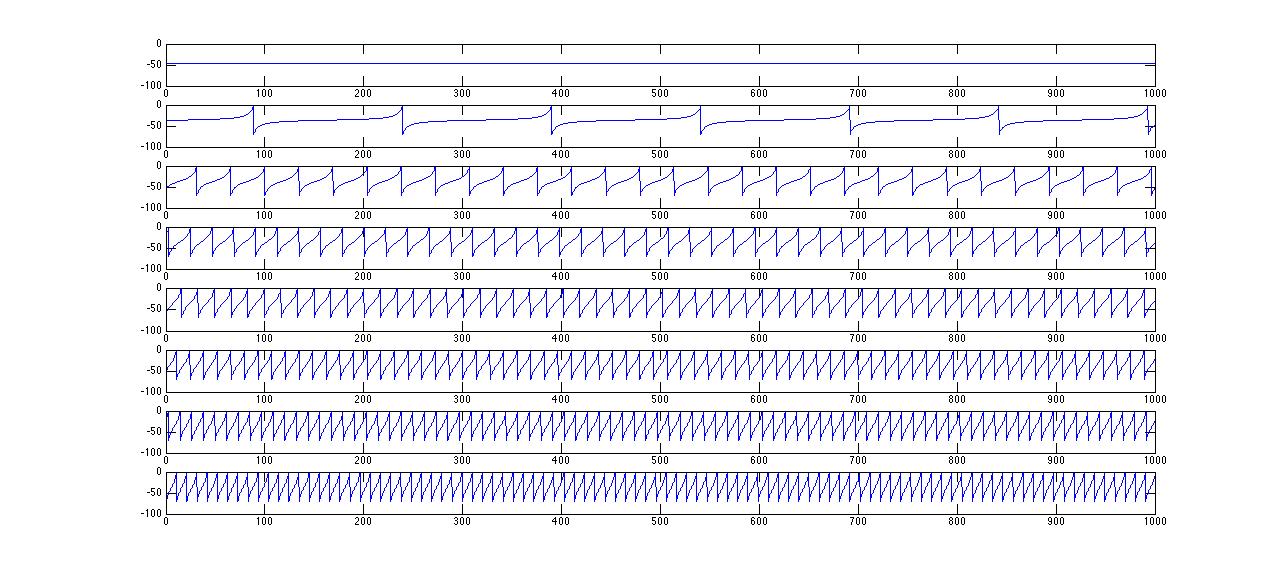


Figure 21 – GPe test data (Izhikevich) [input current from top to bottom: -0.75, -0.25, 1, 3, 5, 7, 9pA]

The GPe Izhikevich neuron was capable of mimicking the HH test model’s firing frequency, but at the cost of spike waveform shape. In Figure 20 the test data generated with the HH version of the GPe can be viewed. In Figure 21 the data generated with the Izhikevich model using the same inputs as in Figure 20 can be seen. Several observations can be made. First we can easily see the tonic spiking shown in the -0.25pA test data is not replicated in the Izhikevich model. Instead the Izhikevich model shows a constant firing frequency throughout the entire -0.25pA trial. There is also a distinct difference in between the spike shape of the two models. In Figure 21 the Izhikevich model exhibits much slower upstrokes than the Izhikevich model, as well as sharper down strokes. The Izhikevich model has a higher resting potential than the HH model as well due to the trained “vr” value of approximately -49mV.

One method to alleviate the difference in resting potential is to set different upper and lower bounds on the value of “vr”. When running the GA trials for the GPe, the upper and lower bound values of “vr” and “vt” were set to a wide range to allow for a large parameter search space. The spiking characteristics of the neuron were of higher priority than the resting potential value. However the range of the bounds should be reduced to allow for optimal waveform fit.

## Multi-Class Model Testing and Results

The parameters for each type of Izhikevich neuron were determined and the multi-class model was modified to include the parameters. After the modifications were completed, the model was tested. To ensure the parameters had been modified correctly, each class of neuron was tested individually with no network connection. After confirming each class of neuron was operational, a simulation of the network was carried out.

Upon running the complete network model the results were analyzed and compared to the HH model. Both models were run for 2000mS and the data generated between 1000mS and 2000mS is analyzed. The models were run in a healthy state, free of any DBS input, and only the TC neurons received an input from outside the neural network from the sensory motor cortex. The results from the original network model developed by Rubin and Terman can be seen in Figure 23. The network model using the Izhikevich neurons with the estimated parameters can be seen in Figure 22. It can easily be verified that the Izhikevich neurons do not replicate the results from the Rubin and Terman model.

The primary focus of Rubin and Terman model was to show that inputs from the sensory-motor cortex into the TC cells would be relayed as an output of the TC cells in a healthy brain. In a brain with Parkinson’s disease these cells will not be able relay the output. However if supplied with DBS, the TC cells would regain the ability to relay the signal from the sensory-motor cortex. We can easily see in the top right graph of Figure 22, that the TC neuron never fires. However we can see the periodic pulses in the TC membrane potential due to the input from the sensory-motor cortex. Similar to the TC cells, the STN and GPe cells of the Izhikevich network never fire when interconnected into the network. This behavior clearly differs from the behavior shown in Figure 23.

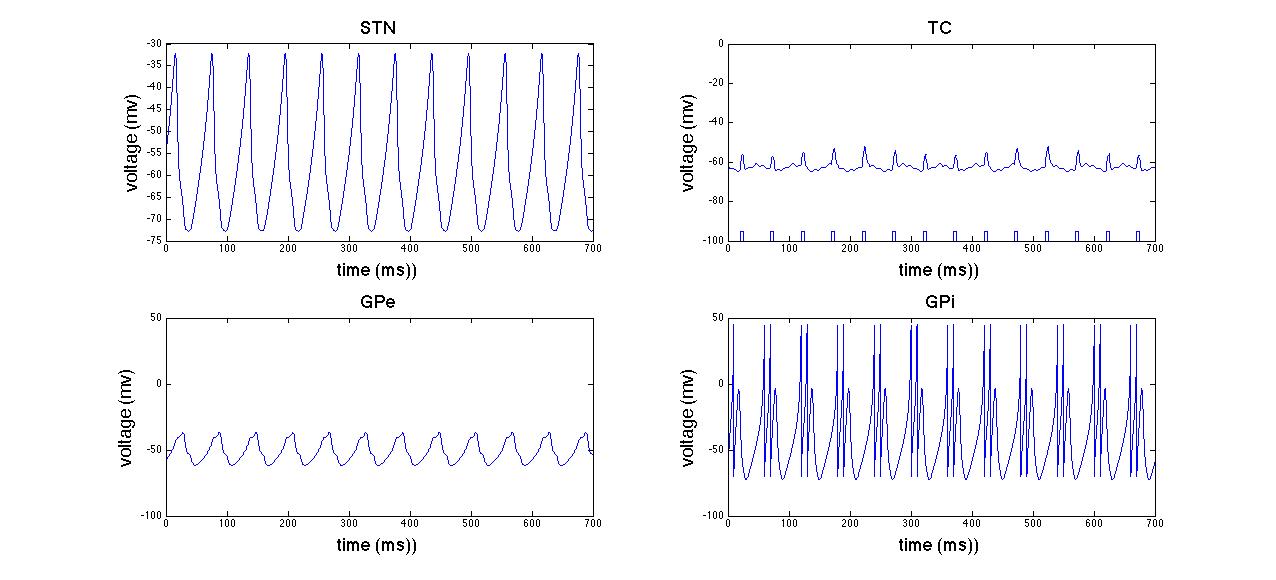


Figure 22 - Network snapshot using the Izhikevich model and estimated parameters in a healthy state

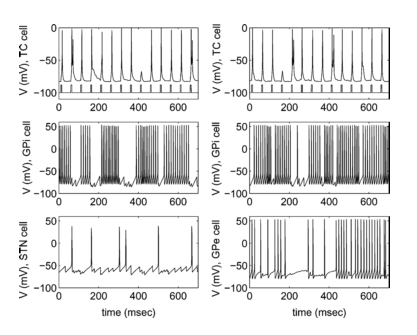


Figure 23 - Multi-class network model snapshot using HH neurons in healthy state

## Discussion of Network Model Results

There are several reasons the interconnected Izhikevich model is incapable of replicating the original HH model. The network model is extremely sensitive as was seen in the preliminary work. When running the model using HH neurons, a time step of 1e-6 in conjunction with a 4th order Runge Kutta algorithm was necessary to replicate Rubin and Terman’s model. The process was sped up using an adaptive 4/5 Runge Kutta algorithm, but still required a minimum error between iterations of 1e-12. The sensitivity of the neuron behavior is only amplified when interconnected in the network. If one neuron type is not fitted properly, the entire network is affected and starts to malfunction. This makes it relatively impossible to determine which neuron type is causing the issues in the network.

The Izhikevich neuron model has a base version as previously described. For each neuron the same version of the Izhikevich neuron was used and only the parameters of the base model were tuned. For each neuron type only two possible values of “b” were determined during the training process. Izhikevich suggests [18] the use of multiple different vectors of parameters that can be used under different operating conditions. If different parameter vectors are learned for several different input current and membrane potential settings, this may allow the Izhikevich network model to accurately replicate the HH network model. However there is currently no method, outside of manual tuning, for setting the conditions when different vectors should be used.

Another possible way to decrease error between the Izhikevich and HH models is to alter the Izhikevich neuron model. The base version of the Izhikevich neuron model was shown in Equation 1, Equation 2, and Equation 3. However modifications can be made to the model to help tune the spiking characteristics and waveform. For example in Equation 14 we can see the highlighted changes in the original Izhikevich model. This modified version takes into account the value of the recovery variable to aid in scaling both the spiking threshold and the membrane reset value after a spike has fired. Modifications such as the one seen in Equation 14 can help fine-tune the behavior and spike shape of each type of neuron and may provide another method to eliminate the error between the HH models and Izhikevich model. The primary issue with these modifications is that the modifications appear to be made manually. Izhikevich offers no foundation as to why or how a model such as the one seen in Equation 14 is developed. [18]

Equation 14 - Modified Izhikevich Model

The spiking characteristics of a neuron are extremely complex. Even thought the Izhikevich neuron is capable of replicating a plethora of spiking mechanisms, it is not possible to train a neuron under every possible condition. The trials carried out to train the four types of neurons only contained two basic settings. For the TC and STN neurons, the neurons were trained using constant excitatory input, which produces constant firing and using long periods of inhibition, which should result in post-inhibitory spiking. The GPe/GPi neuron was trained using data from trials using constant excitatory input resulting in constant firing frequencies, as well as a constant inhibitory input, resulting in either a constant firing frequency or a lack of spiking activity all together. From the individual neuron test results we can see none of the neurons were capable of reproducing anything other than constant firing patterns from excitatory input with any degree of accuracy. It is clear that the model must be modified to promote post-inhibitory spiking patterns seen in the TC and STN cells, as well as tonic firing patterns seen in the GPi.

## Computational speed/resource comparison

An investigation to the potential benefits of using an Izhikevich neuron model in place of an HH model was carried out. The attempt to replicate the work of Rubin and Terman using the Izhikevich model was unsuccessful. However the potential benefit of using the Izhikevich model, if properly trained, is investigated.

The run-time of the Izhikevich network model was compared with the HH network model. The primary advantage of using the Izhikevich neurons is that the model only consists of two differential equations in comparison to the HH model, which uses a minimum of 4 differential equations. The HH model also has additional calculation necessary for several variables and is extremely sensitive to time step size.

The results in Figure 24 show the average run time over 10 trials for each setting. The trials were carried out on a Macbook Pro with a 2.4 GHz Intel i5 processor. During the preliminary phases of research the Rubin and Terman model was replicated using C++. It was found that an adaptive Runge Kutta algorithm offered the best trade off of speed and accuracy. In Figure 24 we can see the average run times for the Rubin and Terman network model to produce 100mS of simulated data.

The HH model using a fixed time step 4th order Runge Kutta method exhibits an inverse relationship between run-time and time step size. Although smaller time steps will allow for shorter run-times, the accuracy of any trial using a times step larger than 1e-6mS can result in loss of spiking characteristics for HH models. Therefore it is necessary to use an adaptive Runge Kutta algorithm. The adaptive algorithm preserves spiking shape, characteristics, and provides a run time on the same order of magnitude as a fixed Runge Kutta with a time step of 0.01mS.

The Izhikevich model shares the same inverse relationship between run time and step size as the HH model. However the Izhikevich model in much faster in comparison to the HH model in all trials. It must be noted that the Izhikevich tests were carried out using a forward Euler method, which is far less complex than the 4th order Runge Kutta method. However the Izhikevich model has been shown to provide spiking behaviors while using the forward Euler method and a time step of 0.01mS. [18]

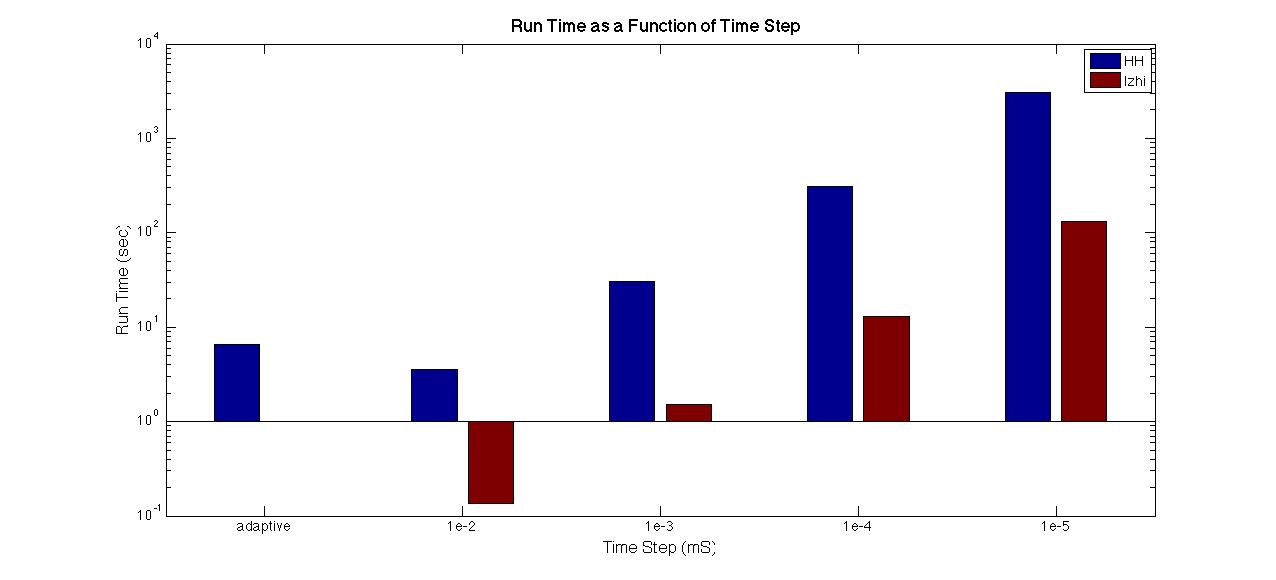


Figure 24 - Run time analysis

# Conclusion

The primary goal of the proposed work was to replicate the work of Rubin and Terman using the Izhikevich neuron in place of the HH neuron. This task was to be carried out by training 4 different types of neurons using a genetic algorithm and the PPTD fitness function. The overall outcome of the experiment was unsuccessful. The Izhikevich network model failed to replicate the HH network model.

During the course of the research, it was found that the PPTD algorithm is incapable of training the Izhikevich neuron based on biological membrane potential data or data derived from the HH model. The PPTD algorithm looks at the membrane potential and the derivative of the membrane potential at each time step. Since the Izhikevich neuron has an instantaneous reset, the downstroke of a spike does not match the spike of a real neuron. This causes a large error to be induced during the training process any time a spike occurs.

After further research it was found that the f-I curve of a neuron would be the optimal measure for the fitness function of the genetic algorithm. The f-I curve fitness function offered mixed results. During tests when constant excitatory current was supplied to the neuron the constant firing frequency of each type of neuron was replicated with less than 15% error. However the f-I curve error function failed to replicate the post-inhibitory bursting patterns of the STN and TC neurons as well as the tonic firing pattern of the GPe and GPi neurons. When the learned parameters were implemented in the Izhikevich network model, the TC neurons failed to relay the sensory-motor input signals.

The Izhikevich neuron should not be completely ruled out as a computationally efficient replacement for the HH model. Additional changes can be made to both the training procedure and the Izhikevich model structure to possibly allow for replication of the HH model. Larger training sets could be used to test the neuron over a wider range of input current values and settings. Settings include: neuronal response after short periods of inhibition, prolonged periods of inhibitions, inhibitions followed by excitation, etc. The training procedure could also determine several parameter vectors. Each set of parameters would only apply to specific settings. The Izhikevich model can also be altered in various ways as previously described in 5.4.

Although the Izhikevich neuron may be a viable option for replacing the HH neuron model, it seems as though the HH model is still the optimal choice for small network simulations. Although it is computationally complex, it offers a degree of accuracy capable of replicating biological recordings. Coupled with an adaptive algorithm, the HH model can provide both accuracy and run-times that are on par with the Izhikevich neuron when replicating small networks. However the need for a computationally efficient model such as the Izhikevich model is still present for large-scale network simulations.

# References

[1] C. Hammond, H. Bergman, and P. Brown, “Pathological synchronization in Parkinson’s disease: networks, models and treatments.,” *Trends in neurosciences*, vol. 30, no. 7, pp. 357–64, Jul. 2007.

[2] A. Beuter and J. Modolo, “Delayed and lasting effects of deep brain stimulation on locomotion in Parkinson’s disease.,” *Chaos (Woodbury, N.Y.)*, vol. 19, no. 2, p. 026114, Jun. 2009.

[3] Y. Guo, J. E. Rubin, C. C. McIntyre, J. L. Vitek, and D. Terman, “Thalamocortical relay fidelity varies across subthalamic nucleus deep brain stimulation protocols in a data-driven computational model.,” *Journal of neurophysiology*, vol. 99, no. 3, pp. 1477–92, Mar. 2008.

[4] J. E. Rubin and D. Terman, “High frequency stimulation of the subthalamic nucleus eliminates pathological thalamic rhythmicity in a computational model.,” *Journal of computational neuroscience*, vol. 16, no. 3, pp. 211–35, 2004.

[5] M. S. Okun, “Deep-brain stimulation for Parkinson’s disease.,” *The New England journal of medicine*, vol. 367, no. 16, pp. 1529–38, Oct. 2012.

[6] P. Brown and D. Williams, “Basal ganglia local field potential activity: character and functional significance in the human.,” *Clinical neurophysiology : official journal of the International Federation of Clinical Neurophysiology*, vol. 116, no. 11, pp. 2510–9, Nov. 2005.

[7] A. Kent and W. Grill, “Instrumentation to record evoked potentials for closed-loop control of deep brain stimulation,” *… in Medicine and Biology Society, EMBC, …*, 2011.

[8] A. Pascual, J. Modolo, and A. Beuter, “Is a computational model useful to understand the effect of deep brain stimulation in Parkinson’s disease?,” *Journal of integrative neuroscience*, vol. 5, no. 4, pp. 541–59, Dec. 2006.

[9] C. a Davie, “A review of Parkinson’s disease.,” *British medical bulletin*, vol. 86, pp. 109–27, Jan. 2008.

[10] P. Pollak, P. Krack, E. Moro, A. Mendes, S. Chabardes, and A. Koudsie, “Treatment Results : Parkinson ’ s Disease,” vol. 17, 2002.

[11] J. Obeso, C. Olanow, and M. Rodriguez-Oroz, “Deep-brain stimulation of the subthalamic nucleus or the pars interna of the globus pallidus in Parkinson’s disease,” *N Engl J Med*, vol. 345, no. 13, pp. 956–63, Sep. 2001.

[12] “Parkinson’s Disease.” [Online]. Available: http://biomed.brown.edu/Courses/BI108/BI108\_2008\_Groups/group07/Parkinsons.html.

[13] A. L. Benabid, A. Koudsié, A. Benazzouz, L. Vercueil, V. Fraix, S. Chabardes, J. F. LeBas, and P. Pollak, “Deep brain stimulation of the corpus luysi (subthalamic nucleus) and other targets in Parkinson’s disease. Extension to new indications such as dystonia and epilepsy,” *Journal of Neurology*, vol. 248, no. S3, pp. 37–47, Sep. 2001.

[14] C. W. Olanow, M. F. Brin, and J. A. Obeso, “The role of deep brain stimulation as a surgical treatment for Parkinson’s disease.,” *Neurology*, vol. 55, no. 12 Suppl 6, pp. S60–6, Jan. 2000.

[15] A. Hodgkin and A. Huxley, “A Quantitive Description of Membrane Current and Its Application to Conduction and Excitation in Nerve,” *The Journal of physiology*, pp. 500–544, 1952.

[16] D. Terman, J. E. Rubin, a C. Yew, and C. J. Wilson, “Activity patterns in a model for the subthalamopallidal network of the basal ganglia.,” *The Journal of neuroscience : the official journal of the Society for Neuroscience*, vol. 22, no. 7, pp. 2963–76, Apr. 2002.

[17] E. M. Izhikevich, “Simple model of spiking neurons.,” *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, vol. 14, no. 6, pp. 1569–72, Jan. 2003.

[18] E. M. Izhikevich, *Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting*, vol. First. MIT Press, 2007, p. 441.

[19] M. Dur-e-Ahmad, W. Nicola, S. A. Campbell, and F. K. Skinner, “Network bursting using experimentally constrained single compartment CA3 hippocampal neuron models with adaptation.,” *Journal of computational neuroscience*, vol. 33, no. 1, pp. 21–40, Aug. 2012.

[20] G. Kumar, V. Aggarwal, N. V Thakor, M. H. Schieber, and M. V Kothare, “Optimal parameter estimation of the Izhikevich single neuron model using experimental inter-spike interval ( ISI ) data,” *System*, pp. 3586–3591, 2010.

[21] K. P. Michmizos and K. S. Nikita, “Addition of deep brain stimulation signal to a local field potential driven Izhikevich model masks the pathological firing pattern of an STN neuron.,” *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, vol. 2011, pp. 7290–3, Aug. 2011.

[22] D. Terman and E. Izhikevich, “State space,” *Scholarpedia*, vol. 3, no. 3, p. 1924, 2008.

[23] P. Achard and E. De Schutter, “Complex parameter landscape for a complex neuron model.,” *PLoS computational biology*, vol. 2, no. 7, p. e94, Jul. 2006.

[24] W. Van Geit, E. De Schutter, and P. Achard, “Automated neuron model optimization techniques: a review.,” *Biological cybernetics*, vol. 99, no. 4–5, pp. 241–51, Nov. 2008.

[25] W. Gerken, L. Purvis, and R. Butera, “Genetic algorithm for optimization and specification of a neuron model.,” *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, vol. 4, no. Ml, pp. 4321–3, Jan. 2005.

[26] T. M. Mitchell, *Machine Learning*, Internatio. Singapore: McGraw-Hill Book Co., 1997, p. 414.

[27] “Global Optimization Toolbox User ’s Guide.” Natick, MA, p. 581, 2012.

[28] T. M. Mitchell, *Machine Learning*, Internatio. Singapore: McGraw-Hill Book Co., 1997, p. 414.

[29] M. Galassi, J. Davies, J. Theiler, and B. Gough, *GNU scientific library : Reference Manual*, 1.15 ed., no. April. 2002, p. 523.

# APPENDIX

## GPi (GPe) Neuron

The GPi and GPe neurons will be modeled in a nearly identical manner with the exception of their synaptic input currents. The equations describing GPi and GPe functionality are listed below in Equations 15 - 23.

Equation 15 - Membrane Voltage Equation for GPe/GPi

In addition to the potassium, sodium, and leakage currents, the model takes into account low-threshold T-type calcium current (It), calcium current (­ICa), synaptic input from the STN (), and an input from other GPi (GPe) neurons (. There is also an applied voltage (Iapp), which represents a constant input from the striatum. The conductance values for each ion channel are represented by the *gx* variables in each current equation. The slowly operating gating variables *n, h,* and *r* are a function of both time and voltage and are governed by Equations 21-23 (*X* = n, h, r).

Equation 16

Equation 17

Equation 18

Equation 19

Equation 20

Equation 21 - Gating variable (function of voltage)

Equation 22 - Time constant for gating variable

Equation 23 - Steady State Voltage Dependence for Gating Variables

The steady state voltage dependence for each gating variable (*X* = *n, m, h, a, r, or s*) is determined by Equation 23.

## STN

The membrane potential for the STN neuron is defined in Equation 24. The STN neuron has the same currents with the exception of synaptic input, applied current, and the DBS input. The synaptic input to the STN neurons comes from the GPe (. The DBS input will be discussed later sections. When the model is running without DBS, this current can be ignored.

Equation 24 - STN Membrane Potential

The equations governing the voltage gating variables are defined Equations 21-23.

## TC

Equation 25 - TC Membrane Potential

Equation 26

Equation 27

The TC relay cell membrane potential is defined in Equation 25. The leakage and sodium currents are defined identically to the GPe/GPi/STN neurons. The gating variables for the TC neuron have been consolidated and therefore the potassium and slow calcium current have a different representation as seen in Equations 21-23. The *n* gating variable has been consolidated and only the *r* and *h* gating variables will be used. The equations for the gating variables are listed in Equations 28 and 29.

Equation 28

Equation 29