THE UNIVERSITY OF TOKYO

Compensating Temperature Dependent Characteristics of a Subthreshold-MOSFET Analog Silicon Neuron

サブスレショルド MOSFET アナログシリコンニューロンの温度依存特性補償

Master's Dissertation

Ethan Joseph Green
Department of Electrical Engineering and Information Systems
Student ID: 37-146884
Advising Professor: Takashi Kohno

Submitted on February 4, 2016

Table of Contents

Chapter 1: Introduction

1.1. Overview

Silicon neurons are biologically inspired VLSI (very-large-scale integrated) circuits that replicate the electrophysiological behavior of nerve tissue. This research looks at an analog silicon neuron based on qualitative neuronal modeling. In the future, such circuits may be used as elements in massive networks of artificial neurons. However, many technical challenges must be addressed before such large networks are practical. This research looks at the critical issue of temperature sensitivity of subthreshold MOSFET circuits.

1.2. Motivation

Neurons, the fundamental cells of the nervous system, are incredible calculating machines. The human brain for example contains about 100 billion neurons, interconnected with 1 quadrillion (10^{15}) synapses. These cells regulate body functions, process sensory data, store memories, execute judgements, perform learning, and do a host of other complex tasks essential to animal life.

Neurons and the neural networks they form operate in ways fundamentally different from the transistor logic used in digital computers. Neurons send and receive signals in a noisy environment, have a high error tolerance, calculate in massively parallel networks, and require little power. The entire human brain for example consumes only 20 watts.

Neurons are excellent at performing basic tasks like memory recall, pattern recognition, learning, and selective attention, not to mention more complicated tasks like riding a bicycle through unfamiliar streets or memorizing a difficult piece of music. The motivation to develop neuromorphic technology stems from the desire to create computers that replicate all these wonderful features.

1.3. Types of Silicon Neurons

Many types of silicon neurons are being developed with divergent approaches based on intended applications and desired degree of detail. Silicon neurons can be divided into two broad categories, digital silicon neurons and analog silicon neurons.

1.4. Digital Silicon Neurons

Digital silicon neurons operate in discrete time with simplified neuronal models. These circuits suffer from quantization noise and often require high power, however, due to their relative simplicity, have already been arranged into massive networks like IBM's True North which contains 1 million programmable digital neurons with 256 million synapses [1].

1.5. Analog silicon neurons

Analog silicon neurons operate in continuous time similar to biological neurons which are also analog in nature [2]. These circuits can be divided into three categories:

1.5.1. Conductance-based neurons

Analog silicon neurons based on conductance models are designed to accurately replicate biological neuron behavior down to the cellular level. These circuits are biologically accurate, but often complex, difficult to design and tune, and require high power [2].

1.5.2. Leaky Integrate and Fire

Silicon neurons based on Leaky Integrate and Fire models are mathematically simple, relying on a reset of state variables for their spiking mechanism [2]. These silicon neurons have less complicated circuitry that is easier to implement, but suffer from uniform spike shape. Biological research suggests that the shapes of action potentials may be important in neuron communication, and these circuits do not have the facility for such variation [7].

1.5.3. Qualitative neurons

Qualitative neurons are based on approximations of conductance models and can replicate similar dynamics with less complexity. These models can create a variety of spike shapes and dynamics with circuitry that is easier to implement and requires less power. Qualitative silicon neuron circuits aren't as biologically accurate as conductance-based circuits, but are likely more feasible than Leaky Integrate and Fire models.

The circuit used in this research is a qualitative silicon neuron introduced by Kohno and Aihara in 2014 [3]. Figure 1.1 shows an enlarged photograph of the VLSI architecture, and Figure 1.2 shows the silicon neuron in its bed in the laboratory.

1.6. Applications

Silicon neuron technology has many potential applications in the future. Artificial neurons may be the main elements in future CPU's for autonomous machines or computers with artificial intelligence. Silicon neurons can also be used in medical applications, such as designing artificial brains to run medical experiments, or developing prostheses and brain-machine-interfaces that connect living tissue to computers or mechanical components. Research into hybrid networks connecting silicon neurons to living neurons is already being carried out [4][5]. Beyond biology, medicine, and robotics, the low-power characteristics of silicon neurons may be exploited for low power computing of all kinds of complex problems.

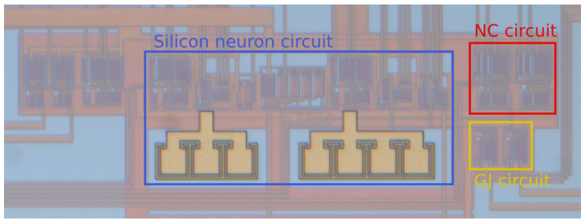


Figure 1.1: Photograph of the silicon neuron circuit taken with a microscope

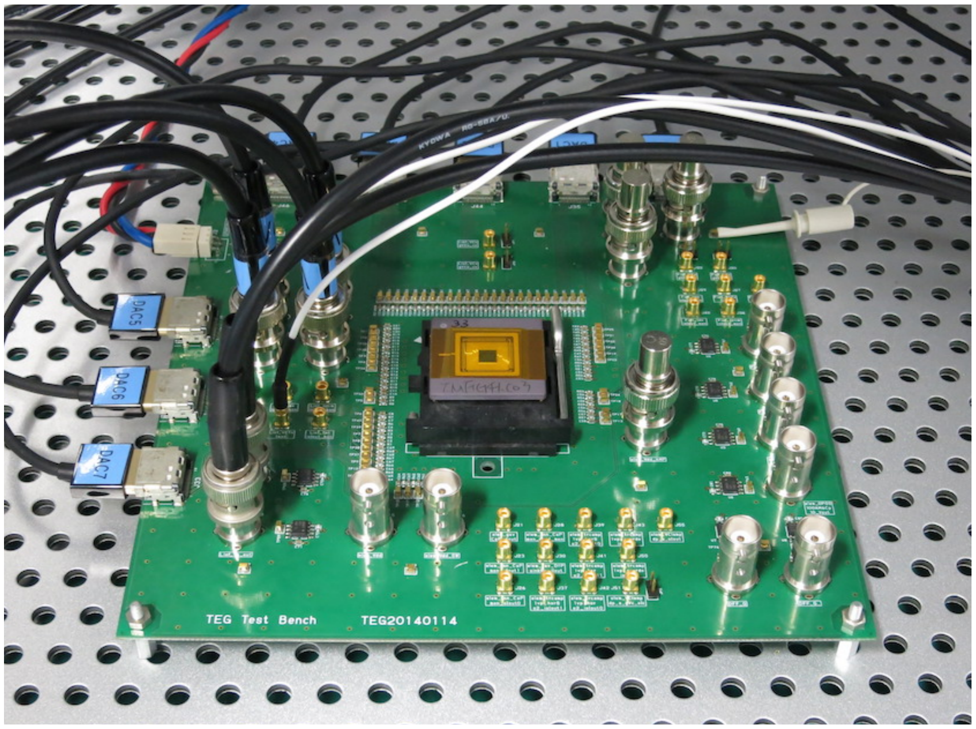


Figure 1.2: Full circuit board with the silicon neuron resting in the center

1.7. Structure of this work

First, Chapter 2 explains some of the basic features of a qualitative neuron model. Chapter 3 lays out the issue of temperature sensitivity for subthreshold MOSFET circuits. Chapter 4 details the features of the silicon neuron used in this research. Chapters 5 through 8 look at strategies for tuning the parameters of the circuit to compensate for temperature-induced changes in behavior. Chapters 5 and 6 explain trial and error based strategies and Chapters 7 and 8 augment these strategies with automation. Chapter 8 proposes the use of a Differential Evolution algorithm to tune the circuit. Finally, Chapter 9 concludes this work with a discussion of the results.

Chapter 2: Qualitative Modeling

2.1. Conductance models

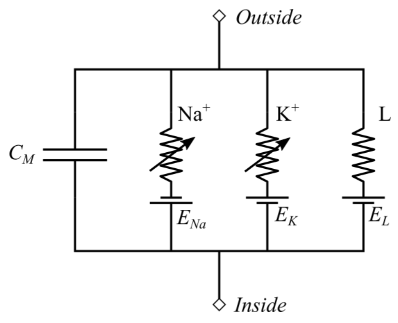


Figure 2.1: Equivalent Circuit for the Hodgkin-Huxley Model

The Hodgkin-Huxley model of the nerve cell membrane is a Nobel Prize winning model originally published in 1952 and serves as the basis of much research regarding neural modeling and neuromorphic engineering.

Hodgkin and Huxley measured the flow of ions across the membrane of a squid nerve axon to make a comprehensive set of differential equations that accounts for the individual species of ions [6]. Figure 2.1 shows the equivalent circuit of this model with membrane capacitance C_M, a leak channel L of constant resistance, and ion channels for Na$^+$ and K$^+$ ions with variable resistances. E_{Na}, E_K, and E_L represent the reversal potentials of these ions.

The Hodgkin-Huxley model accounts for a wide variety of experimentally observed nerve cell behaviors. Two significant classes of neurons that the model can replicate are Class I and Class II neurons. Class I neurons can oscillate at an arbitrarily low frequency and Class II neurons begin oscillation at a set minimum frequency.

The Hodgkin-Huxley model is biologically accurate and can be extended to further levels of detail by adding more ion channels. However, the model is complicated, highly nonlinear, and has many variables and parameters, making it difficult to implement on a circuit.

2.2. Qualitative modeling

Qualitative modeling seeks to recreate similar dynamical behavior of the Hodgkin-Huxley model by using judicious approximation to create a system with simpler equations and fewer variables [3][7][8][9][10]. With proper dynamical structure, such a system can theoretically reproduce Class I and Class II periodic firing, in addition to other neuron behaviors. Qualitative models should ideally be less demanding to analyze and also easier to implement on a circuit.

2.3. The nullclines and phase plane of a nonlinear system

A simplified mathematical model of the circuit based on qualitative neuron modeling to be described in Chapter 4 can be written as:

$$C_v \frac{dv}{dt} = f_1(v,n) + I_{stim} \tag{2.1}$$

$$\frac{dn}{dt} = f_2(v,n) \tag{2.2}$$

This system replicates a wide variety of spiking behavior with only 2 variables. Variable *v* represents the membrane potential, variable *n* represents abstracted ionic activity, I_{stim} is an applied stimulus current, and C_v is the membrane capacitance. A full description of the equations in this system can be found in Chapter 4.

In the field of nonlinear dynamics, an important feature of a nonlinear system is its phase plane, a graph of one variable versus another [11][12]. The phase plane also contains the *nullclines*: curves on which the derivatives of each variable equal zero. In the system above, functions f_1 and f_2 indicate the shape of the *v* and *n*-nullclines. For $I_{stim} = 0$, the equations of the nullclines are:

$$\frac{dv}{dt} = f_1(v,n) = 0 \tag{2.3}$$
$$\frac{dn}{dt} = f_2(v,n) = 0 \tag{2.4}$$

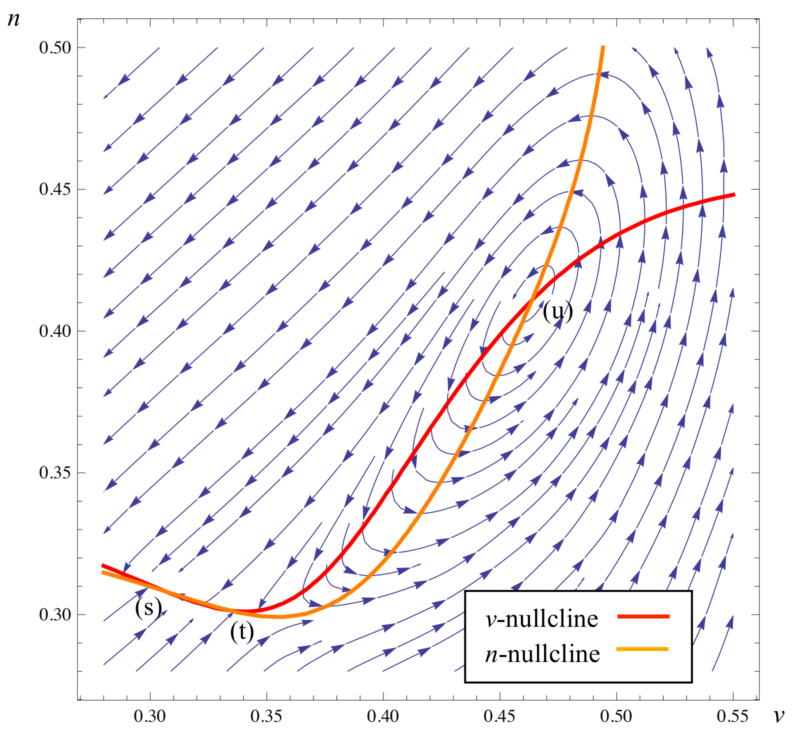
Figure 2.2 (a) shows a plot of the phase plane with the *v* and *n*-nullclines highlighted in red and orange. The blue streamlines indicate the possible trajectories an action potential could take through the phase plane.

To replicate Class I neuron behavior, the nullclines must be made to cross three times [12]. To accomplish this, f_1 is assigned a pronounced cubic “s” shape which intersects with the *n*-nullcline in three places when $I_{stim} = 0$. The leftmost intersection of the nullclines (Figure 2.2 (b)) is a stable attracting node which corresponds to the resting state of the system. All trajectories in its vicinity are attracted to this point, and trajectories farther away eventually will be pulled back to it.

The middle intersection slightly to the right of the stable point is a saddle node which attracts trajectories vertically and repels them horizontally. This point corresponds to the threshold current to generate an action potential, referred to as I_{th}. A trajectory from the left must have a sufficiently strong stimulus to overcome the repelling strength of the saddle point and attraction of the stable node before beginning a cycle around the phase plane. Figure 2.2 (b) zooms in on these two points. The uppermost intersection (Figure 2.2 (a)) is an unstable node which repels all trajectories. Action potentials will cycle around this point.

The stability of each of the nullcline intersections can be characterized algebraically with techniques from nonlinear dynamics described briefly in Endnote 1.

(a)



(b)

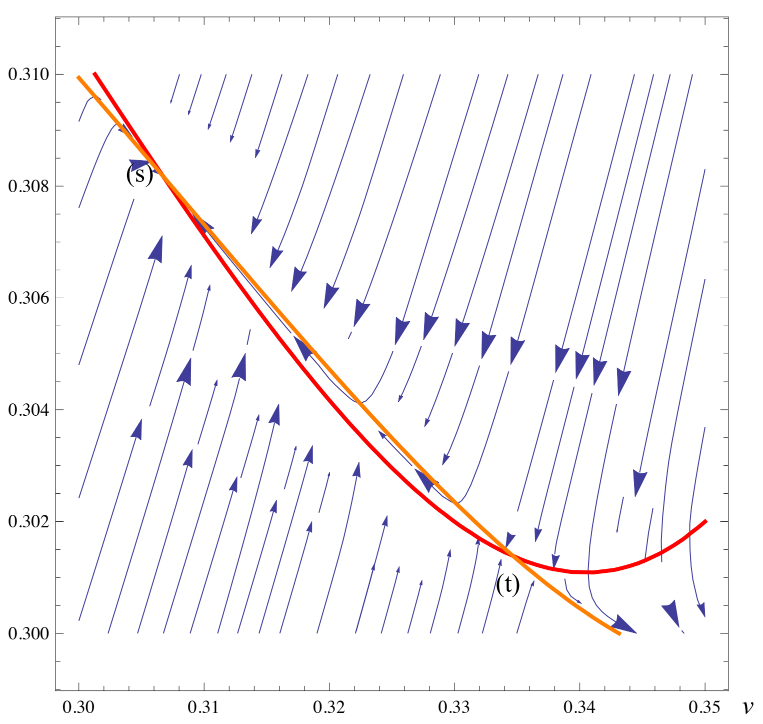


Figure 2.2: (a) Phase plane of Class I behavior and (b) zoom-in showing the stable node (s), saddle node (t), and unstable node (u)

2.4. Single action potential mechanism

The system begins in its rest state at the stable node. Applying pulse stimulus I_{stim} to the system will raise the membrane potential momentarily, meaning the state will shift rightward in the phase plane. If the stimulus is weak, the trajectory will be repelled by the saddle node and attracted back to the stable node, quickly returning to the rest state [12]. A stimulus of sufficient strength will push the trajectory beyond the saddle node to a larger trajectory that travels around the unstable node and eventually back to the stable node, forming a single action potential. With no other stimulus, the system will remain at rest on the stable node. Figure 2.3 (a) shows this path in the phase plane, and Figure 2.3 (b) shows the time-series plot.

It is interesting to note that the membrane potential reaches its maximum value when the trajectory crosses the v-nullcline. The nullcline represents when the derivative of a variable equals zero, so the derivative always switches sign ($+ \leftrightarrow -$) when a trajectory crosses the curve. In the case of an action potential, the derivative of v is positive as the membrane potential rises, is momentarily zero at the maximum of the spike, and then becomes negative as the membrane potential falls.

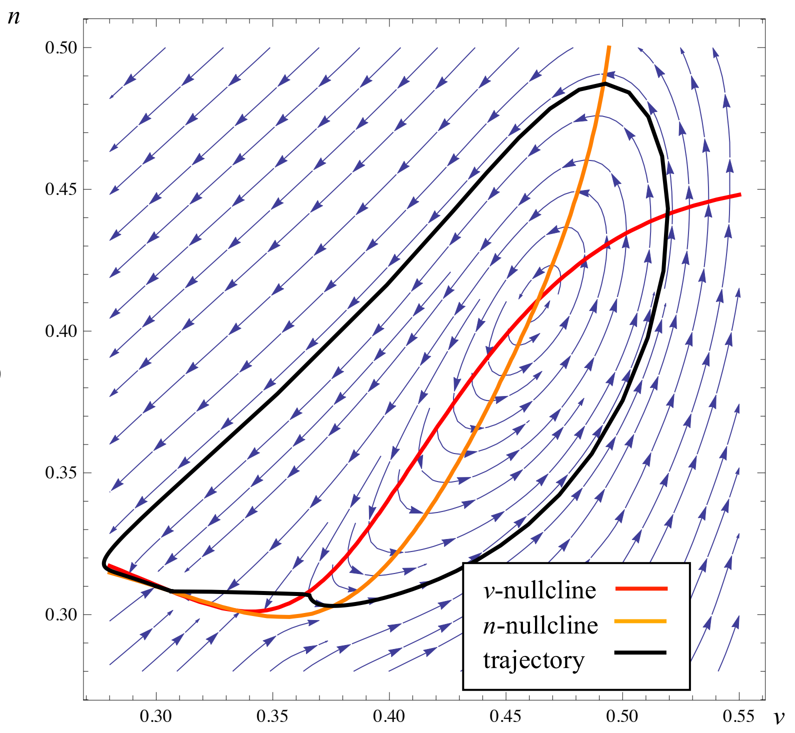
2.5. Class I Oscillation Mechanism

I_{stim} vertically translates the values of f_1 in Equation 2.1, so a sustained stimulus applied to the system will shift the v-nullcline upwards. When this happens, the stable node and saddle node will approach, merge, and suddenly annihilate each other, a process known as *bifurcation* [11]. The already circular path around the unstable node becomes a closed orbit, known as a *limit cycle*. With no stable rest state, the trajectory will tend towards this circular path and begin to oscillate. The black trajectory in Figure 2.4 (a) merges with the limit cycle. The loss of the stable and saddle nodes is clear in the figure by the position of the nullclines. Figure 2.4 (b) shows the time-series plot of periodic spiking.

When the sustained stimulus current is removed, the v-nullcline will return to its original position, reintroducing the stable node and saddle node which existed before. The stable node will attract the trajectory and the system will then return to its rest state.

A significant feature of Class I behavior is that the oscillation frequency can be arbitrarily low [12]. If the applied stimulus is just strong enough to move the v-nullcline slightly beyond the n-nullcline, a narrow channel will form. Trajectories traveling through the channel will slow depending on how close the nullclines are. This can lead to near-zero firing frequency. Conversely, a strong stimulus current will increase the size of the channel, leading to higher firing frequency.

(a)



(b)

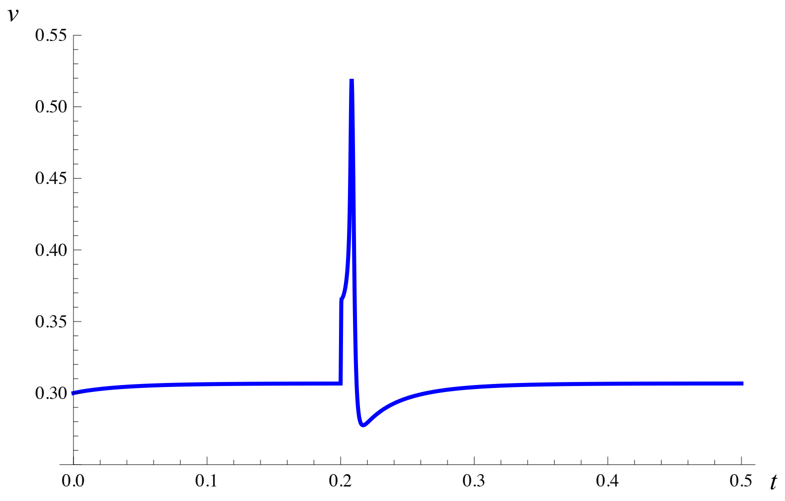
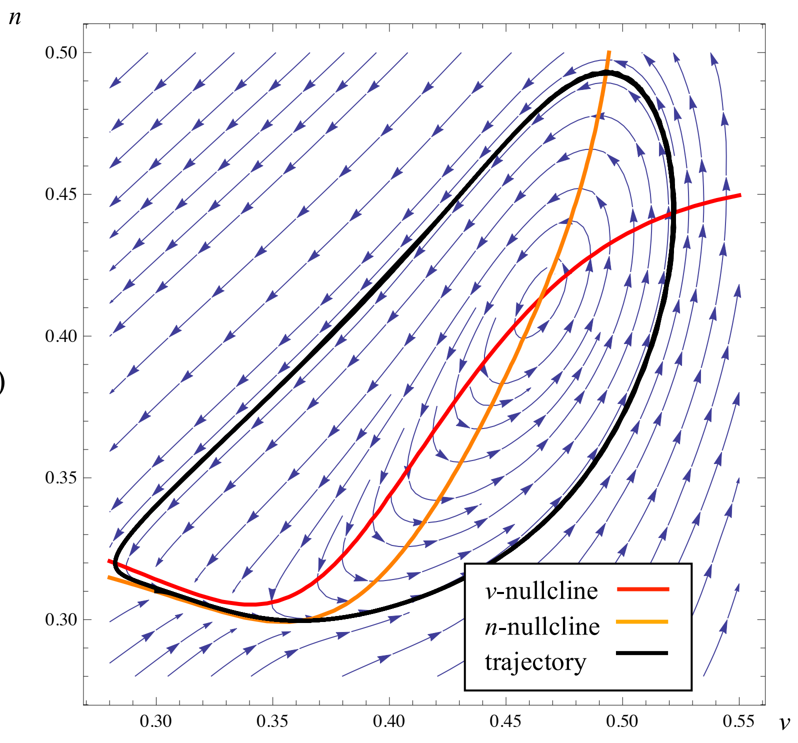


Figure 2.3: (a) Phase plane with trajectory and (b) time-series of a single action potential

(a)



(b)

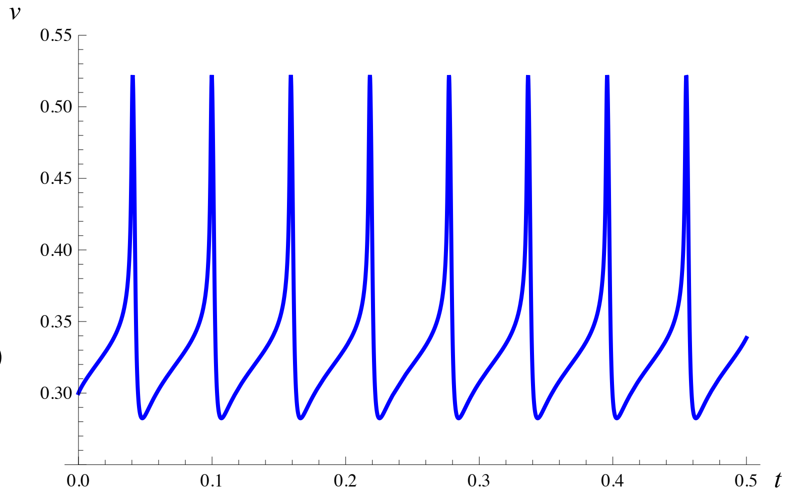


Figure 2.4: (a) Phase plane with trajectory and (b) time-series of periodic spiking behavior

2.6. Endnotes

1) Nodes in the phase plane are characterized by forming a Jacobian Matrix, the matrix of partial derivatives of each combination of variables in the system, and solving the associated linear system[11].

$$\begin{bmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial v} & \frac{\partial f_1}{\partial n} \\ \frac{\partial f_2}{\partial v} & \frac{\partial f_2}{\partial n} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.5}$$

$$\begin{bmatrix} \frac{\partial f_1}{\partial v} & \frac{\partial f_1}{\partial n} \\ \frac{\partial f_2}{\partial v} & \frac{\partial f_2}{\partial n} \end{bmatrix} \triangleq \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{2.6}$$

$$\lambda^2 - (a + d)\lambda + \begin{vmatrix} a & b \\ c & d \end{vmatrix} = 0 \tag{2.7}$$

The numerical values of the partial derivatives at each node are input into the matrix, and the system is solved. The behavior of the linear system at each node corresponds to the local behavior at the points in the full nonlinear system. The eigenvalues determined by Equation 2.7 describe the behavior at the node. Two negative eigenvalues denote a stable node, two positive eigenvalues denote an unstable node, and a negative and positive eigenvalue denote a saddle node.

2) The Class I bifurcation discussed in this chapter in which two nodes annihilate each other and oscillatory behavior emerges is an example of a *Hopf Bifurcation* [11][12].

Chapter 3: Temperature Dependency of Subthreshold MOSFETs

3.1. Overview

The majority of MOSFET circuits are operated above-threshold, a regime more suitable for the high speed operation of digital circuits. Until recently, subthreshold operation of MOSFETs was considered novel, but with the advent of many low power applications for electronics, this regime may take on increased importance. For the silicon neuron in this research, operating transistors in the subthreshold region allows for low power consumption and desirable exponential characteristics that yield the sigmoidal current-voltage curves described in Chapter 4. However, a significant drawback of subthreshold-operated transistors is pronounced temperature sensitivity.

Figure 3.1 (a) shows simulations of the silicon neuron's frequency response to a sustained 4 pA stimulus at different temperatures. 0.5°C changes in temperature dramatically alter the firing frequency, with periodic firing ceasing at 26.5°C. The influence of temperature on a single action potential due to a 200 pA pulse stimulus (500 μs duration) can be seen in Figure 3.2 (b). Higher temperatures reduce the size of the action potential. A change in the resting potential is also apparent.

As these two figures show, slight changes in temperature can dramatically affect the operation of the circuit. Addressing this problem is a key step in developing this qualitative silicon neuron into a functional electronic component.

3.2. Source of temperature sensitivity

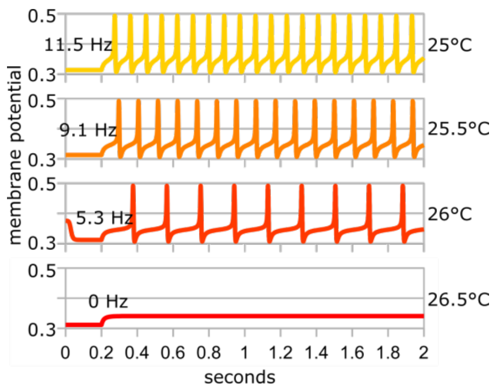
The equation for the saturation drain current in the subthreshold regime is:

$$I_d = I_0 \exp\left(\frac{\kappa \mathrm{V_g} - \mathrm{V_s}}{U_T}\right) \tag{3.1}$$

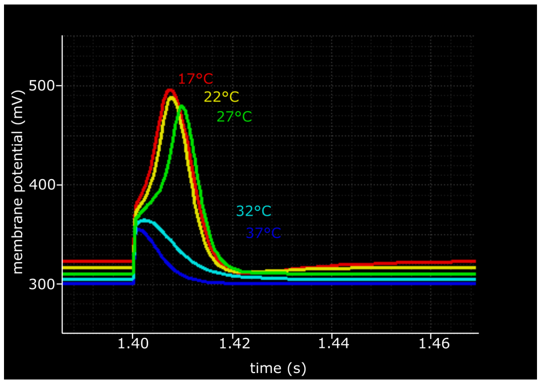
The equation for linear drain current is:

$$I_d = I_0 \exp\left(\frac{\kappa \mathrm{V_g} - \mathrm{V_s}}{U_T}\right)\left(1 - \exp\left(\frac{-\mathrm{V_{ds}}}{U_T}\right)\right) \tag{3.2}$$

$\mathrm{V_s}$ is the source voltage, $\mathrm{V_g}$ is the gate voltage, and $\mathrm{V_{ds}}$ is the drain-to-source voltage of the MOSFET. I_0 is the off-current that passes through the transistor when $\mathrm{V_g} = \mathrm{V_s}$, κ is the capacitive coupling ratio, and U_T is the thermal voltage [13].



(a)



(b)

Figure 3.1: Illustration of temperature sensitivity with (a) frequency response of the silicon neuron to a 4 pA sustained stimulus starting at 0.2 seconds and (b) response to a 200 pA 500 μs pulse stimulus

Both U_T and I_0 are temperature dependent factors. U_T is directly proportional to the absolute temperature in Kelvins (T):

$$U_T = \frac{kT}{q} \tag{3.3}$$

q is the charge of an electron and k is the Boltzmann Constant. I_0 has a more complex relationship with temperature depending on temperature dependent factors U_T , the threshold voltage of the transistor V_{T0}, and the electron mobility constant μ.

$$I_0 = 2\mu \frac{C_{ox}}{\kappa} \frac{W}{L} U_T^2 \exp\left(\frac{-\kappa V_{T0}}{U_T}\right) \tag{3.4}$$

W and L are the width and length of the transistor and C_{ox} is the gate oxide capacitance. I_0 has a pronounced influence on the transistor current with higher temperatures corresponding to stronger currents. Figure 3.2 shows a curve of I_0 versus temperature for a typical PMOS transistor (W = 500 nm, L = 16 µm) used in the silicon neuron circuit.

3.3. Past Approaches

Environmental control has been the main means of dealing with the problem of temperature sensitivity of the silicon neuron. In the laboratory, the circuit is operated in an air-conditioned chamber displayed in Figure 3.3 (a). The air-conditioner circulates air of constant temperature over the circuit, thus drawing off excess heat and maintaining a constant environment. This solution suits the purposes of the laboratory, but is bulky, immobile, and expensive.

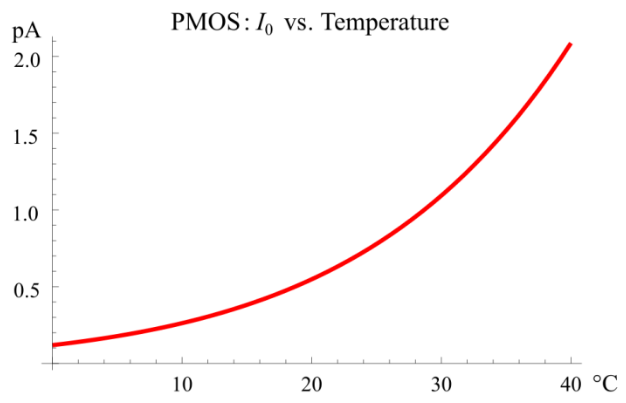
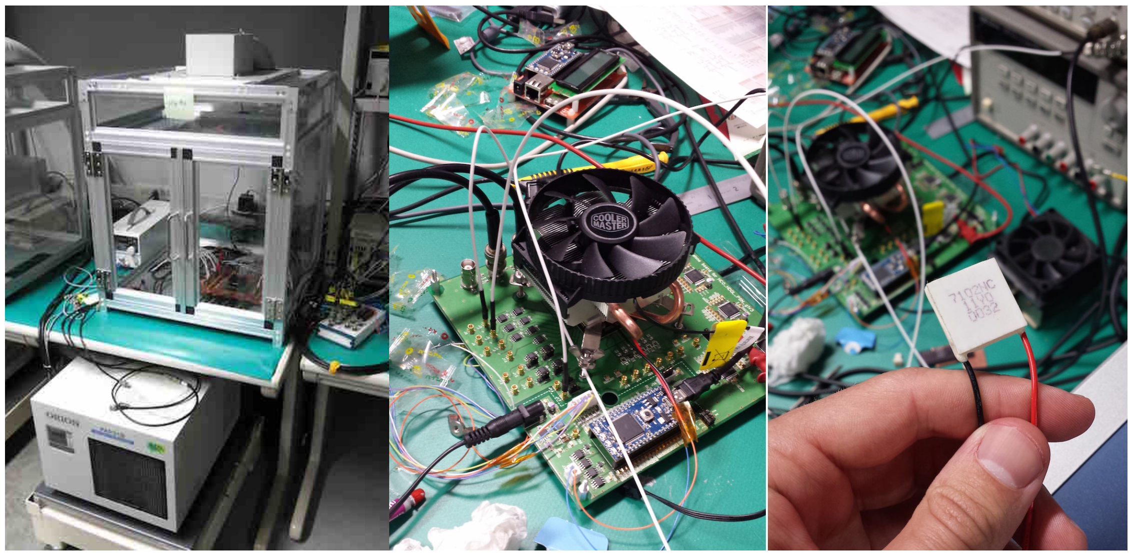


Figure 3.2: Plot of I_0 versus temperature for a typical PMOS transistor used in the silicon neuron



(a) (b) (c)

Figure 3.3: (a) Air-conditioned chamber, (b) apparatus with cooling fan, and (c) Peltier chip

A low cost mobile apparatus was constructed by Miyake [14] (Figure 3.3 (b)) which uses a cooling fan to draw off heat and a Peltier chip (Figure 3.3 (c)) to act as a heat sink or heat source. With feedback from a temperature sensor, this system can maintain a constant temperature. The apparatus is mobile and has been used to run experiments with a similar subthreshold silicon neuron in other laboratories.

Both of these solutions suffer from the flaw that they consume high power, thus defeating the low power attributes of a subthreshold-operated silicon neuron.

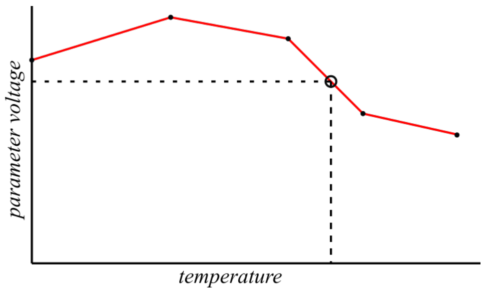


Figure 3.4: Interpolation concept

3.4. Proposed Method: Parameter Control Approach

This research seeks to develop a new approach to dealing with the temperature sensitivity of the circuit by adjusting the silicon neuron's circuit parameters in response to changes in temperature. An understanding of how to control circuit parameters at different temperatures can be used to engineer a feedback mechanism to be incorporated into a future generation of the chip.

The method explored is based on *interpolation*. The circuit is simulated over a wide range of temperatures in regular intervals. Circuit parameters at each interval are sought that yield similar circuit behavior. These parameter sets are then used to *interpolate functions of parameters versus temperature* which can be used to output proper circuit parameter sets for any intermediary temperature. Figure 3.4 illustrates this concept.

The temperatures measured were 22, 27, 32, and 37°C, hereafter referred to as the *pillar temperatures.* A variety of tuning strategies to find *pillar parameter sets* are discussed in Chapters 5 through 8.

The proceeding methods were accomplished with the Spectre simulator in the Cadence environment. An effective strategy developed in a simulated environment can theoretically be applied to an actual VLSI circuit. Furthermore, a tuning strategy incorporating automated optimization like the processes proposed in Chapters 7 and 8 could potentially overcome the problems of transistor mismatch, noise, and variation in a real-world environment.

Chapter 4: Circuit Description

4.1. System Equations

The circuit used in this research is an ultra-low power VLSI silicon neuron designed by Kohno and Aihara in 2014 [3] using the qualitative modeling principles discussed in Chapter 2. The schematic was created with Cadence and simulations were run with Spectre. The circuit was fabricated by Taiwan Semiconductor with a 0.25 μm CMOS process. V_{dd}, the highest voltage used by the components of the silicon neuron, is set to 1.0 V. All the transistors in the circuit are operated in the subthreshold regime, which in combination with the low V_{dd} value contribute to low power consumption around 3 nW.

Figure 4.1 shows a block diagram of the silicon neuron. The circuit is divided into a *v*-block and an *n*-block. Variable *v* represents the membrane potential of the silicon neuron, and variable *n* represents abstracted ionic activity. Each block contains a group of transconductance circuits whose output currents charge and discharge a capacitor. As a convention, the values of *v* and *n* are not defined in reference to V_{ss} (0 V, also referred to as ground), but instead by the voltages over their respective capacitors subtracted from V_{dd}.

The system equations of the silicon neuron are as follows:

$$C_v \frac{dv}{dt} = f_v(v) - g_v(v) + I_{av} - r(n) + I_{stim} \tag{4.1}$$

$$C_n \frac{dn}{dt} = f_n(v) - g_n(v) + I_{an} - r(n) \tag{4.2}$$

C_v and C_n are the capacitances of the capacitors, $f_x(v)$, $g_x(v)$, and $r(n)$ ($x = v, n$) are functions that represent sigmoidal current-voltage characteristics of the transconductance circuits, I_{av} and I_{an} are constant current sources, and I_{stim} is an externally applied stimulus current.

4.2. Description of the Transconductance Circuits

4.2.1. $f_x(v)$

Figure 4.2 (a) shows the $f_x(v)$ circuit which consists of a differential pair amplifier coupled with a cascoded current mirror. PMOS transistors M1, M2, and M4 compose the fundamental units of the differential pair: M1 supplies current that is shared by M2 and M4 [13]. If the gate voltages on these two transistors, V_{in} and V_{dlt}, are equal, the tail current will be divided evenly. If V_{in} is farther from V_{dd} than V_{dlt}, more current will flow on the M4 side of the differential pair. This current is in turn copied by the current mirror composed of NMOS transistors M6 and M7. If all transistors are operated in the saturation region, V_{casc} will have less influence on the current in the current mirror and can thus be determined approximately.

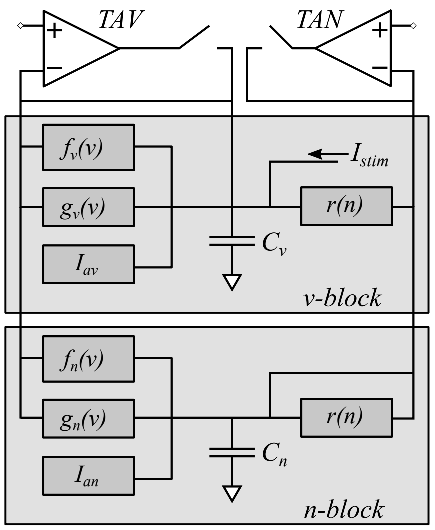


Figure 4.1: Block diagram of the silicon neuron

The current-voltage equation for *$f_x(v)$* can be algebraically derived from the equation for subthreshold saturation current (Equation 3.1) and is as follows:

$$f_x(v) = \frac{M_x}{1+\exp\left(-\frac{\kappa}{U_T}(v-\delta_x)\right)} \tag{4.3}$$

The curve of this equation is sigmoidal. M_x corresponds to the current in M1 which is controlled by V_b and shared by the differential pair. The variable v corresponds to V_{in}. δ_x corresponds to V_{dlt} and can be used as an offset to shift the characteristic curve along the v-axis. κ is the capacitive coupling ratio of the PMOS transistors used in the circuit and U_T is the thermal voltage.

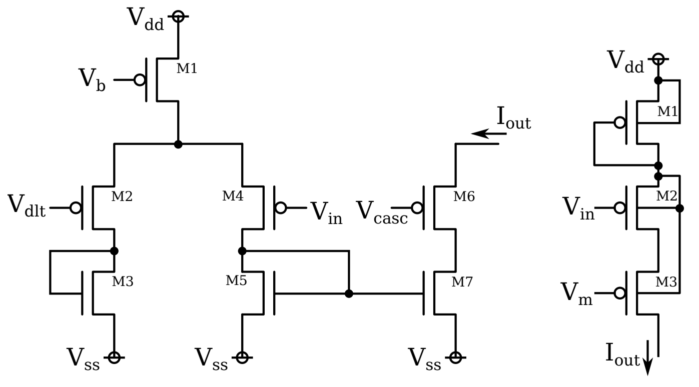
M_x in turn can be defined by the equation for subthreshold current:

$$M_x = I_0 \exp\left(\frac{\kappa|V_b - V_{dd}|}{U_T}\right) \tag{4.4}$$

Endnote 1 of this chapter discusses further low-power functionality of *$f_x(v)$*.

4.2.2. *$g_x(v)$*

Figure 4.2 (b) shows the *$g_x(v)$* circuit which is based on a PMOS cascode circuit with source degeneration. M3 is the cascode stage transistor and M1 provides source degeneration to M2. Source degeneration essentially flattens out the current-voltage characteristics of the transistor by placing a buffer stage which raises the source voltage [13].



(a) $f_x(v)$ (b) $g_x(v)$

Figure 4.2: Circuit diagrams for (a) $f_x(v)$ and (b) $g_x(v)$

A unique aspect of this circuit is that the bulk voltage of the transistors in the cascode are detached from V_{dd} and instead tied to the drain of M1. Using the equation for saturation region current (Equation 3.1) for M1 and M3, linear region current (Equation 3.2) for M2, and accounting for the detached bulk voltage of the cascoded transistors, the equation for this circuit can algebraically be derived to be:

$$g_x(v) = I_0\sqrt{\frac{\exp\left(\frac{\kappa}{U_T}\theta_x\right)}{1+\exp\left(-\frac{\kappa}{U_T}(v-\theta_x)\right)}} \tag{4.5}$$

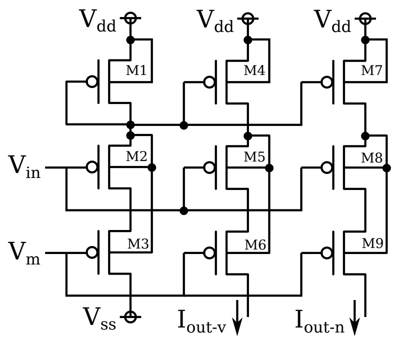
The square root makes the sigmoidal characteristics of the current-voltage curve more gradual. θ_x corresponds to V_m, the voltage bias on the cascode stage, and I_0 is the transistor off-current.

A major attribute of this circuit is its simplicity. With only three transistors, the circuit requires minimal current and thus contributes to the overall low-power characteristics of the silicon neuron.

For the purposes of this research, $g_v(v)$ and $g_n(v)$ are considered equivalent circuits. Endnote 2 of this chapter discusses further functionality of $g_n(v)$.

4.2.3. *r(n)*

Figure 4.3 shows the *r(n)* circuit which operates according to the same principles of $g_x(v)$. However, this circuit is comprised of two parts that output equivalent currents to the *v*-block and *n*-block of the silicon neuron. The two *r(n)* boxes in the silicon neuron diagram (Figure 4.1) represent these two equivalent currents governed by the following equation:



r(n)

Figure 4.3: Circuit diagram for *r(n)* showing equivalent current outputs for both circuit blocks, $I_{out\text{-}v}$ and $I_{out\text{-}n}$

$$r(n) = I_0 \sqrt{\frac{\exp\left(\frac{\kappa}{U_T}\theta_r\right)}{1+\exp\left(-\frac{\kappa}{U_T}(n-\theta_r)\right)}} \tag{4.6}$$

θ_r corresponds to the bias voltage V_m. *r(n)* functions as an intersection between the *v* and *n*-blocks since its output current is the same in both.

4.2.4. I_{ax}

I_{an} and I_{av} are transconductance amplifiers that provide constant currents to each block. Their output currents are scaled by V_{in}. Since supplying constant current is their only function, their mathematical properties are not significant in the operation of the silicon neuron and their bias voltages can be determined approximately.

4.3. Referencing Variables to V_{dd}

As mentioned above, the variables *v* and *n* are coded by subtracting the voltage over the capacitors in the *v* and *n*-blocks from V_{dd}. This convention is adopted because V_{in} for each of the transconductance circuits is tied to a PMOS transistor. PMOS transistors pass more current when their gate voltages are farther from V_{dd}, so defining *v* as $V_{dd} - V_{in}$ is convenient; a higher value for *v* corresponds to more current.

There are a few significant reasons why the circuit components in the silicon neuron are mainly comprised of PMOS transistors:

1) PMOS transistors draw less current than NMOS transistors due to a low I_0 value. This contributes to the silicon neuron's low-power characteristics.

2) In the CMOS process used, only PMOS transistors are fabricated with an independent bulk voltage which is utilized in the cascode circuits in *$g_x(v)$* and *r(n)*.
3) The independent bulk voltage also makes PMOS transistors more resistant to noise.

In this work, any discussion of membrane potential *v* and variable *n* imply the above convention. V_{in} and the phrase "input voltage" as used in Chapter 5 will refer to voltages referenced to V_{ss} (0 V).

4.4. Nullcline Mode

The two transconductance amplifiers *TAV* and *TAN* at the top of the circuit diagram are used to draw the nullclines of the system which essentially represent the steady-state characteristics of the circuit. As mentioned in Chapter 2, the term "nullcline" comes from the discipline of nonlinear dynamics and refers to a curve on which the derivative of a variable is equal to zero. The nullcline curves of this system can be determined by setting the system equations equal to zero. From Equations 4.1 and 4.2:

$$\frac{dv}{dt} = 0 \tag{4.7}$$

$$\frac{dn}{dt} = 0 \tag{4.8}$$

Thus,

$$f_v(v) - g_v(v) + I_{av} - r(n) = 0 \tag{4.9}$$

and

$$f_n(v) - g_n(v) + I_{an} - r(n) = 0 \tag{4.10}$$

At the points on the nullcline of each circuit block, the output currents of the transconductance circuits balance each other, hence no current passes over the capacitor and the voltage remains constant. The nullclines of the system are significant because they can be used to construct a phase plane which indicates the behavior of the circuit, namely if it is operating in Class I or Class II mode according to Hodgkin and Huxley's classification (Chapter 2).

The nullcline drawing procedure is as follows:

1) V_m of *r(n)* is set to V_{dd}, thus switching this circuit off and bypassing it.
2) The non-inverting input of *TAN* (+), the transconductance amplifier for the *n*-block, is set to a constant voltage, ideally the median voltage of this block during normal transient

operation. The inverting input (–) of *TAN* is connected to the *n*-block and this negative feedback arrangement maintains a constant voltage.

3) The non-inverting input of *TAV* (+), is stepped from 0 V to 1.0 V (or any other desired range). The inverting input (–) is connected to the *v*-block so the negative feedback will force the system to the voltage of each step.
4) The steady-state output currents of *TAV* and *TAN* at each voltage step are recorded and used to represent the *v* and *n*-nullclines.

Since the *r(n)* circuit is bypassed in nullcline mode, the output currents of *TAV* and *TAN* represent the equivalent current necessary from *r(n)* to achieve a steady-state. In this sense, the nullclines are monotonically related to the variable *n* through $r^{-1}(n)$ (Endnote 4). The method of using negative feedback to hold each circuit block at a constant voltage is inspired the technique of *voltage clamping* used by Hodgkin and Huxley when they took measurements of the ionic currents of across a squid axon membrane.

4.5. Endnotes

1) The current mirror attached to the differential pair in $f_x(v)$ is tied to a series of current mirrors which can be used to further boost the output current (Figure 4.4). These current mirrors are switched on and off by bias voltages V_{en0} and V_{en1}. Transistors M8 and M9 in the middle current mirror can be used to double the output current, and transistors M10 and M11, each comprised of two parallel transistors (specified as m=2 in the diagram) can be used to triple the output current. Turning on both supplementary current mirrors will quadruple the output current.

   These output current boosters can be used to reduce the current required by the differential pair, thus lowering the overall power consumption of this component.

2) $g_n(v)$ includes a series of switches (Figure 4.5) that can be used to move its current-voltage curve rightward along the V_{in} axis. The equation for this circuit then becomes:

$$g_n(v) = I_0 \sqrt{\frac{R_0 \exp\left(\frac{\kappa}{U_T}\theta_n\right)}{1 + R_1 \exp\left(-\frac{\kappa}{U_T}(v - \theta_n)\right)}} \tag{4.11}$$

   $R_x = 2$ ($x = 0, 1$) when its corresponding switch is turned on by setting V_{swxp} to V_{dd_sw} and V_{swxn} to V_{ss}. $R_x = 1$ when V_{swxp} is set to V_{ss} and V_{swxn} is set to V_{dd_sw}.

3) Both *r(n)* and I_{ax} contain an additional stage which outputs another equivalent current for off-chip monitoring purposes.

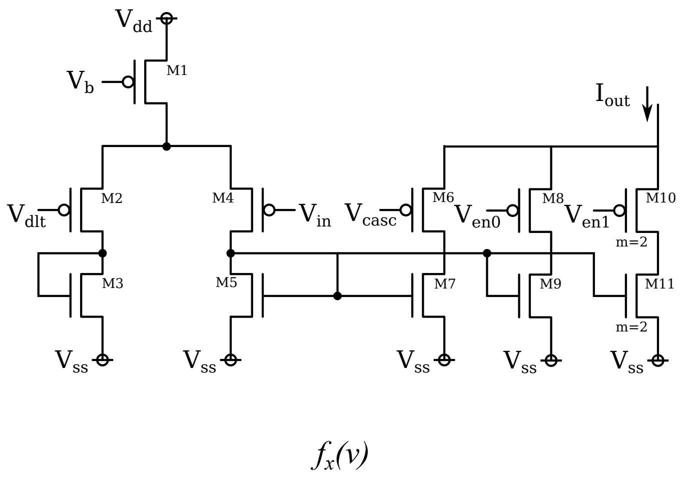


Figure 4.4: Complete *$f_x(v)$* circuit with current multipliers

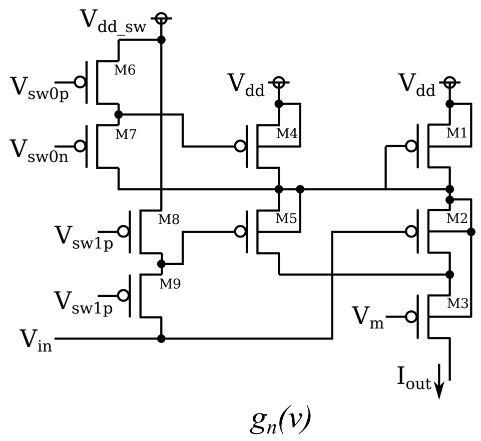


Figure 4.5: Complete *$g_n(v)$* circuit with switches that shift the current-voltage characteristics

4) The vertical axis of the phase plane of the mathematical model is *n* (Chapter 2). The vertical axis of the phase plane drawn by the circuit in nullcline mode is *r(n)*. Because *n* and *r(n)* are monotonically related, the relative positions of the intersections of the nullclines can still be determined by the silicon neuron's nullcline mode.

5) Cadence library references:

full silicon neuron	NAHAR_g3v1aT	Rev. 1.0, 2/1/2014
$f_x(v)$	fxv_DfP_sinkQx	Rev. 1.0, 29/12/2013
$g_v(v)$	gxv_CsP_s	Rev. 1.0, 1/1/2014
$g_n(v)$	gxv_CsP	Rev. 1.0, 27/12/2013
$r(n)$	rnn_CsP_mon	Rev. 1.0, 2/1/2014
I_{ax}	trcamp_Ivp_horS_o2	Rev. 2.0, 2/1/2014
TAX	trcamp_Ivp_hor_o2	Rev. 1.0, 2/1/2014

Chapter 5: Full Parameter Approach

5.1. Overview

The first step in dealing with the temperature dependent characteristics of the silicon neuron involved investigating the temperature dependency of each individual transconductance circuit. Using parameters that induce Class I bursting at room temperature (27°C) as a benchmark, each component was simulated at 17, 22, 27, 32, and 37°C with the Spectre circuit simulator. The current-voltage characteristics from 0 to 1 V (V_{dd}) were plotted using DC steady-state analysis. Figures 5.1 (a), 5.2 (a), 5.3 (a), 5.4 (a), and 5.5 (a) and Figures 5.6 (a) and (b) show clear temperature dependency for each of the circuits: $f_v(v)$, $f_n(v)$, $g_v(v)$, $g_n(v)$, $r(n)$, I_{av}, and I_{an}. Higher temperatures increase the range of output current and lower temperatures decrease the range of output current.

Next, to compensate for the temperature-induced change in current, circuit parameters were adjusted to achieve similar current-voltage characteristics at each of the pillar temperatures. The parameters were tuned in 0.5 mV increments, corresponding to the realistic precision of voltage biases for VLSI circuits.

5.2. Circuit Components

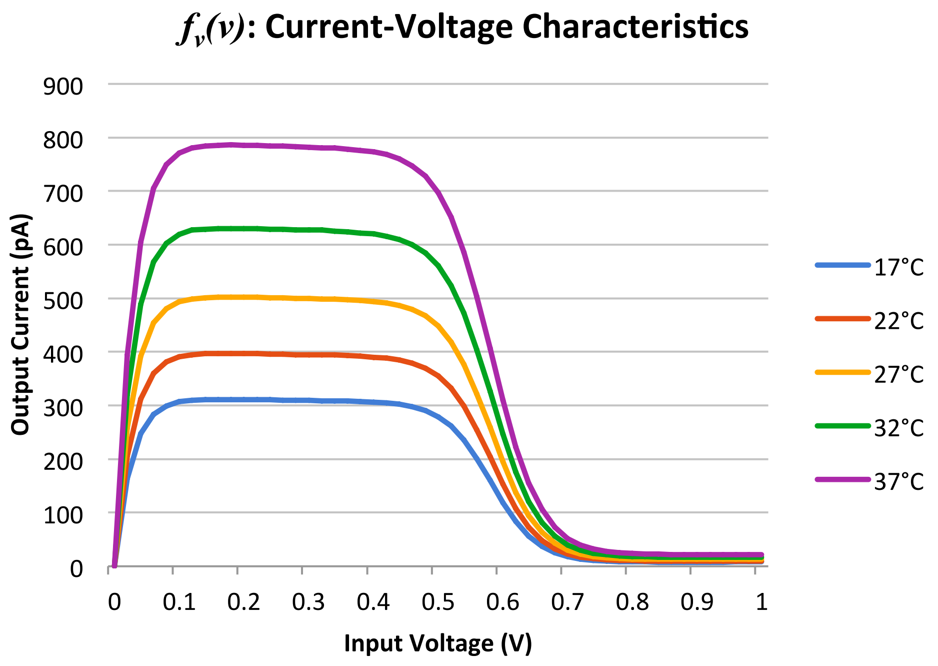
5.2.1. $f_x(v)$

In the $f_x(v)$ circuit, transistor M1 (Figure 4.2), which controls the tail current of the differential pair, has the greatest overall influence on the output current. V_b, the bias voltage on the gate of this PMOS transistor, can thus be used to scale the output current, with a voltage farther from V_{dd} increasing current and closer to V_{dd} decreasing current. By adequately adjusting the voltage bias of M1, the current-voltage characteristics can be made to match the benchmark curve at various temperatures. Figures 5.1 (b) and 5.2 (b) show the current-voltage characteristics of the $f_v(v)$ and $f_n(v)$ circuits with V_b adjusted to compensate for changes in temperature. These curves illustrate that with the right parameters, the current-voltage curves for $f_x(v)$ at the simulated temperatures can be made to nearly overlap.

Since transistor M7, the cascode stage of the current mirror, should ideally be saturated, V_{casc} has less influence on the output current and can thus be held constant.

V_{dlt}, the offset voltage of the differential pair, was kept constant at all temperatures for both $f_n(v)$ and $f_v(v)$; the curves overlapped well enough that no horizontal offset was deemed necessary.

(a)



(b)

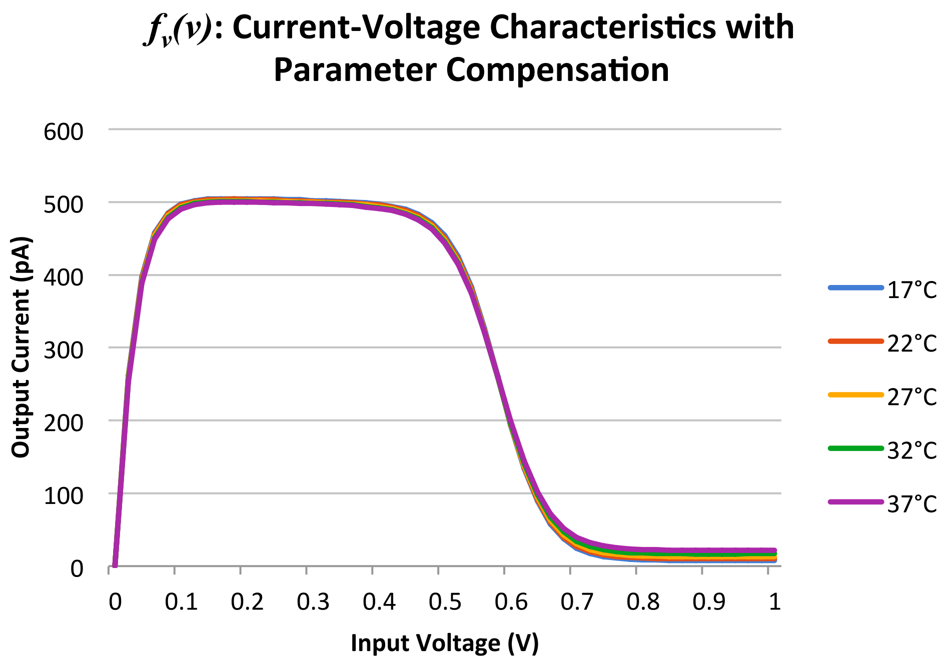
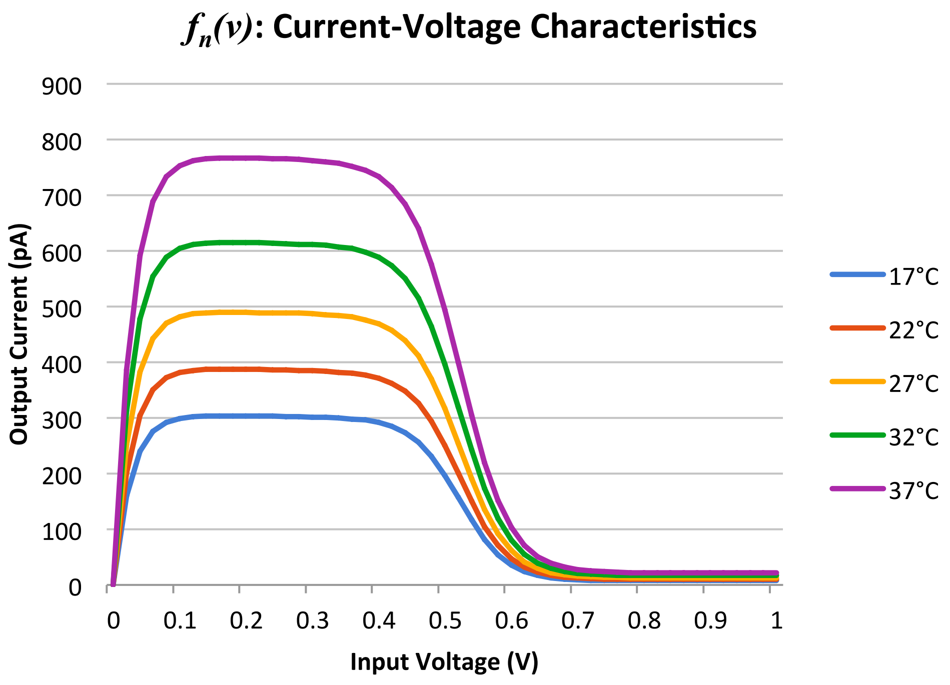


Figure 5.1: Current-voltage characteristics of $f_v(v)$ at various temperatures (a) before and (b) after parameter compensation

(a)



(b)

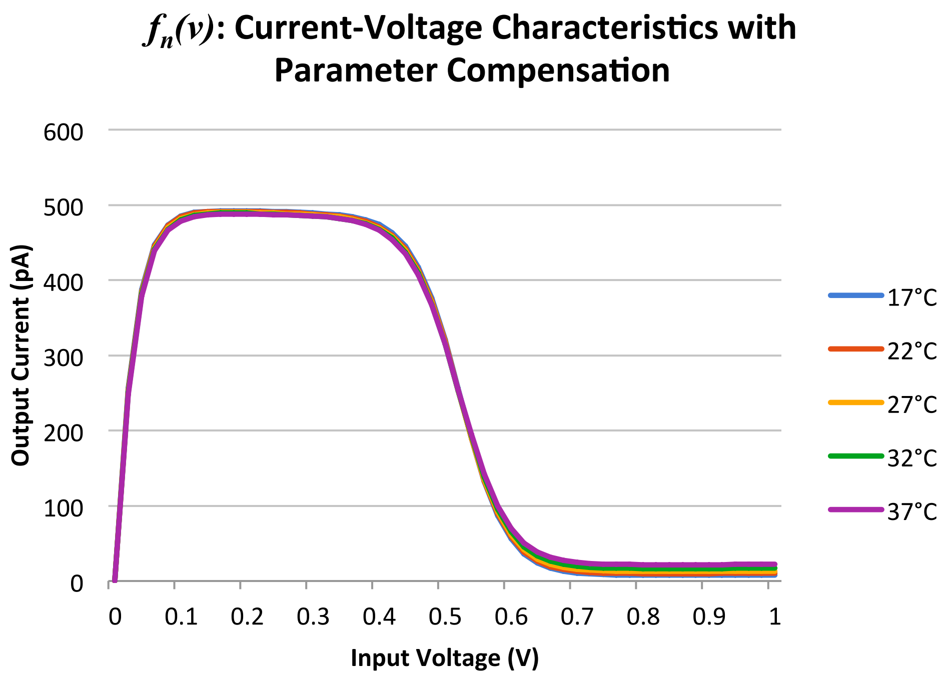


Figure 5.2: Current-voltage characteristics of $f_n(v)$ at various temperatures (a) before and (b) after parameter compensation

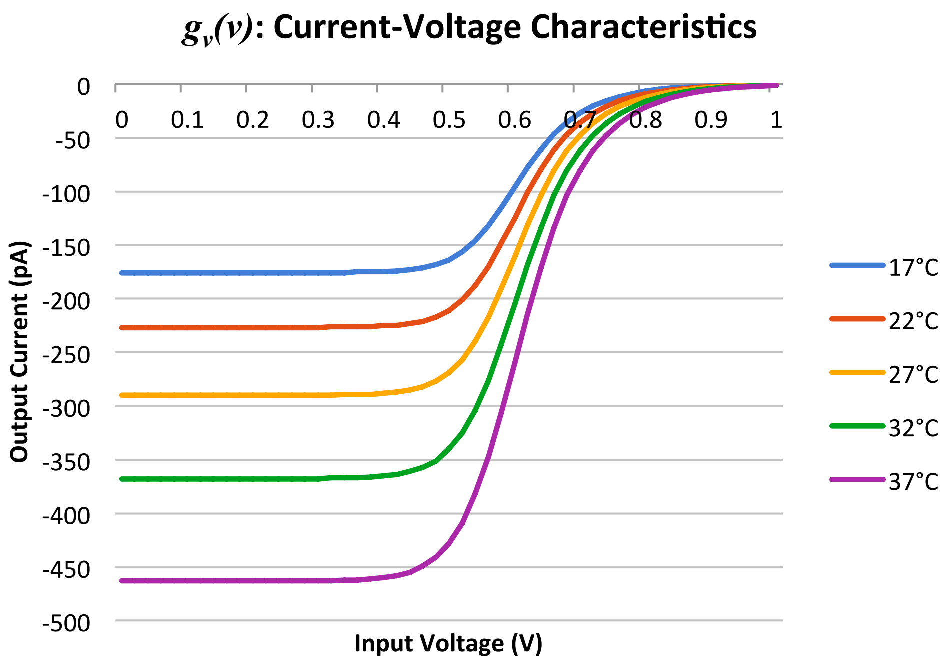
5.2.2. *$g_x(v)$ and $r(n)$*

The same method was also applied to the *$g_x(v)$* and *$r(n)$* components which are all based on the same fundamental circuit. The only tunable parameter for these circuits is V_m, the bias voltage on the gate transistor in the cascode stage. Adequately raising the voltage bias on this transistor closer to V_{dd} at higher temperatures compensates for the stronger current. Figures 5.3 (b), 5.4 (b) and 5.5 (b) show current-voltage curves for the *$g_v(v)$*, *$g_n(v)$*, and *$r(n)$* circuits with V_m adjusted for temperature changes. For all circuits, the range of the output current was matched, but higher temperatures shifted the curves rightward. Since these circuits offer only one degree of adjustment, no immediate way to compensate for this offset exists.

5.2.3. *I_{ax}*

The *I_{ax}* circuit supplies a constant current, so its current-voltage characteristics are not directly utilized by the silicon neuron. For different temperatures, simply adjusting V_{in} to yield the same output current was deemed adequate and a fine-tuning scheme dealing with V_b and V_{dlt} unnecessary. Figures 5.6 (a) and 5.6 (b) show the current-voltage characteristics of the *I_{av}* and *I_{an}* circuits. The horizontal lines represent the output currents of *I_{av}* and *I_{an}* for the 27°C benchmark settings. The intersections of these lines with the current-voltage curves indicate which voltages to use as V_{in} at the pillar temperatures to output the same current.

(a)



(b)

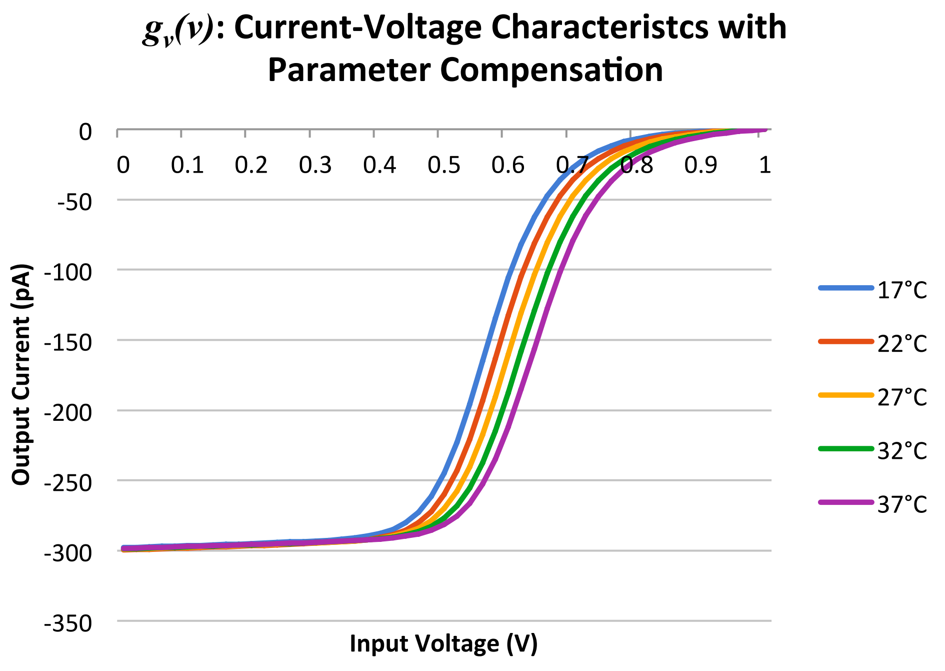
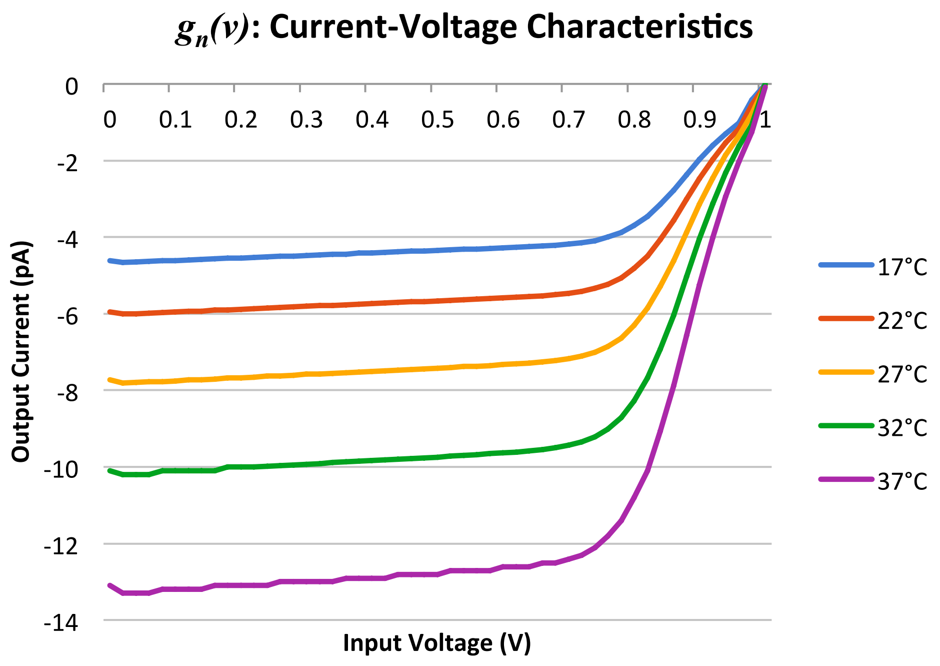


Figure 5.3: Current-voltage characteristics of $g_v(v)$ at various temperatures (a) before and (b) after parameter compensation

(a)



(b)

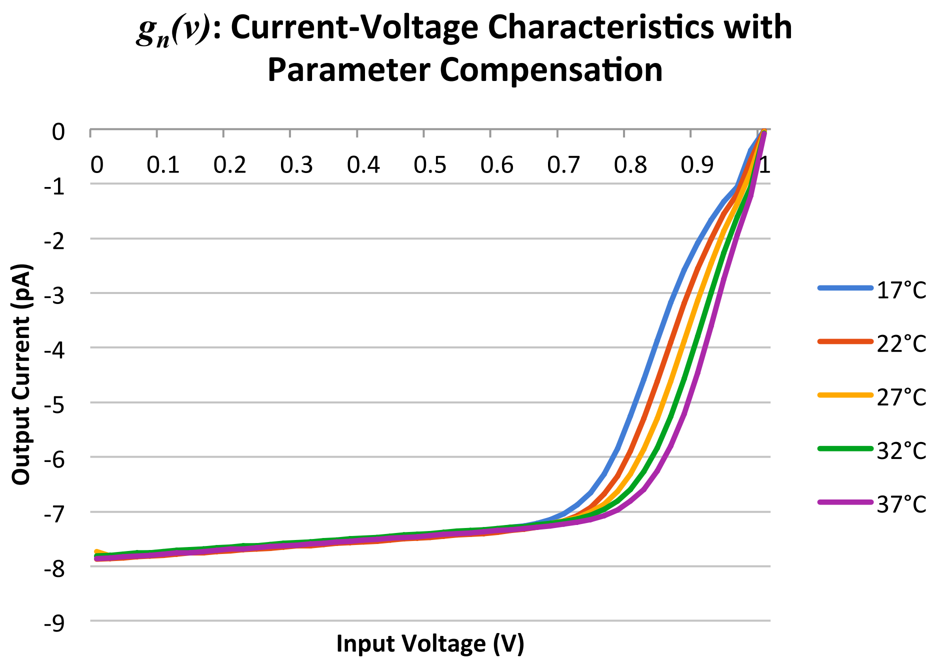
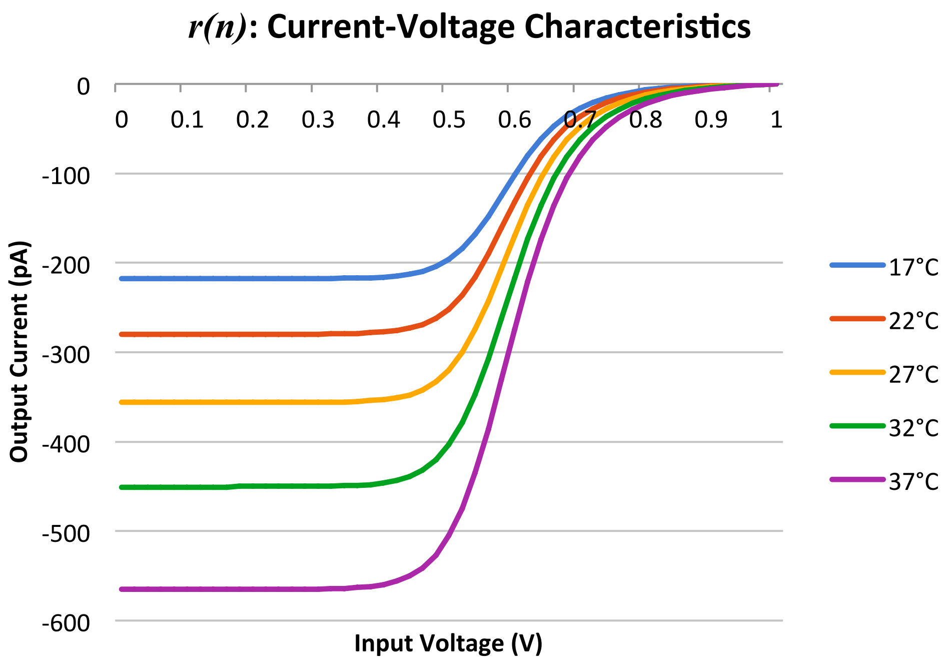


Figure 5.4: Current-voltage characteristics of $g_n(v)$ at various temperatures (a) before and (b) after parameter compensation

(a)



(b)

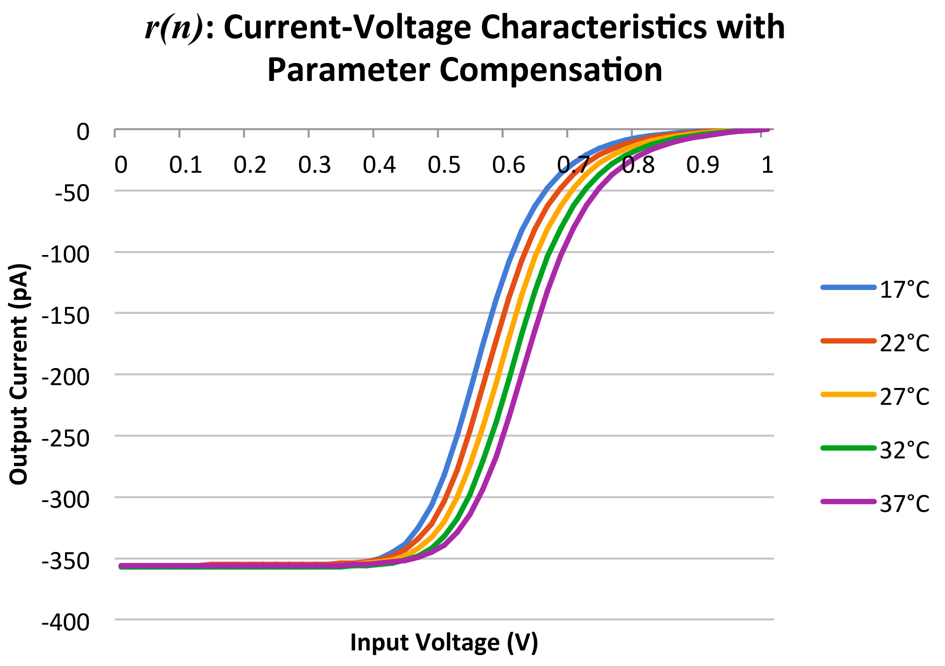
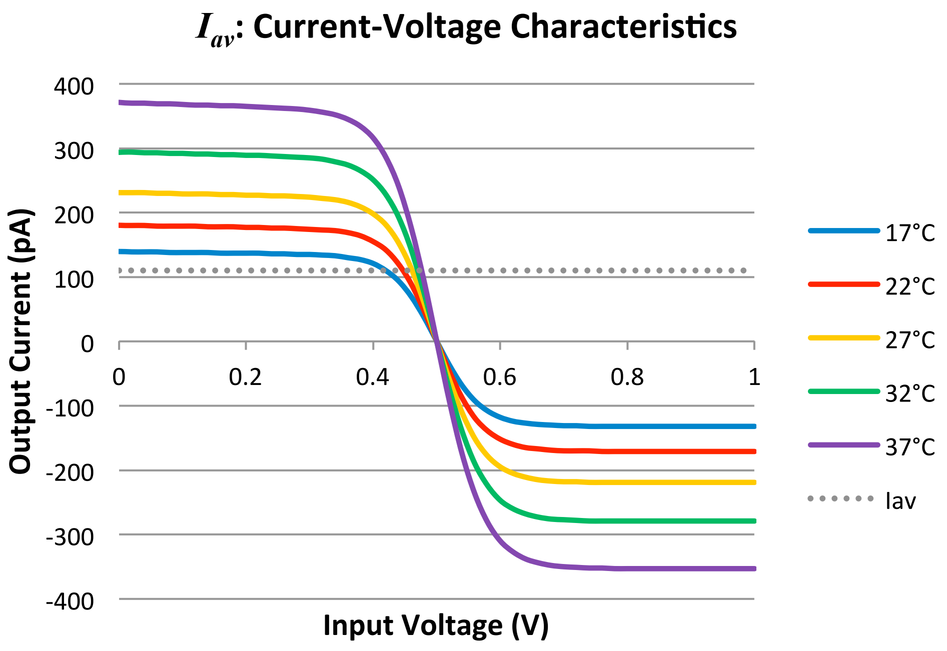


Figure 5.55: Current-voltage characteristics of *r(n)* at various temperatures (a) before and (b) after parameter compensation

(a)



(b)

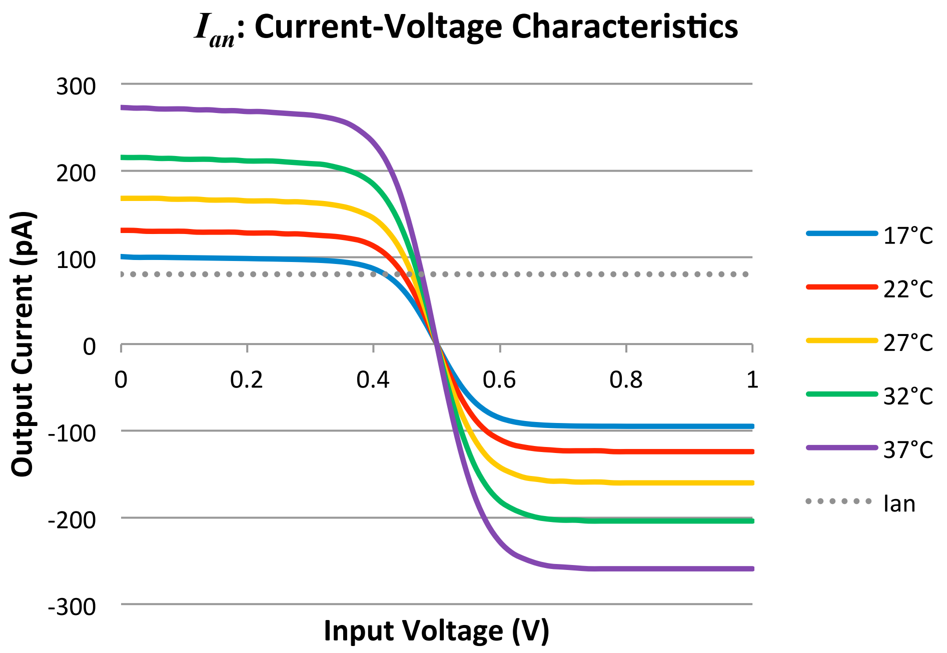


Figure 5.6: (a) I_{av} and (b) I_{an} current-voltage characteristics at various temperatures. The dotted lines "Iav" and "Ian" represent the constant currents output by each component when the silicon neuron is operated at 27°C in the benchmark Class I mode.

5.3. Tuning the nullclines and conducting transient simulations

Figure 5.7 shows a plot of the *v* and *n* nullclines of the silicon neuron operated with the 27°C benchmark parameters in Class I mode (Endnote 1). Using the parameters determined by the methods in section 5.2 as a starting point, the circuit was simulated in nullcline mode at the pillar temperatures. V_{in} of I_{av} and I_{an}, V_b of $f_v(v)$ and $f_n(v)$, V_m of $g_v(v)$ and $g_n(v)$, as well as the V_{dlt} offsets of $f_n(v)$ and $f_v(v)$ were all tuned with trial and error in an effort to make the nullclines at all temperatures match the 27°C benchmark nullclines as closely as possible. Since $r(n)$ is bypassed in nullcline mode, V_m of this circuit was temporarily ignored.

The circuit parameters affect the nullclines of their respective circuit blocks in the following ways: V_{in} of I_{av} and I_{an} govern the vertical position, V_b of $f_v(v)$ and $f_n(v)$ influence the overall curvature, the V_{dlt}'s of $f_v(v)$ and $f_n(v)$ induce horizontal offset, and V_m of $g_v(v)$ and $g_n(v)$ influence the subtle curvature of the lower cusp, which can be seen in the cubic ("s" like) shape of the *v*-nullcline. This curvature is essential for the Class I bifurcation discussed in section 2.5.

For each temperature, once the circuit parameters had been tuned to generate nullclines similar to the benchmark curves, the silicon neuron was then subjected to a transient analysis. The silicon neuron's frequency response to 5 pA and 10 pA sustained stimuli as well as the threshold current to generate an action potential with a 500 μs pulse stimulus were all recorded. The circuit parameters were then tuned to best replicate the behavior of the silicon neuron at 27°C with the benchmark parameters, plotted in row 3 of Figure 5.8. $r(n)$, which mainly affects the shape of the action potential, was reintroduced and tuned as well.

The frequency responses were determined with Spectre's frequency calculator. The threshold currents were calculated by simulating successively stronger 500 μs pulse stimuli in 0.5 pA steps. When an action potential formed, the current from the previous step was recorded as the threshold current.

Table 5.1 shows the circuit parameters for each temperature which best replicated the benchmark behavior. The circuit behaviors for each of these parameter sets are recorded in the bottom three rows of the table. Figure 5.8 shows the time-series plots of the silicon neuron simulated with the parameter sets for the pillar temperatures. Similar periodic behavior is apparent in each transient simulation. The 5 pA stimulus is applied from 0.2 to 1 second, and the 10 pA stimulus is applied from 1.2 to 2 seconds. The recorded frequencies are the average firing frequency in 0.25 to 0.75 seconds for the 5 pA stimulus and the average firing frequency in 1.25 to 1.75 seconds for the 10 pA stimulus**.** Figure 9 shows plots of the *v* and *n*-nullclines for each parameter set at its respective temperature overlaid on the 27°C benchmark nullclines for comparison. Figure 5.10 shows a plot of the circuit parameters versus temperature. Irregularity in the progression of some of the parameters is apparent.

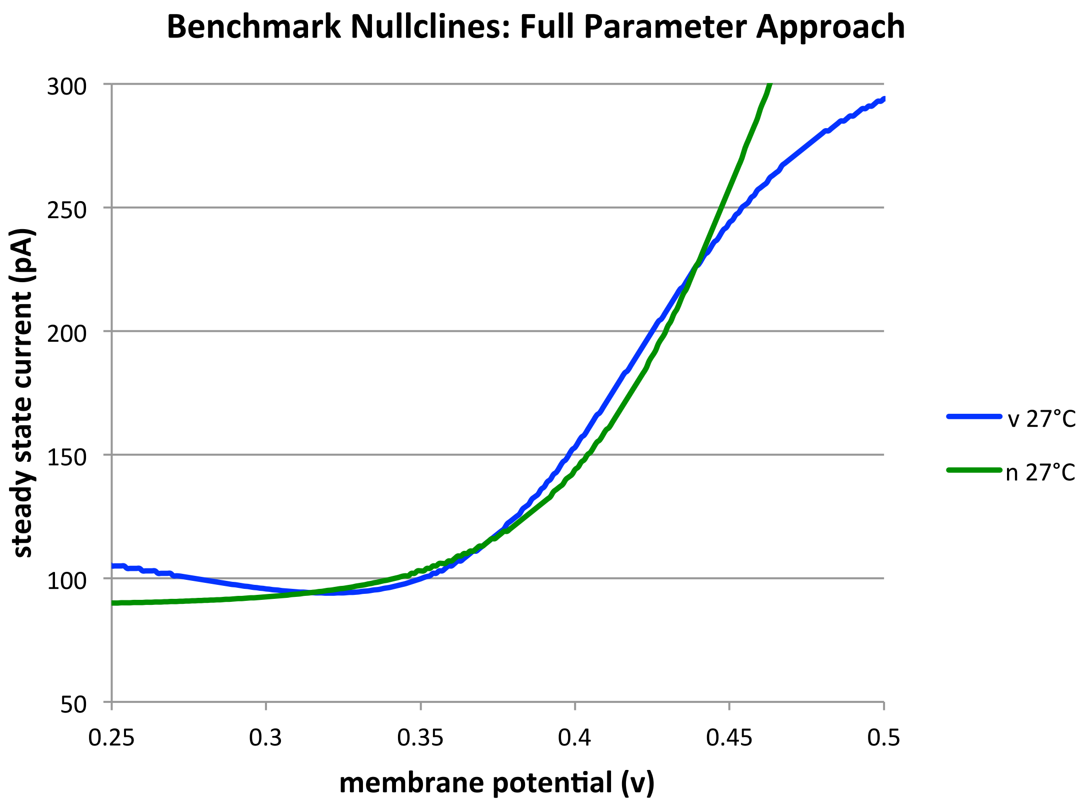


Figure 5.7: Nullclines of the silicon neuron circuit in Class I mode with benchmark parameters at 27°C

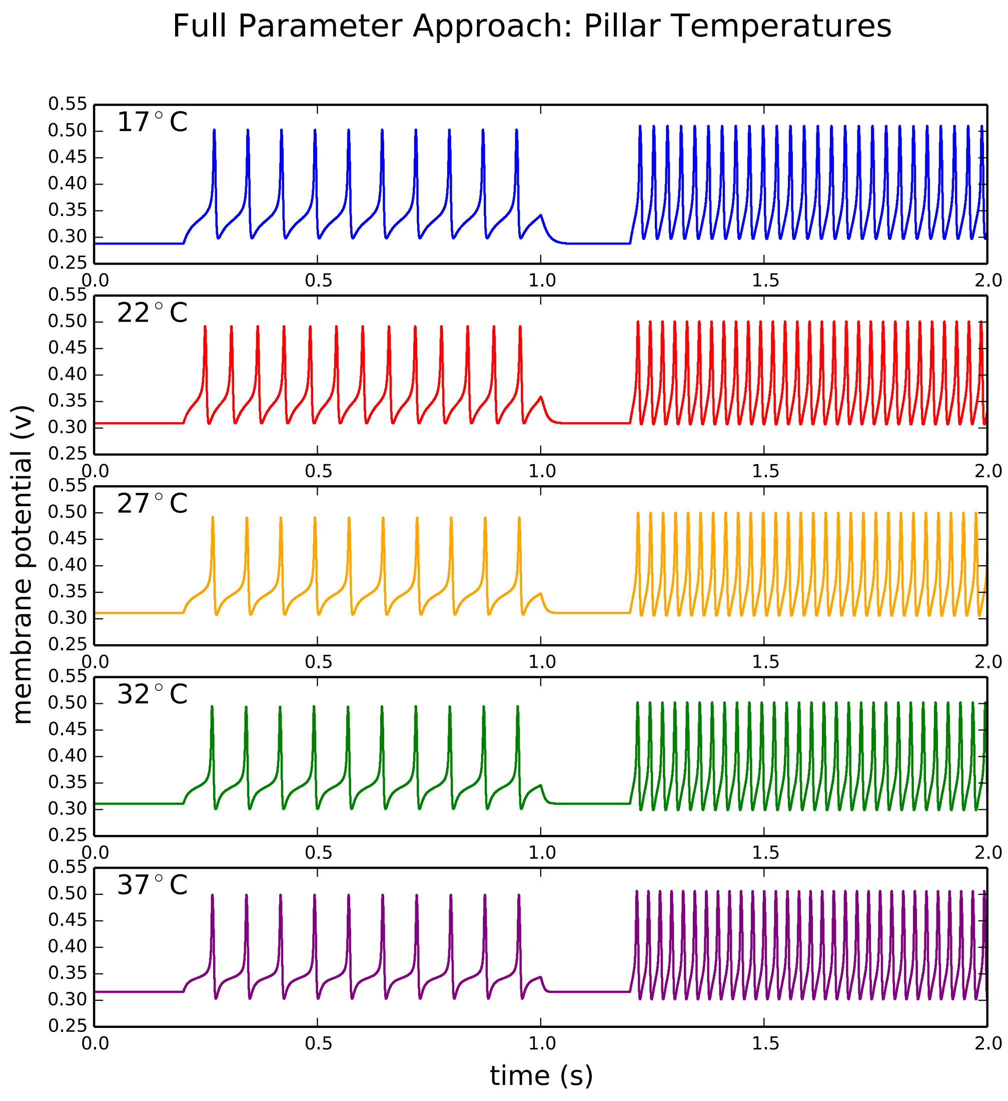


Figure 5.8: Time-series response of the silicon neuron to 5 pA and 10 pA sustained stimuli (0.2–1 and 1.2–2 seconds respectively) at pillar temperatures with parameters tuned with the full parameter approach

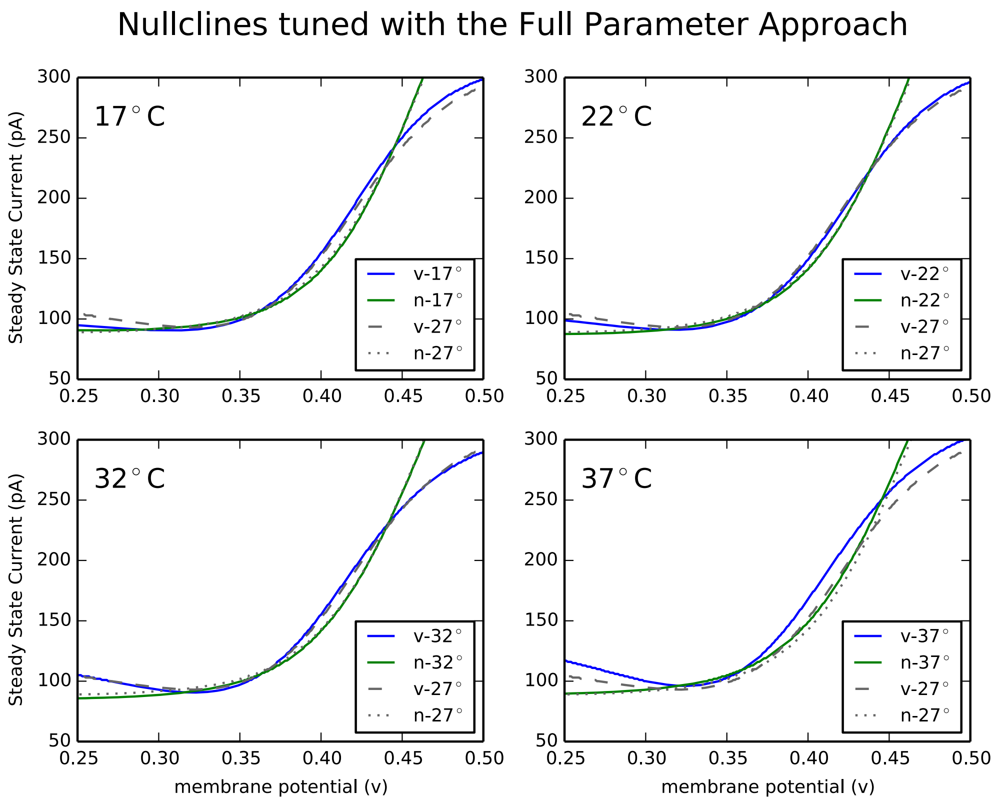


Figure 5.9: *v* and *n*-nullclines of the tuned parameter sets at pillar temperatures

Table 5.1: Circuit Parameters from the full parameter approach

Temperature (°C)	17	22	27	32	37
fv_Vb (mV)	256.5	248.5	238	230	219
fv_Vdlt (mV)	567	571	580	588	596
gv_Vm (mV)	464	451	432	420	398
fn_Vb (mV)	255	246	237	228.5	219
fn_Vdlt (mV)	517	520	520	521	524
Iav_Vin (mV)	437	452	461	469	472
gn_Vm (mV)	220	178.5	156	132.5	113
Ian_Vin (mV)	395	445.5	461	473	479
rn_Vm (mV)	480.5	463.5	445	427	413
I_{th} (pA)	250	191	186	169	145
5 pA response (Hz)	13.3	17.1	13.2	13.2	13.2
10 pA response (Hz)	32.7	36.5	35.7	36	38.6

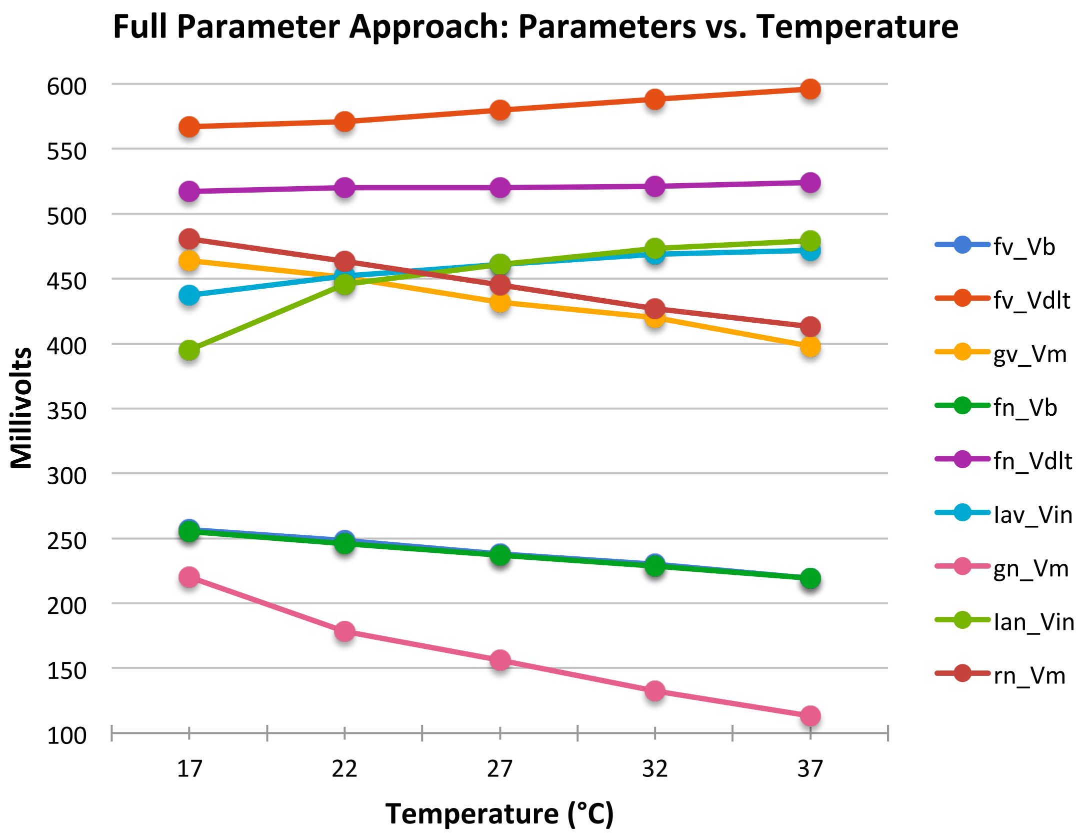


Figure 5.10: Pillar parameters versus temperature for the full parameter approach

Table 5.2: Analysis of interpolation results for the full parameter approach (0.5 mV precision)

5 pA stim average frequency	11.9 Hz
5 pA stim average % change	-9.3%
5 pA stim average | % change |	58.1%
5 pA stim frequency range	2.4-26.1 Hz
10 pA stim average frequency	35.5 Hz
10 pA stim average % change	-0.6%
10 pA stim average | % change |	7.8%
10 pA stim frequency range	26.8-40.4 Hz
Average threshold current I_{th}	185.4 pA
Average I_{th} % change	-0.1%
I_{th} average | % change |	18.5%
I_{th} range	131-303.5pA

5.4. Interpolation

These parameter sets were then input into a *Mathmatica* script which interpolates curves of the circuit parameters versus temperature and returns parameter sets for any intermediary temperature. Unlike in Figure 5.10, 2nd order smoothing was used in the interpolation. These interpolated parameter sets for temperatures from 17 to 37°C in 0.5°C steps were then input into Spectre and simulated with the same transient analysis to check the silicon neuron's threshold behavior and response to sustained stimuli. Two interpolation methods were used, one which outputs parameters to the nearest 0.5 mV, and another which outputs parameters to the nearest 0.1 mV. The 0.5 mV method matches the precision of the original tuning strategy, and the 0.1 mV method was used to see if higher interpolation precision would lead to better circuit performance.

Figure 5.11 shows the frequency response of the silicon neuron operated over the entire range of temperatures with interpolated parameters rounded to the nearest 0.5 mV. Figure 5.12 shows the same for parameters rounded to the nearest 0.1 mV. As in Figure 5.8, the 5 pA sustained stimuli was applied from 0.2 to 1 second, and the 10 pA stimulus was applied from 1.2 to 2 seconds. The simulations for pillar temperatures (the same as Figure 5.8) are highlighted in orange.

As can be seen in the figures, the 5 pA sustained stimulus failed to induce periodic firing for many intermediary temperatures. When the 5 pA stimulus did induce firing, the frequency varied greatly at different temperatures. The frequency response to the 10 pA stimulus was more consistent. Table 5.2 shows data of how the circuit behaviors from the 0.5 mV interpolation method compare to the benchmark behavior. "Average % change" refers to the average percent change from the benchmark behavior and "average | % change |" refers to the same operation using absolute values of the percent change.

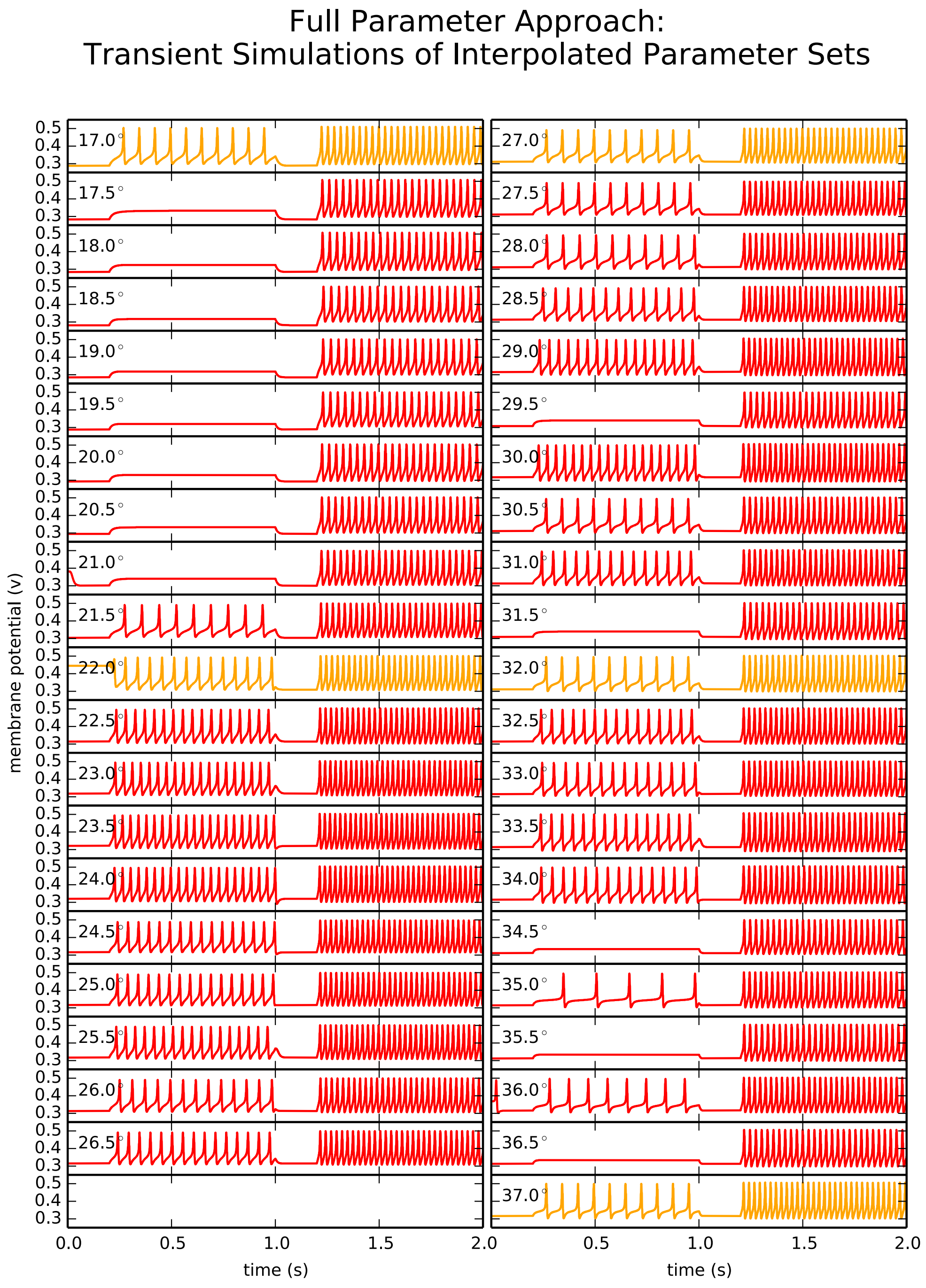


Figure 5.11: Transient simulations of the silicon neuron at temperatures from 17 to 37°C in 0.5°C steps with interpolated parameters rounded to the nearest 0.5 mV. A 5 pA sustained stimulus is applied from 0.2 to 1 second and a 10 pA sustained stimulus is applied from 1.2 to 2 seconds.

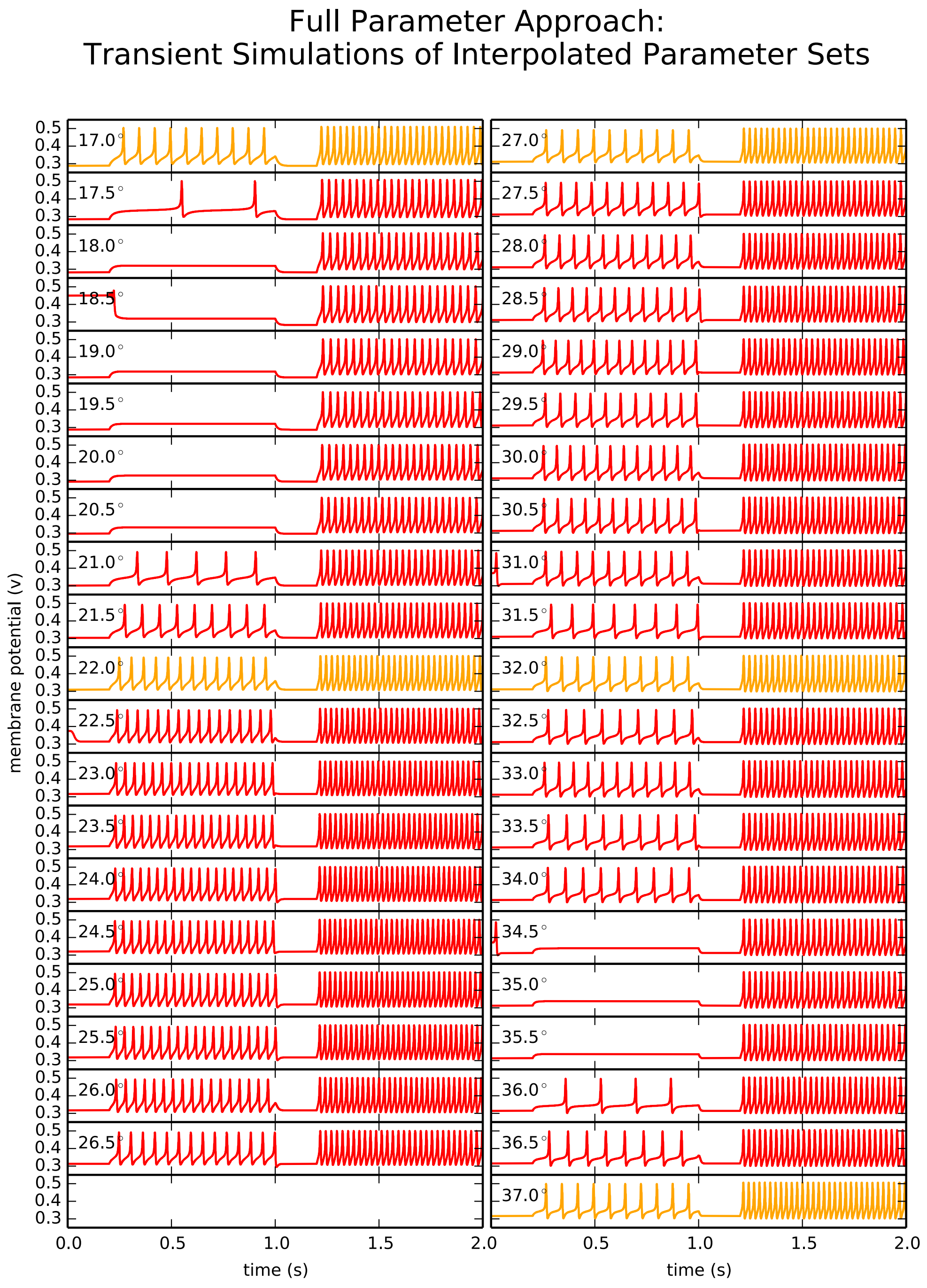


Figure 5.12: Transient simulations of the silicon neuron at temperatures from 17 to 37°C in 0.5°C steps with interpolated parameters rounded to the nearest 0.1 mV. A 5 pA sustained stimulus is applied from 0.2 to 1 second and a 10 pA sustained stimulus is applied from 1.2 to 2 seconds.

5.5. Endnotes

1) Table 5.3 lists the terms of the system equations and their associated parameters in the Spectre simulator.

Table 5.3: Spectre Circuit Parameters

Function	Related Term	Spectre Parameter	Benchmark Voltage
I_{stim}		Istim_Vb	300m
		Istim_Vdlt	500m
		Istim_Vin	500m
$f_v(v)$	M_v	fv_Vb	238m
	M_v	fv_Vcasc	300m
	M_v	fv_Enx1	0
	M_v	fv_Enx2	0
	δ_v	fv_Vdlt	580m
$g_v(v)$	θ_v	gv_Vm	432m
I_{av}		Iav_Vin	461m
		Iav_Vdlt	500m
		Iav_Vb	413m
$f_n(v)$	M_n	fn_Vb	237m
	M_n	fn_Vcasc	300m
	M_n	fn_Enx1	0
	M_n	fn_Enx2	0
	δ_n	fn_Vdlt	520m
$g_n(v)$	θ_n	gn_Vm	156m
	R_2	gn_R20n	0
	R_2	gn_R20p	2.5
	R_2	gn_R21n	0
	R_2	gn_R21p	2.5
I_{an}		Ian_Vin	461m
		Ian_Vdlt	500m
		Ian_Vb	390m
$r(n)$	θ_r	rn_Vm	445m

Chapter 6: Limited Parameter Approach

The parameters versus temperature plot for the full parameter approach (Figure 5.10) sometimes shows irregular progressions because some parameters were tuned more aggressively than others to achieve similar circuit behavior. The degree of available control was unwieldy and difficult to use, which suggests that a more efficient tuning strategy could make use of fewer parameters.

For the limited parameter approach, 5 influential circuit parameters were selected: the bias voltage V_b which governs the tail current of $f_n(v)$, the bias voltages V_m of the cascoded stages of $g_v(v)$ and $r(n)$, and V_{in} of the constant current sources I_{av} and I_{an}. All other circuit parameters were held constant. Since the $g_n(v)$ circuit is unnecessary for creating a properly shaped n-nullcline for Class I behavior, this circuit was turned off by setting its V_m bias voltage to V_{dd}.

Without the influence of $g_n(v)$, new Class I benchmark circuit parameters and behaviors at 27°C had to be determined. Column 4 of Table 6.1 shows these benchmark parameters and the corresponding circuit behaviors at 27°C. Figure 6.1 shows a plot of the benchmark nullclines and row 3 of Figure 6.2 shows the frequency response of the silicon neuron at 27°C with the new benchmark parameters.

As in the previous method, the silicon neuron was simulated at 17, 22, 32, and 37°C and the selected circuit parameters were tuned to match the benchmark nullclines. Then the parameters were retuned to most accurately replicate the 27°C benchmark Class I behavior. Parameters were tuned in 0.5 mV increments. Table 1 shows the pillar parameters sets for each temperature along with their recorded behaviors, Figure 6.3 shows the nullclines for each parameter set overlaid on the benchmark nullclines for comparison, and Figure 6.4 shows a plot of the parameters versus temperature. The progression of parameters in this plot is noticeably smoother than in the plot for the full parameter approach as recorded in Figure 5.10.

Again, these pillar parameter sets were input into a *Mathematica* script which interpolates functions of the parameters versus temperature and returns parameter sets for all the intermediary temperatures in the range from 17 to 37°C in 0.5°C steps. Again, two interpolation methods were used, one with output parameters rounded to the nearest 0.5 mV, and one with parameters rounded to the nearest 0.1 mV. The transient simulation results for these parameter sets are recorded in Figures 6.5 and 6.6. For the parameters with 0.5 mV precision, only the 5 pA sustained stimulus at 29°C failed to induce firing. For the parameters with 0.1 mV precision, all stimuli induced firing. The frequency responses to the 5 pA stimuli varied, but the responses to the 10 pA stimuli were more consistent. The threshold current was also recorded with the same method as described in section 5.3.

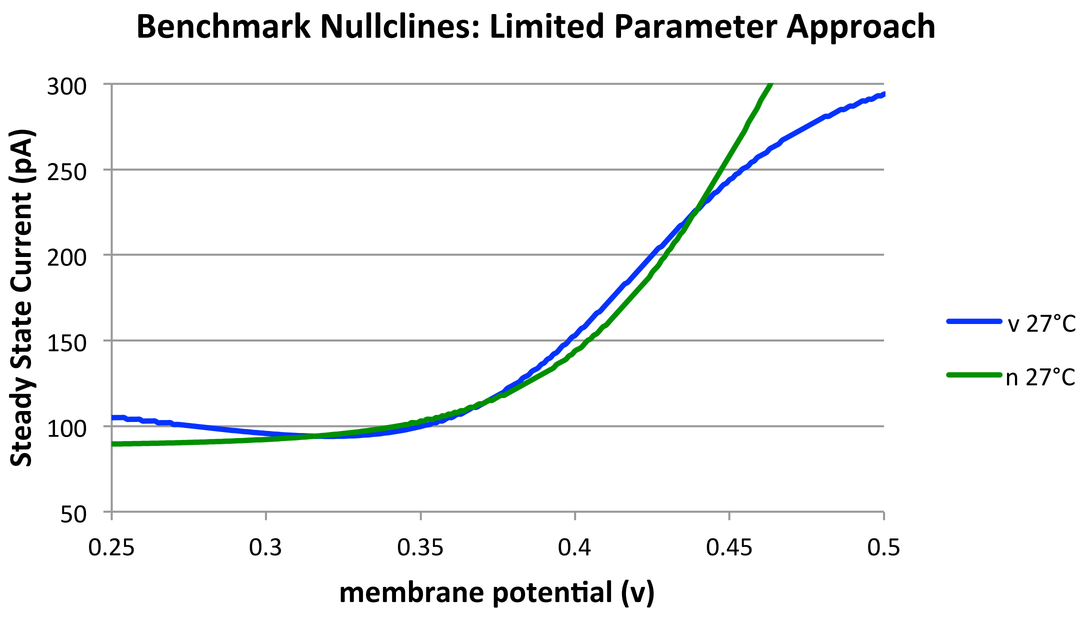


Figure 6.1: Benchmark nullclines for the limited parameter approach

Table 6.1: Circuit parameters from the limited parameter approach

Temperature (°C)	17	22	27	32	37
gv_Vm (mV)	388	417	432	443	449.5
fn_Vb (mV)	257	248	237	231	219
Iav_Vin (mV)	434	450	461	468.5	473.5
Ian_Vin (mV)	420	449.5	464.5	475	481.5
rn_Vm (mV)	480.5	462.5	445	430	415
I_{th} (pA)	201	185	178.5	164	147
5 pA response (Hz)	16.3	15.8	15.1	15.7	18.5
10 pA response (Hz)	39	38.8	36.2	38.7	39.4

Table 6.2 includes quantitative analysis of how the data from both interpolation methods compare to the benchmark behavior data. The time-series plots and the data in the table indicate that the limited parameter approach was more effective at achieving consistent circuit behavior than the full parameter approach. Increasing the precision of the interpolation also further improved the results.

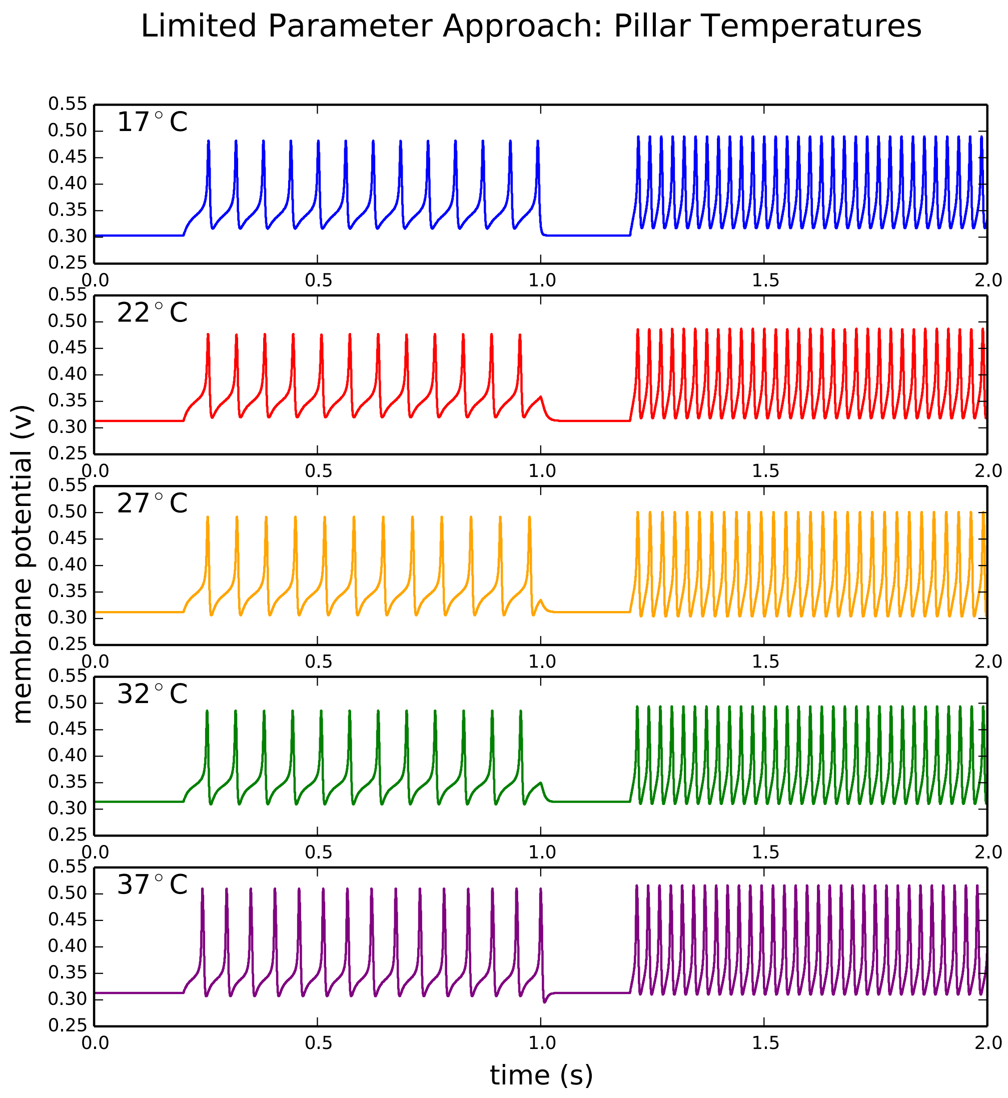


Figure 6.2: Time-series response of the silicon neuron to a 5 pA and 10 pA sustained stimulus (0.2-1 and 1.2-2 seconds respectively) at pillar temperatures with parameters tuned with the limited parameter approach

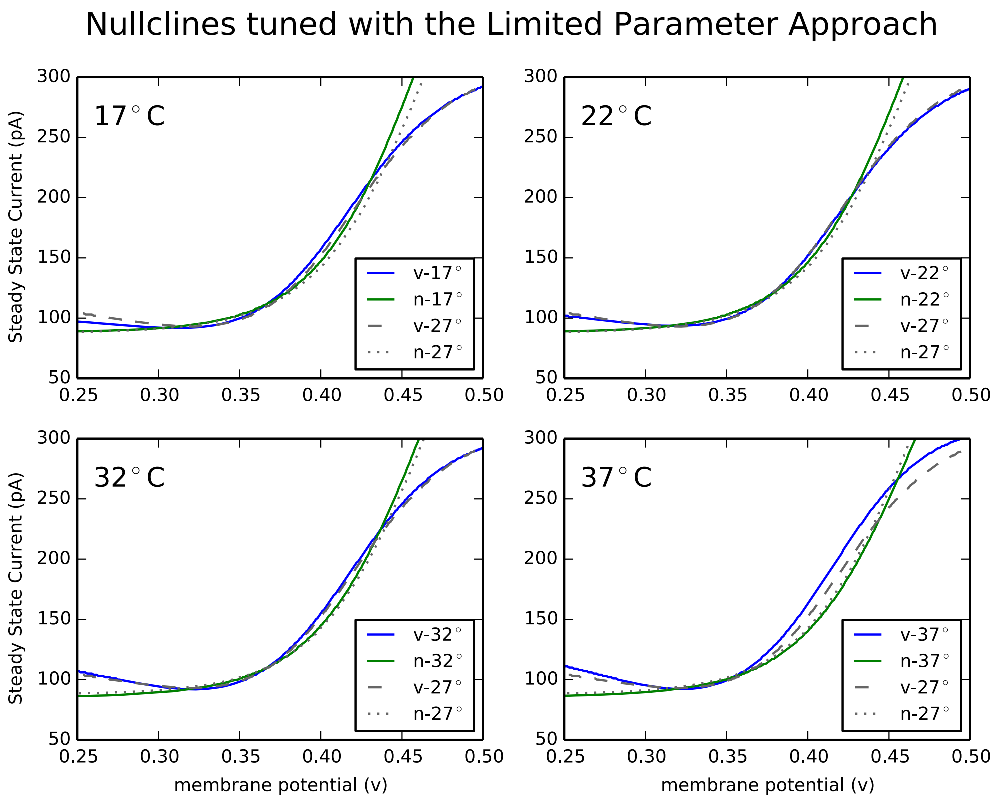


Figure 6.3: *v* and *n*-nullclines of the tuned parameter sets at pillar temperatures

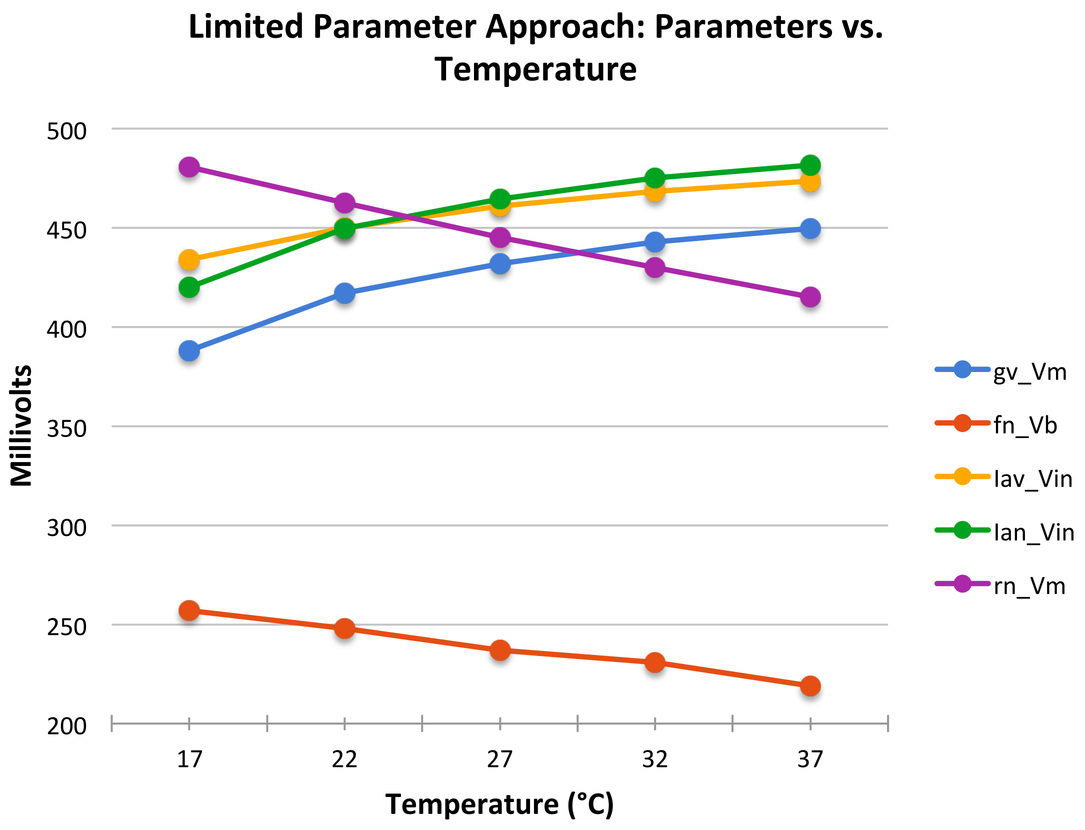


Figure 6.4: Pillar parameters versus temperature for the limited parameter approach

Table 6.2: Analysis of interpolation results for limited parameter approach with 0.5 mV and 0.1 mV interpolation precision

Interpolation precision	**0.5 mV**	**0.1 mV**
5 pA stim average frequency	15.8 Hz	15.2 Hz
5 pA stim average % change	5.2%	0.9%
5 pA stim average | % change |	34.7%	18.2%
5 pA stim frequency range	3.7-27.1 Hz	6.4-20 Hz
10 pA stim average frequency	38.7 Hz	38.2 Hz
10 pA stim average % change	6.8%	5.6%
10 pA stim average | % change |	7.1%	5.6%
10 pA stim frequency range	34.6-43.2 Hz	36.1-40.4 Hz
Average threshold current I_{th}	172 pA	176pA
Average I_{th} % change	-3.6%	-1.4%
I_{th} average | % change |	12.2%	8.7%
I_{th} range	115.5-216 pA	147-213 pA

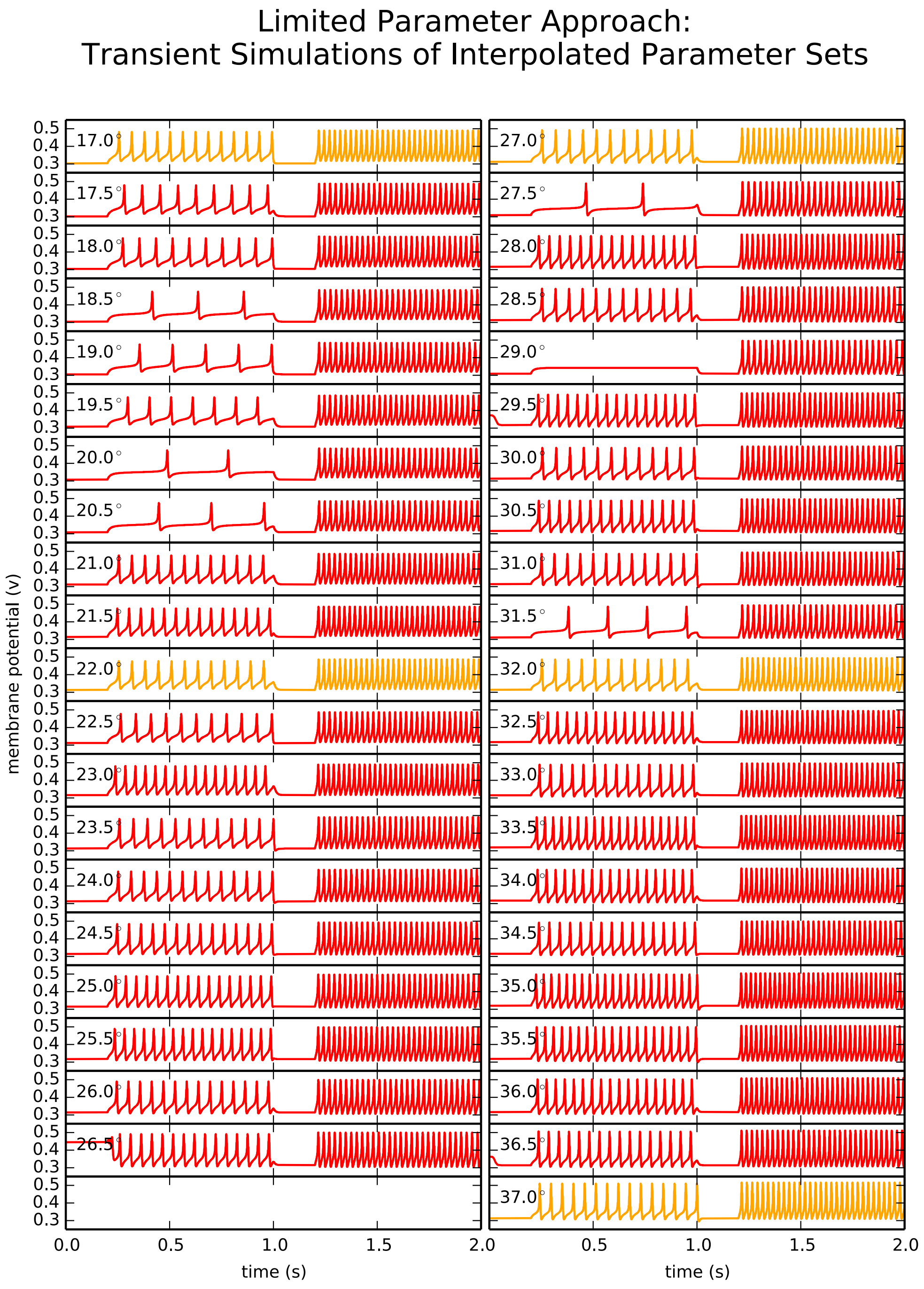


Figure 6.5: Transient simulations of the silicon neuron at temperatures from 17 to 37°C in 0.5°C steps with interpolated parameters rounded to the nearest 0.5 mV. A 5 pA sustained stimulus is applied from 0.2 to 1 second and a 10 pA sustained stimulus is applied from 1.2 to 2 seconds.

Limited Parameter Approach:
Transient Simulations of Interpolated Parameter Sets

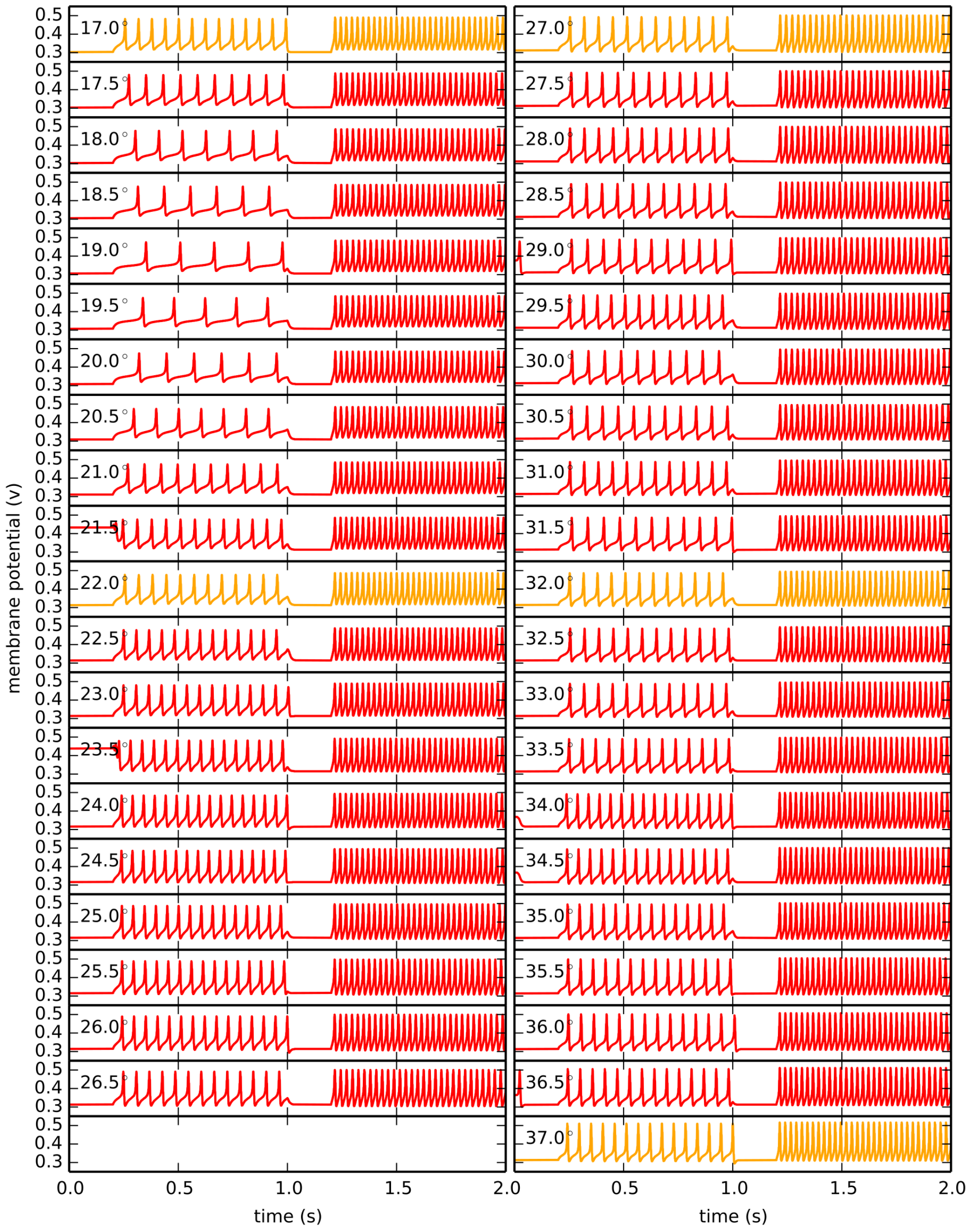


Figure 6.6: Transient simulations of the silicon neuron at temperatures from 17 to 37°C in 0.5°C steps with interpolated parameters rounded to the nearest 0.1 mV. A 5 pA sustained stimulus is applied from 0.2 to 1 second and a 10 pA sustained stimulus is applied from 1.2 to 2 seconds.

Chapter 7: Brute Force Optimization

7.1. Automated Optimization

Tuning the silicon neuron with trial and error was straightforward and effective but also time-consuming and imprecise. Adequate parameter sets were found, but without checking all of the possibilities, one can never know if better parameter sets exist, even in the nearby vicinity. Furthermore, adding additional temperature steps greatly increases the total time necessary to tune the silicon neuron. Clearly, a more efficient approach would entail an automated optimization strategy that analyzes numerous parameter combinations.

To accomplish this, a script was written in Python that can call the Spectre simulator directly from the terminal and analyze the simulation data without activating the GUI. This script was then integrated with various optimization functions available in the SciPy library.

7.2. Brute Force

The first SciPy function used was *scipy.optimize.brute()* [15], a brute force algorithm which checks all possible parameter combinations in a set range to search for the minimum of a function, here referred to as the *cost function*. The algorithm also employs a second step which uses the values of the cost function in the neighborhood of the minimum to predict a point between parameter combinations where the true solution may fall.

The cost function was written to account for the three significant circuit behaviors: the threshold current to generate an action potential, the frequency response to a 5 pA sustained stimulus, and the frequency response to a 10 pA sustained stimulus. The input vector **x** for the cost function is comprised of the circuit parameters.

The Spectre simulator was set to conduct a 4 second transient simulation which first fired successively stronger pulse stimuli to determine the threshold, and then applied two 0.8 second sustained stimuli, one at 5 pA and the other at 10 pA. The frequency responses to the sustained stimuli were calculated with a counter that noted the times when the membrane potential rose above 440 mV.

To approximate the threshold current, the current of the pulse stimuli (500 μs) were stepped in 8.9 pA intervals, corresponding to 5% of the benchmark threshold current, 178.5 pA. The time that a full action potential forms was used as an index value that represents a rough approximation of the percent difference from the benchmark threshold, to a range within 5%. This was calculated by a counter that noted the first time the membrane potential rose above 450 mV.

These three behaviors were then made into a vector **n** and subtracted from **b**, a vector of the corresponding 27°C benchmark behaviors. Vector **n** is a function of parameter input vector **x** since its values must be determined by a Spectre simulation. The magnitude of this difference vector was then used as the value for the cost function *f(***x***)*.

$$\mathbf{n}(\mathbf{x}) = \begin{bmatrix} \text{threshold index} \\ \text{5 pA stimulus response} \\ \text{10 pA stimulus response} \end{bmatrix}, \ \mathbf{b} = \begin{bmatrix} 1.21 \\ 15.87 \\ 37.18 \end{bmatrix} \tag{7.1}$$

$$f(\mathbf{x}) \triangleq \sqrt{(\mathbf{n}(\mathbf{x}) - \mathbf{b})^2} \tag{7.2}$$

The vectors **n** and **b** are somewhat crude, but the relative weights of each element to the 10 pA frequency response can be defined as follows:

$$\mathbf{w} = \begin{bmatrix} 0.0325 \\ 0.427 \\ 1 \end{bmatrix} \tag{7.3}$$

This cost function puts the most weight on the frequency response to a 10 pA sustained stimulus and the least weight on the threshold current.

With the cost function tied to the Python code, this brute force algorithm was used to fine-tune the results from the limited parameter approach. A 2 mV range around each parameter from the limited parameter approach was used to create the parameter matrices for the brute force algorithm shown in Table 7.1. Each of the 5 parameters include 5 steps, meaning $5^5 = 3125$ possible combinations of parameters to input to the Spectre simulator. One run of the Spectre simulator and cost function took approximately 4 seconds, meaning an entire run of the brute force algorithm for one pillar temperature took approximately 3.5 hours.

Table 7.2 shows the results of the brute force optimization algorithm rounded to the nearest 0.1 mV for the each temperature. Figure 7.3 shows a plot of these parameters versus temperature and Figure 7.1 shows the nullclines. The progressions of parameters versus temperature are similar to those of the limited parameter approach and relatively smooth.

These results were then input into the interpolation script and sets of parameters for 17 to 37°C in 0.5°C steps were generated and subjected to transient simulations. Figure 7.4 shows all the transient results for the interpolation with 0.1 mV precision. Table 7.3 shows analysis of how the parameter sets found by this method compare to the benchmark. Again, "average % change" refers to the average percent change from the benchmark behavior and "average | % change |" refers to the same operation using absolute values of the percent change. The average absolute values of the percent change showed improvement for the threshold current I_{th} and the frequency response to a 10 pA sustained stimulus. For the 5 pA sustained stimulus, this index was slightly poorer but comparable. These data show quantitatively that tuning with brute force optimization improved the results of the interpolated parameter sets. A complete discussion of these results can be found in section 9.1.

Table 7.1: Brute Force Matrices

gv_Vm	387	387.5	**388**	388.5	389
fn_Vb	256	256.5	**257**	257.5	258
Iav_Vin	433	433.5	**434**	434.5	435
Ian_Vin	419	419.5	**420**	420.5	421
rn_Vm	479.5	480	**480.5**	481	481.5

17°C

416	416.5	**417**	417.5	418
247	247.5	**248**	248.5	249
449	449.5	**450**	450.5	451
448.5	449	**449.5**	450	450.5
461.5	462	**462.5**	463	463.5

22°C

gv_Vm	442	442.5	**443**	443.5	444
fn_Vb	230	230.5	**231**	231.5	232
Iav_Vin	467.5	468	**468.5**	469	469.5
Ian_Vin	474	474.5	**475**	475.5	476
rn_Vm	429	429.5	**430**	430.5	431

32°C

448.5	449	**449.5**	450	450.5
218	218.5	**219**	219.5	220
472.5	473	**473.5**	474	474.5
480.5	481	**481.5**	482	482.5
414	414.5	**415**	415.5	416

37°C

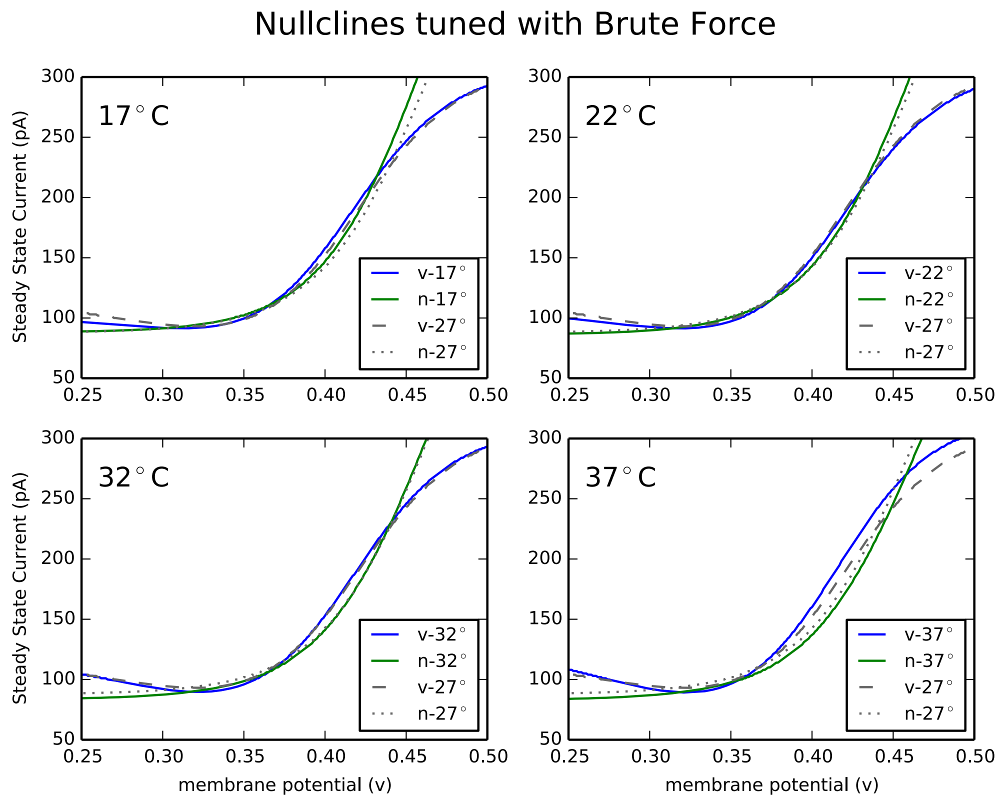


Figure 7.1: Nullclines from results of brute force optimization

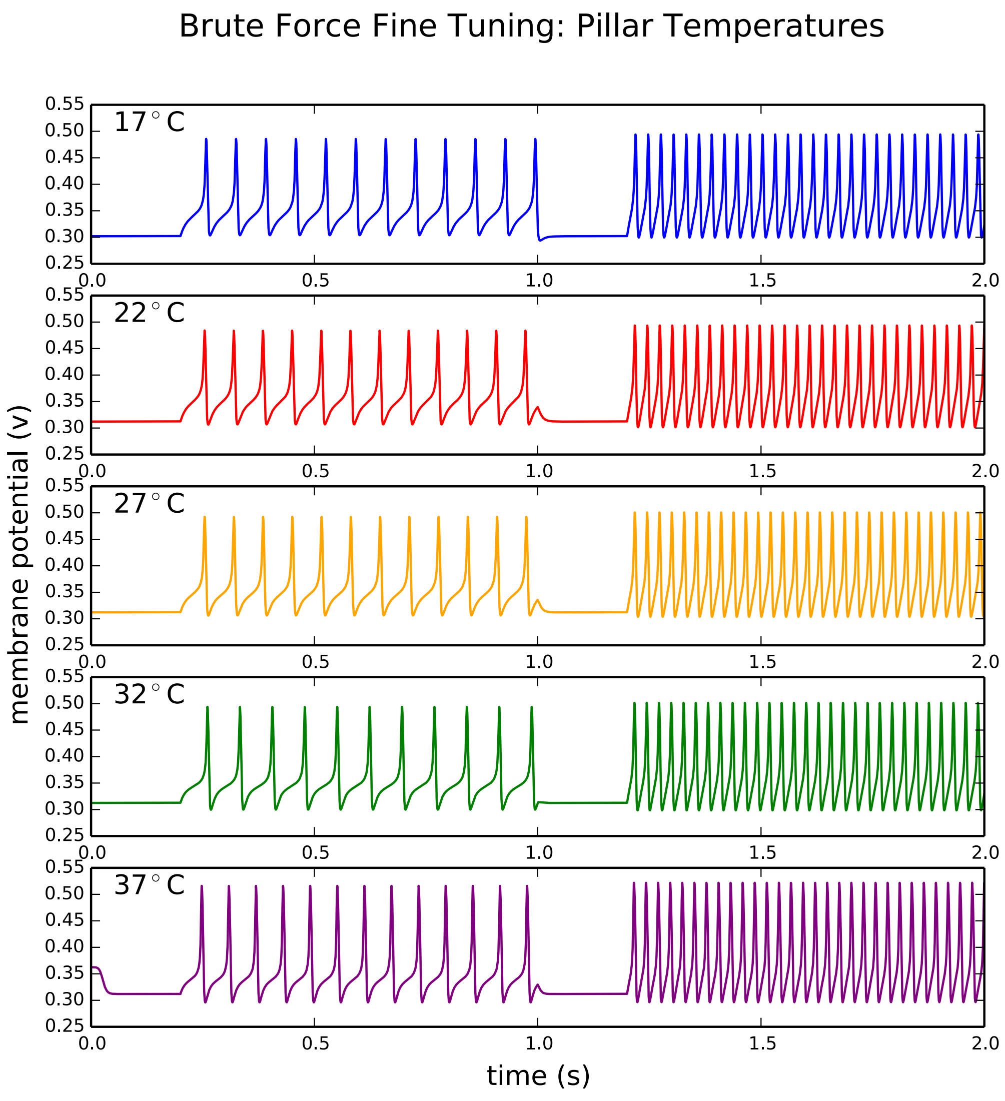


Figure 7.2: Time-series response of the silicon neuron to 5 pA and 10 pA sustained stimuli (0.2-1 and 1.2-2 seconds respectively) at pillar temperatures with parameters tuned with brute force optimization

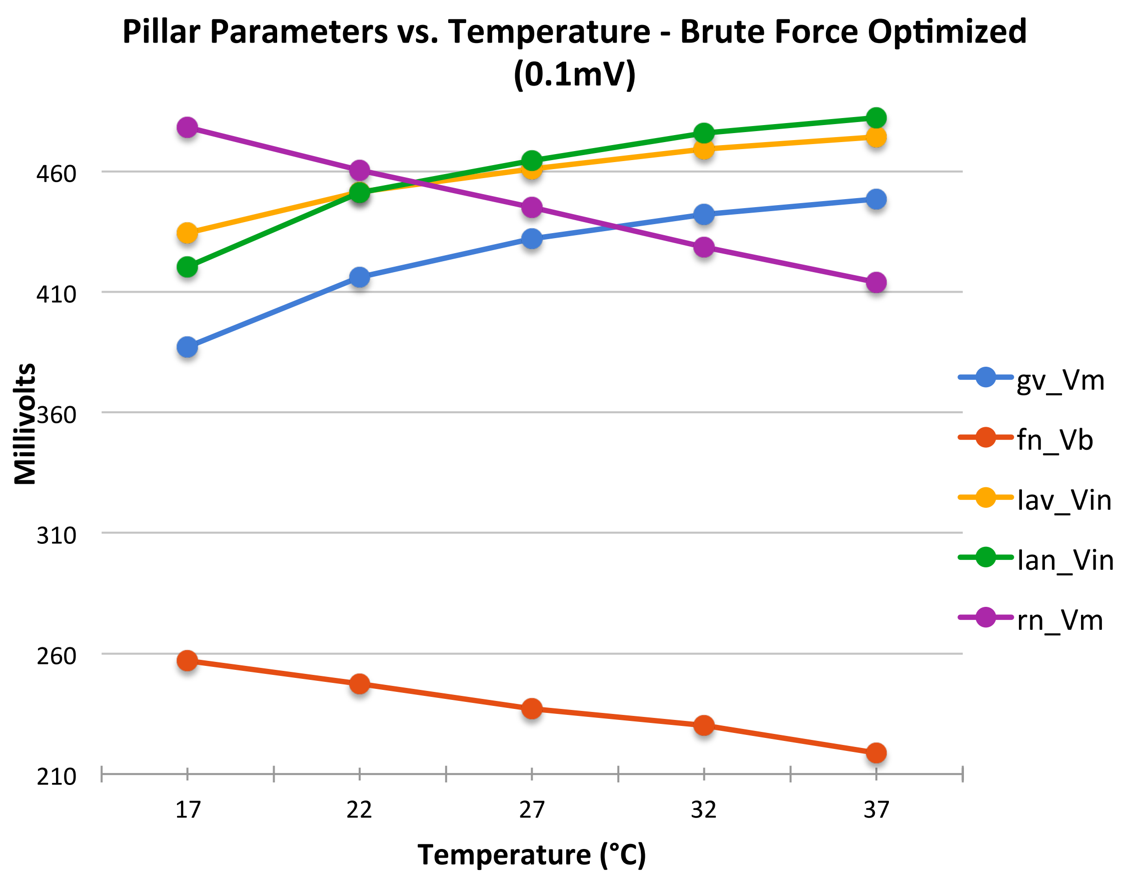


Figure 7.3: Pillar parameters vs. temperature for brute force optimization

Table 7.2: Circuit parameters for brute force optimization

Temperature (°C)	17	22	27	32	37
gv_Vm (mV)	387	416.1	432	442.2	448.6
fn_Vb (mV)	257	247.4	237	230.1	218.7
Iav_Vin (mV)	434.4	451.4	461	469.2	474.2
Ian_Vin (mV)	420.4	451.2	464.5	475.7	482.3
rn_Vm (mV)	478.2	460.3	445	428.6	413.9
Ith (pA)	1.4	1.2	178.5	1	0.04
5 pA response (Hz)	15.8	16.1	15.1	14.4	17.2
10 pA response (Hz)	37.2	37.1	36.2	37.1	37.4

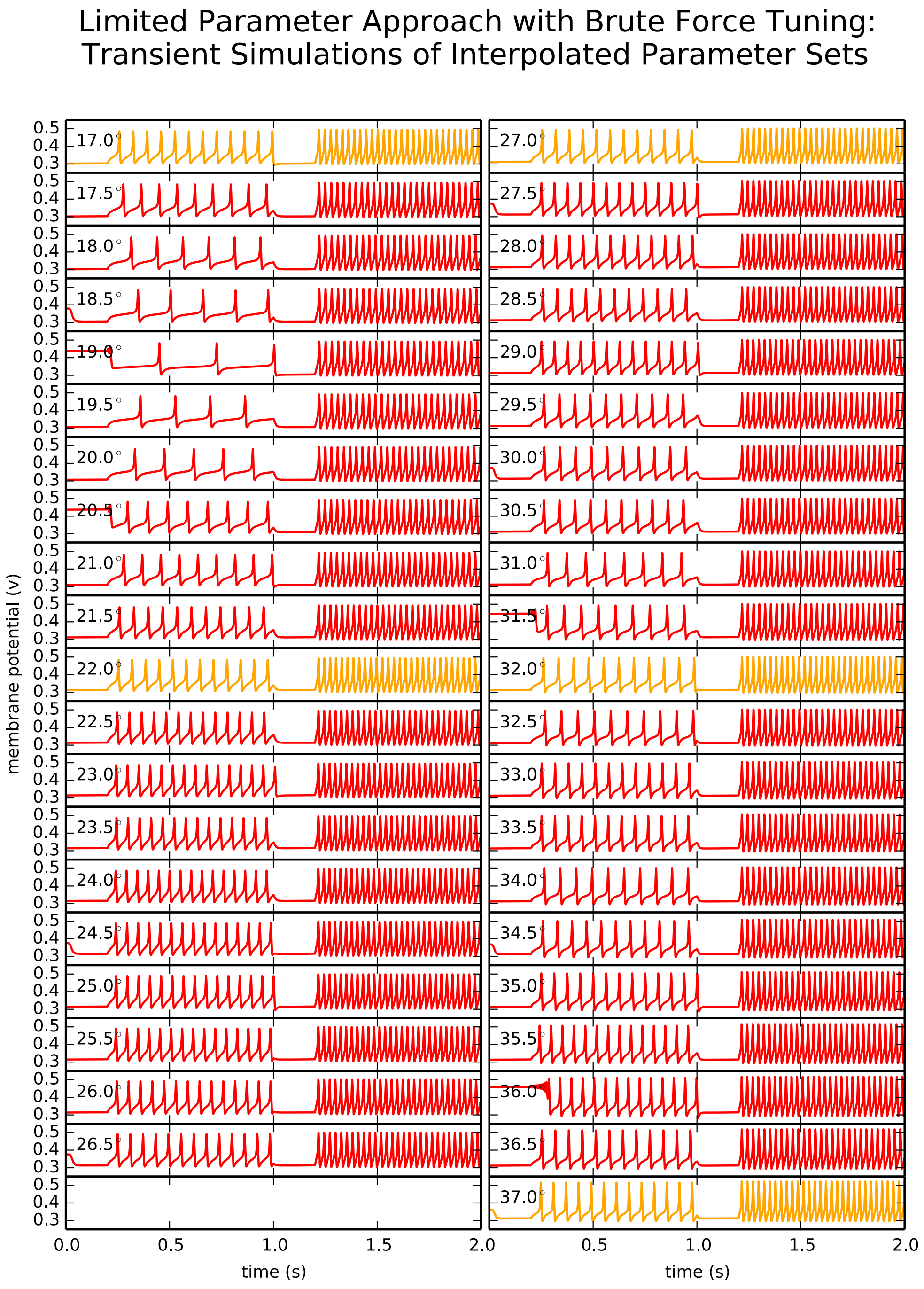


Figure 7.4: Transient simulations of the silicon neuron at temperatures from 17 to 37°C in 0.5°C steps with interpolated parameters rounded to the nearest 0.1 mV. Pillar temperatures were optimized with brute force. A 5 pA sustained stimulus is applied from 0.2 to 1 second and a 10 pA sustained stimulus is applied from 1.2 to 2 seconds.

Table 7.3: Analysis of interpolation results for brute force tuning

5 pA stim average frequency	14.8 Hz
5 pA stim average % change	-7.9%
5 pA stim average | % change |	18.4%
5 pA stim frequency range	3.4-20.1 Hz
10 pA stim average frequency	37.0 Hz
10 pA stim average % change	-0.2%
10 pA stim average | % change |	2.2%
10 pA stim frequency range	34.3-38.5 Hz
Average threshold current I_{th}	177.5 pA
Average I_{th} % change	-0.5%
I_{th} average | % change |	8.3%
I_{th} range	147-215.5 pA

7.3. Endnotes

1) Simulations and algorithms were run on a qemu virtual machine running on a physical machine with two X5570 processors (2.9 GHz, 8 threads)

Chapter 8: Optimization with Differential Evolution

8.1. Background

Another optimization strategy involves Differential Evolution, an algorithm developed by Storn and Price, which can search for the global minimum of a function by evolving a population of solution vectors over multiple generations [16]. The algorithm has many attributes, including relatively fast searching time and the ability to deal with non-differentiable or non-linear functions, as well as functions with discrete parameters or noise. Grassia, et al. (2011), used this algorithm to successfully optimize the circuit parameters of a Hodgkin-Huxley conductance based VLSI silicon neuron [17][18]. The researchers used Differential Evolution to independently optimize the circuit parameters of each ionic channel. The fully tuned circuit was then able to replicate the behavior of various classifications of biological neurons. For their circuit, Differential Evolution also outperformed other common optimization algorithms like Simulated Annealing. The successful application of the algorithm to a conductance model suggests that a similar approach could work well with a silicon neuron based on qualitative modeling.

8.2. Differential Evolution Algorithm

The function the Differential Evolution algorithm seeks to minimize is referred to as the *cost function* $f(\mathbf{x})$. The input to the cost function may be a vector of any dimension, but the output must be a scalar value. For consistency with the discussion of the silicon neuron, the elements of the input vector will be referred to as parameters. The steps of the Differential Evolution algorithm used in this research are:

1) Define the bounds of the parameter space.
2) Create a population of solution vectors within the bounds of the parameter space either randomly or in evenly subdivided intervals.
3) Each generation g consists of NP solution vectors $\mathbf{p}^{i,g}$.

$$\mathbf{p}^{0,g}, \mathbf{p}^{1,g}, \mathbf{p}^{2,g} \dots, \mathbf{p}^{NP-2,g}, \mathbf{p}^{NP-1,g} \tag{8.1}$$

4) From this population, chose a *base vector* which yields the lowest value when input into the cost function.

$$\mathbf{b}^{g} = \mathbf{p}^{\text{best},g},\ f\left(\mathbf{p}^{\text{best},g}\right) = \text{minimum value} \tag{8.2}$$

5) Select a *target vector* $\mathbf{p}^{0,g}$.
6) Choose two distinct *random vectors* from the population that are also distinct from the base vector and target vector.

$$\mathbf{x}^{1} = \mathbf{p}^{\text{rand1}},\ \mathbf{x}^{2} = \mathbf{p}^{\text{rand2}},\ \text{rand1} \neq \text{rand2} \neq \text{best} \tag{8.3}$$

7) Add the *weighted difference* of the two random vectors to the base vector to create a *mutant vector* $\mathbf{v}^0$. *M* is called the *mutation constant.*

$$\mathbf{v}^0 = \mathbf{b}^g + M\times(\mathbf{x}^1 - \mathbf{x}^2) \tag{8.4}$$

8) Select elements from the mutant vector and target vector with a defined probability to create a *trial vector* $\mathbf{u}^0$**.** The probability of selection is referred to as the *recombination constant.*

$$\mathbf{u}^0 = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{n-1} \end{bmatrix},\ u_i = \begin{cases} v_i & rand(0,1) \le Cr \\ p_i & otherwise \end{cases} \tag{8.5}$$

9) If the trial vector $\mathbf{u}^0$ performs better than the target vector $\mathbf{p}^{0,g}$ when input into the cost function, it replaces the target vector and proceeds to the next generation. Otherwise, the trial vector is thrown out and the target vector maintains its place in the population.

$$\mathbf{p}^{0,g+1} = \begin{cases} \mathbf{u}^0 & if\ f(\mathbf{u}^0) \le f(\mathbf{p}^{0,g}) \\ \mathbf{p}^{0,g} & otherwise \end{cases} \tag{8.6}$$

10) Using each member of the population as a target vector, create new mutant vectors and trial vectors with the same base vector $\mathbf{b}^g$, and evolve the target vectors to the next generation.

With:

$$\{\mathbf{p}^{1,g}, \mathbf{v}^1, \mathbf{u}^1\}, \{\mathbf{p}^{2,g}, \mathbf{v}^2, \mathbf{u}^2\}, \dots, \{\mathbf{p}^{NP-2,g}, \mathbf{v}^2, \mathbf{u}^2\}, \{\mathbf{p}^{NP-1,g}, \mathbf{v}^2, \mathbf{u}^2\} \tag{8.7}$$

Evolve:

$$\mathbf{p}^{0,g+1}, \mathbf{p}^{1,g+1}, \mathbf{p}^{2,g+1}, \dots, \mathbf{p}^{NP-2,g+1}, \mathbf{p}^{NP-1,g+1} \tag{8.8}$$

11) Repeat steps (3)-(10) for each successive generation until the cost function has achieved an adequate minimum or a set number of iterations is performed.

$$f(\mathbf{p}^i) = \text{global minimum} \tag{8.9}$$

Numerous variations of the differential evolution algorithm exist, the majority of which are similar to the pattern above. One common variation is to change step 4 to select the base vector randomly [16].

8.3. Application to the silicon neuron

The Differential Evolution algorithm was used to optimize the nullclines. Figure 8.1 shows an abstract view of curve optimization with Differential Evolution. Generation after generation, the curve is brought progressively closer to the ideal solution, in this case one of the system nullclines.

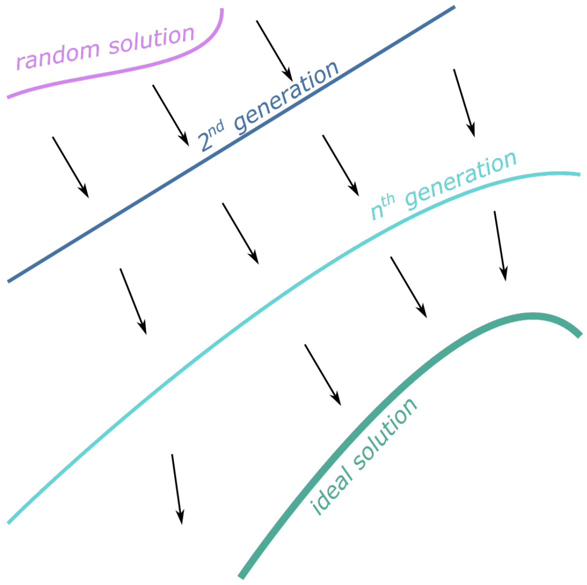


Figure 8.1: Curve matching with an optimization algorithm

Differential Evolution was applied to the silicon neuron by running a Python script which calls the Spectre simulator in conjunction with the *scipy.optimize.differential_evolution()* function from the SciPy library [15]. The cost function constructed takes the circuit parameters as input vector **x**, simulates the silicon neuron in nullcline mode, and calculates the mean absolute errors of the *v* and *n*-nullclines to the benchmark nullclines. These two average error values for the *v* and *n*-nullclines are then combined into a vector whose magnitude is used as the output of the cost function.

$$f(\mathbf{x}) \triangleq \sqrt{\left(\frac{1}{n}\sum_{i=1}^{n}(v_i(\mathbf{x}) - b_i)\right)^2 + \left(\frac{1}{n}\sum_{i=1}^{n}(n_i(\mathbf{x}) - c_i)\right)^2} \tag{8.10}$$

Functions **v**(**x**) and **n**(**x**) represent the nullclines recorded by the Spectre simulator with input parameters **x**, **b** is the benchmark *v*-nullcline, and **c** is the benchmark *n*-nullcline. The *i* indices for v_i, n_i, b_i, and c_i correspond to the individual voltage steps in the DC steady state analysis. Summation maximum *n* should not be confused with *n*-nullcline variable n_i. To limit simulation time and focus on the most significant regions of the phase plane, both the Spectre simulator and the cost function were set to only evaluate a set input voltage range.

For the first application of this algorithm, the 5 circuit parameters used in the limited parameter approach were selected. Since *r(n)* is bypassed in nullcline mode, V_m for this circuit was chosen from the results of brute force optimization in section 7.2. (This led to poor results for 37°C, so *r(n)* for this temperature was tuned by hand). The DC steady-state range was set to 550–700 mV in order to focus on the region where the nullclines cross three times. The population size was set to 40 vectors, 10 times the number of parameters in the cost function. The mutation constant was varied randomly between 0.5 and 1, a process known as dithering. The dithering process and using a population size ten times the size of the input vector are both linked to better simulation results [16][19][20]. The recombination constant was set to 0.7. The parameter bounds were each set to 100 mV ranges with the results of the limited parameter approach roughly in the center.

Table 8.1: Simulations for each run of the Differential Evolution algorithm

Temperature (°C)	Number of Simulations	Time to Completion (hours)
17	4795	2:37
22	8105	4:25
32	5870	3:00
37	5275	3:57

Table 8.2: Parameters optimized with Differential Evolution (rn_Vm comes from brute force for 17-32°C and was tuned by hand for 37°C)

Temperature (°C)	17	22	27	32	37
gv_Vm (mV)	394.3	415.7	432	445.3	456.7
fn_Vb (mV)	253.4	245.2	237	228.7	220.3
Iav_Vin (mV)	433.5	450.4	461	468.3	473.6
Ian_Vin (mV)	418.9	449.4	464.5	474.3	481.2
rn_Vm (mV)	478.2	460.3	445	428.6	407.2
I_{th} (pA)	201	181	178.5	179	189
5 pA response (Hz)	17.3	16.9	15.1	9.2	0
10 pA response (Hz)	38.5	34.5	36.2	38.1	35.1

The algorithm was used 4 times, once for each of the pillar temperatures. Table 8.1 shows the number of simulations run along with the total time to run the algorithm for each temperature. A single complete DC steady state analysis in Spectre took an average of about 2.5 seconds.

Table 8.2 shows the parameters which were returned by the Differential Evolution algorithm. Figure 8.2 compares the nullclines optimized at the pillar temperatures to the benchmark nullclines. The overlapping curves in this figure show clearly that compared with the other methods (Figures 5.9, 6.3, and 7.1), the Differential Evolution algorithm replicated the nullclines more accurately in the measured range.

Figure 8.3 is a plot of the optimized parameters versus temperature and Figure 8.4 shows the transient simulations of the optimized parameters at the pillar temperatures. The 5 pA sustained stimuli for 37°C failed to induce a periodic response. Because of these poor transient results, the interpolation step was not performed. However, the accurately replicated nullclines suggest that modifications to this method could lead to better results.

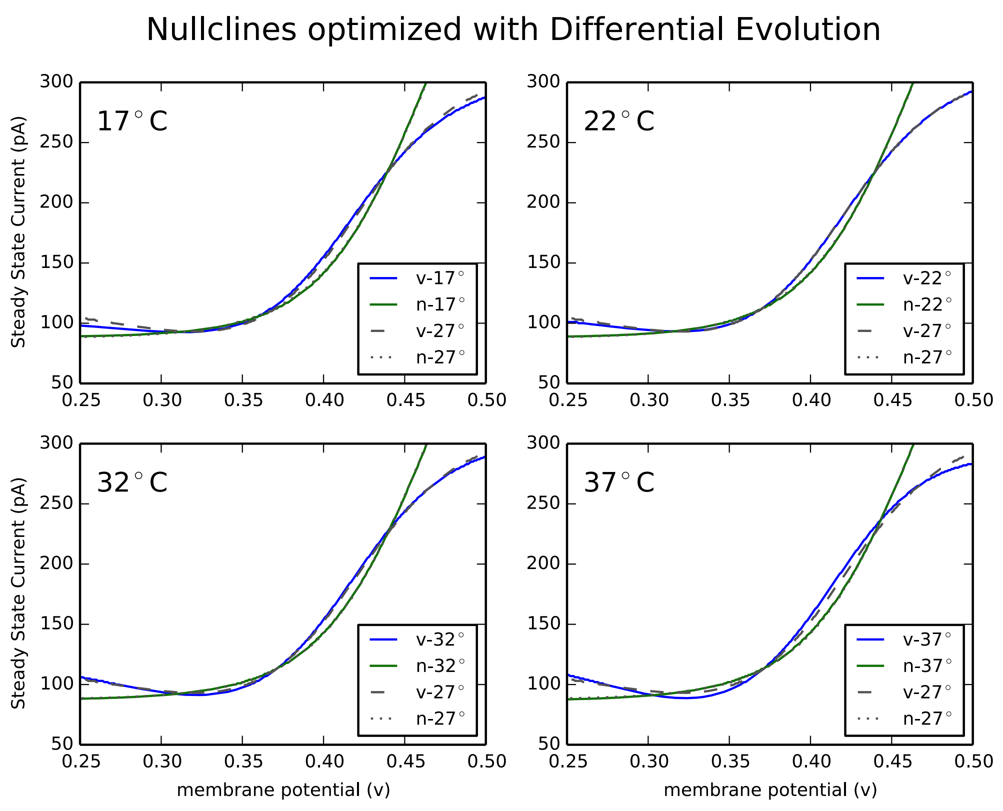


Figure 8.2: Nullclines optimized with Differential Evolution at pillar temperatures compared with benchmark nullclines

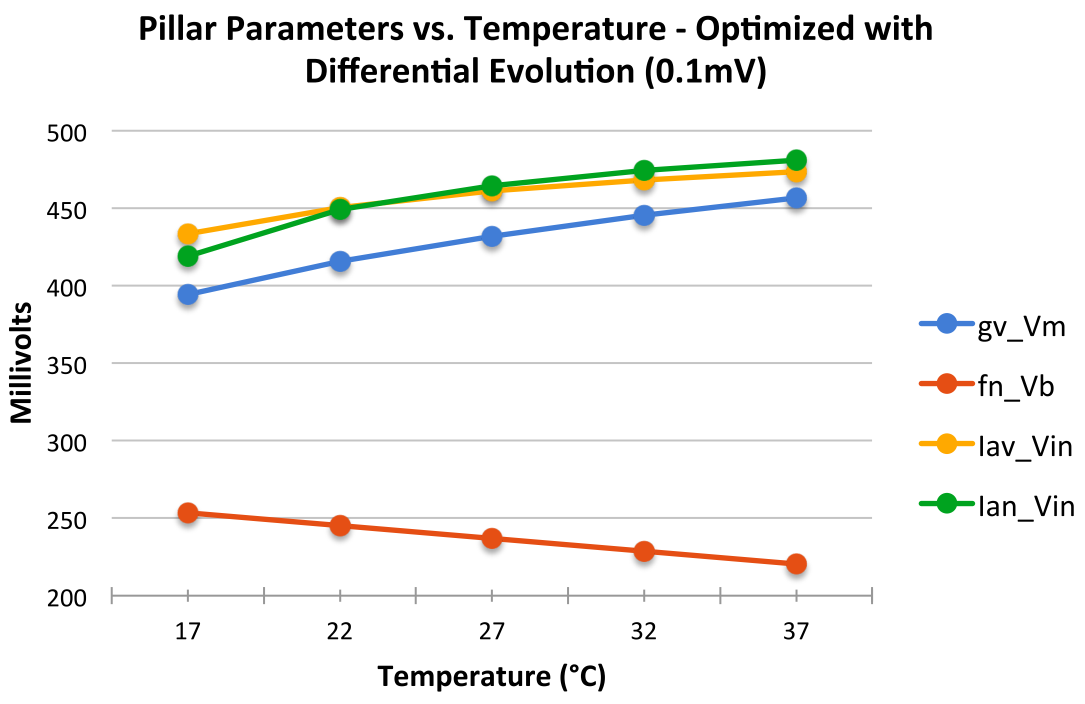


Figure 8.3: Parameters optimized with Differential Evolution versus temperature

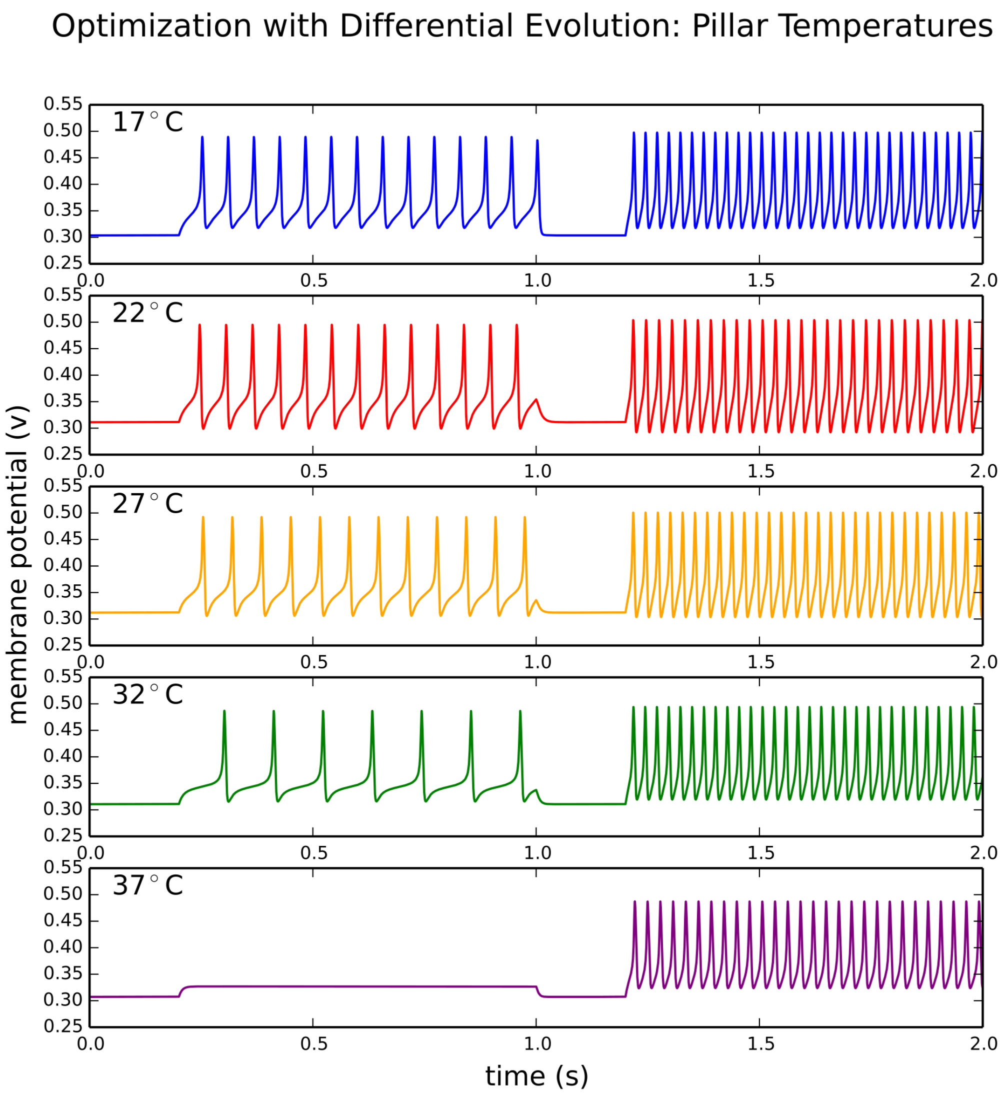


Figure 8.4: Time-series response of the silicon neuron to 5 pA and 10 pA sustained stimuli (0.2-1 and 1.2-2 seconds respectively) at pillar temperatures with nullclines tuned with Differential Evolution. V_m for *r(n)* was selected from the brute force results (section 7.2) for 17, 22, and 32°C, and was adjusted by trial and error for 37°C.

Chapter 9: Conclusion

9.1. Comparison of Methods

Table 9.1 shows the interpolation results from the full parameter approach, the limited parameter approach at two different precisions, and brute force tuning of the limited parameter approach. The pillar parameter sets from the Differential Evolution method did not always induce firing when subject to a 5 pA sustained stimulus and thus were not used for interpolation.

The “average | % change |,” defined in section 5.4 as the average of the absolute values of percent change from the benchmark behavior, serves as a good indicator of the relative strength of the method in question since it shows the overall variance of the behavior from the benchmark.

According to this index, the full parameter approach had the poorest interpolation results, likely due to the unwieldy number of parameters that were difficult to tune consistently with trial and error. Some parameters were tuned more aggressively than others, which lead to irregular progressions of parameters versus temperature. This unevenness may have lessened the effectiveness of the interpolation step which likely benefits from regularity and smoothness. The merit of this method is the high degree of control it provides, suggesting that a modification such as adding an automated optimization step could lead to better results. Differential Evolution or a similar algorithm could be used for such a step.

The limited parameter approach improved the results of the full parameter approach dramatically and was much more straightforward for the user. The progressions of parameters were noticeably smoother. Increasing the interpolation precision from 0.5 mV to 0.1 mV further improved the results of this method.

Brute force optimization improved the results of the limited parameter approach, but was time consuming and suffered from a limited search range. This method worked as a good supplement to the trial and error step, but could not seek a solution outside the immediate vicinity of the parameter space. It is possible that a better parameter set exists beyond the limited range, or perhaps in a distant region of the parameter space. Extending the search radius exponentially increases calculation time, making broad searches difficult. Brute force optimization was better than the limited parameter approach’s 0.5 mV interpolation for all measured behaviors, and for 0.1 mV interpolation, it was better for the 10 pA sustained stimulus response and threshold current, and comparable for the 5 pA sustained stimulus response.

Differential Evolution most accurately replicated the nullclines, but showed poor transient results. While the nullclines are a useful tool for understanding the behavior of the silicon neuron, they only partially describe the system. Additionally, tuning the nullclines ignores the influence of *r(n)*, necessitating a second tuning step. In this case, the parameters from the brute force approach were used with some adjustment.

Even given the poor results, the Differential Evolution algorithm successfully accomplished its input task of matching the nullclines, suggesting that modification of the cost function could improve results. For example, determining the optimal nullclines for each pillar temperature—which may differ from the benchmark nullclines—could be input into the cost function. In other words, the ideal nullclines may change shape at different temperatures.

The issue of precision kept coming up in many aspects of this research. Keeping to 0.5 mV precision proved to be more and more difficult, especially since the the SciPy functions work with machine precision. Increasing the precision was always a tempting way to improve results, but one must recognize that precise bias voltages are difficult to implement on an actual VLSI circuit. Establishing the realistic precision of the optimization strategy is an important future task.

9.2. Future work

The relative merits of each of the approaches suggest that a hybrid strategy with elements of the different methods could lead to better results. One possible approach could be to tune the nullclines with Differential Evolution and then tune the transient behavior with a brute force sweep of the surrounding parameter space.

9.3. Application to a VLSI circuit

Once a good optimization strategy is determined with the Spectre simulator, it can be applied and tested on the actual VLSI silicon neuron in our laboratory. Future generations of the circuit may include programmable on-chip feedback mechanisms that sense temperature and adjust parameters accordingly.

Table 9.1 Analysis of interpolation results for all methods

Circuit Behaviors	Full Param (0.5 mV)	Limited Parameter (0.5 mV)	Limited Parameter (0.1 mV)	Brute Force (0.1 mV)
5 pA stim average frequency	11.9 Hz	15.8 Hz	15.2 Hz	14.8 Hz
5 pA stim average % change	-9.3%	5.2%	0.9%	-7.9%
5 pA stim average | % change |	58.1%	34.7%	18.2%	18.4%
5 pA stim frequency range	2.4-26.1 Hz	3.7-27.1 Hz	6.4-20 Hz	3.4-20.1 Hz
10 pA stim average frequency	35.5 Hz	38.7 Hz	38.2 Hz	37.0 Hz
10 pA stim average % change	-0.6%	6.8%	5.6%	-0.2%
10 pA stim average | % change |	7.8%	7.1%	5.6%	2.2%
10 pA stim frequency range	26.8-40.4 Hz	34.6-43.2 Hz	36.1-40.4 Hz	34.3-38.5 Hz
Average threshold current Ith	185.4 pA	172 pA	176pA	177.5 pA
Average Ith % change	-0.1%	-3.6%	-1.4%	-0.5%
Ith average | % change |	18.5%	12.2%	8.7%	8.3%
Ith range	131-303.5pA	115.5-216 pA	147-213 pA	147-215.5 pA

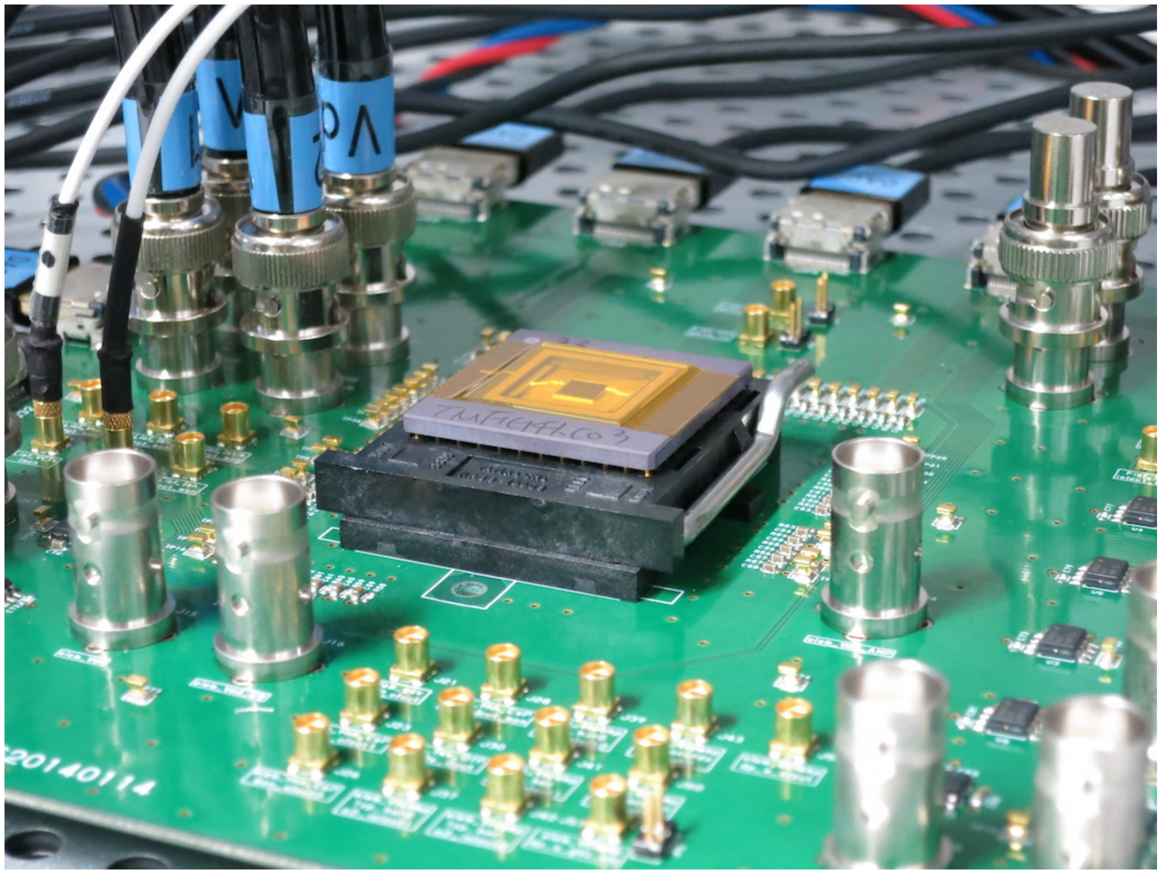


Figure 9.1: Close-up view of the circuit board with the silicon neuron in its bed

Chapter 10: References and Acknowledgements

10.1. References

1) D. Modha, "Introducing a Brain-inspired Computer," http://www.research.ibm.com/articles/brain-chip.shtml, (accessed December, 2015).
2) G. Indiveri, et al., "Neuromorphic silicon neuron circuits," *Frontiers in Neuroscience,* vol. 5, article 73, May 2011.
3) T. Kohno, K. Aihara, "A Qualitative-Modeling-Based Low-Power Silicon Nerve Membrane," Electronics, Circuits, and Systems (ICECS), IEEE, pp. 199-202, December, 2014.
4) Le Masson, et al. "Feedback inhibition controls spike transfer in hybrid thalamic circuits," *Nature*, vol. 417, pp. 854-858, June 2002.
5) Le Masson, et al. "Design of a Bioelectronics Hybrid System in Real Time and in Closed Loop," *Electronics*, vol. 16, no. 2, December 2012.
6) A. Hodgkin, A. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *Journal of Physiology*, vol. 117, pp. 500-544, August 1952.
7) T. Kohno, K. Aihara, "Silicon neuronal networks towards brain-morphic computers," *Non-linear Theory and Its Applications,* IEICE, vol. 5, no. 3, pp. 379-390, 2014.
8) T. Kohno, K. Aihara, "A mathematical-structure-based aVLSI silicon neuron model," *2010 International Symposium on Nonlinear Theory and its Applications,* NOLTA2010, pp. 261-264, September 2010.
9) T. Kohno, K. Aihara, "Improving noise resistance of intrinsic rhythms in a square-wave burster model," *Biosystems*, 112 (2013) 276-283, March 2013.
10) T. Kohno, K. Aihara, "Reducing a fluctuation in burst firing of a square-wave burster silicon neuron model," *Physicon 2011,* September 2011.
11) S. Strogatz, *Nonlinear Dynamics and Chaos*, Westview Press, 2014.
12) J. Rinzel, B. Ermentrout, "Analysis of Neural Excitability and Oscillations," *Methods in Neuronal Modeling*, Massachusetts Institute of Technology, pp. 251-291, 1998.
13) S. Liu, et al., *Analog VLSI: Circuits and Principles*, MIT Press, 2002.
14) M. Miyake, "恒温コントローラを搭載した アナログ VLSI シリコンニューロンの 単独動作システム," University of Tokyo, Kohno Laboratory, February 2015.
15) SciPy.org, "Optimization and rootfinding (scipy.optimize)," http://docs.scipy.org/doc/scipy/reference/optimize.html, (accessed December, 2015).
16) K. Price, R. Storn, *Differential Evolution: A Practical Approach to Global Optimization*, Springer, 2005.
17) F. Grassia, et al., "Tunable neuromimetic integrated system for emulating cortical neuron models," *Frontiers in Neuroscience,* vol. 5, article 134, December 2011.
18) L. Buhry, et al., "Automated Parameter Estimation of the Hodgkin-Huxley Model Using the Differential Evolution Algorithm: Application to Neuromimetic Analog Integrated Circuits," *Neuronal Computation*, 23, 2599-2625, Massachusetts Institute of Technology, 2011.
19) A. Qing, *Differential Evolution: Fundamentals and Applications in Electrical Engineering*, IEEE Press, 2009.
20) G. Onwubolu, D. Davendra, *Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization,* Springer, 2009.
21) R. Byrne, "Dynamical Properties of Excitable Membranes," *From Molecules to Networks,* pp. 181-216, Academic Press, January 2009.
22) A. Wang, et al., *Sub-threshold Design for Ultra Low-Power Systems*, Springer, 2006.

10.2. Publications

1) E. Green, T. Kohno, "Compensating Temperature-Dependent Characteristics of a Subthreshold-MOSFET Analog Silicon Neuron," *The International Conference on Artificial Life and Robotics*, 2016.

10.3. Acknowledgements

I would like to thank Professor Kohno for inviting me to his laboratory and spending many hours with me sharing his unending expertise in electronics, mathematics, and neuroscience. I also would like to express my heartfelt gratitude to my fellow lab members Mariko Miyake, Shusuke Asami, Takuya Nanami, and Taeoh Kwon for their help with homework and research, and making my experience at the University of Tokyo all the more educational and fun.

This study was supported by JST PRESTO and VLSI Design and Education Center (VDEC) at the University of Tokyo with collaboration from Cadence Corporation.