Accurate Mathematical Neuron Models

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Óscar Miranda Domínguez

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Theoden I. Netoff

August, 2012

Acknowledgements

There are many people that have earned my gratitude for their contribution to my time in graduate school.

First, I like to express my deepest gratitude to my advisor, Dr. Theoden Netoff (Tay). He is all about passion. His life is an example and inspiration. We spent a lot of hours working together cranking out data. He also forced me to see the data and results from different angles, and motivated me to develop novel conclusions from my projects. Not only Tay is a great academic, he is also a great friend. He cooked for my family and I following the birth of my second daughter, he invited us to share in thanksgiving and he helped me out after undergoing a medical procedure. This is exactly who Tay is. He takes care of those around him. But, I know that the main thing that I will remember from Tay is that he sparks the joy of living.

I also like to say thank you to my committee member: Drs. Matt Johnson, Hugh Lim, Duane Nykamp and Tryphon T. Georgiou. Thanks for their valuable comments and critical reading of my dissertation.

I have learned a lot from Tyler Stigen. I enjoy his humor sense and also appreciate his patience. I learned to patch, to prepare slices and the fundamentals of linux from him. I also like to say thanks to Jon Gonia. I learned rtxi from him.

I appreciate the companion and constructive criticism received from Bryce Beverlin II, John Ferguson, Vivek Nagarak, Ali Nabi and Abbey Holt.

Dr. Steve Schiff, at Penn State, was a strong motivation and source of knowledge. I learned the fundamentals of linear and non linear Kalman filters from his course: Neural Control Engineering and also from people at his research group: Ghanim Ullah and Yina Wei.

A lot of people from the Tecnológico de Monterrey supported and encouraged me to pursue my PhD (listed alphabetically): Mario Álvarez, Ángel Cid, Jorge Cuevas, Maricela Flores, Lucio Florez, María Teresa González-Garza, Martín Hernández, Sergio Omar Martínez, Gabriela Ruiz, Sergio Serna, Hugo Terashima, and Jorge Valdez, among others.

My gratitude to Tecnológico de Monterrey for its sponsorship and funding since the year 2000, to the University of Minnesota, the Graduate School and the Department of Biomedical Engineering for the Teaching/Research Assistanships received during my PhD training. I also appreciate the funding from the Fellowship of Design of Medical Devices of the University of Minnesota, a Grant-in aid from the University of Minnesota and Dr. Theoden Netoff's 2010-2015 NSF Career Award.

Finally, my eternal gratitude to my whole family, particularly to my wife, Lucy; my daughters Arantxa and Citlali; my father, Epifanio; my mother, Zenaida; my brother, Édgar and Lucy's father and mother, Hermilo and Irma.

Dedication

To my wife, Lucy.

Lucy, agradezco el haber sido bendecido con tu compañía y espero que aún sea largo el camino que recorramos juntos.

Abstract

Computational models of neurons have provided us with a deeper understanding of the basic principles of neuroscience. They can also provide a platform to develop new therapies with a more directed approach.

Here I propose different methods to generate models that can accurately predict behaviors from neurons. I have developed different models that can describe neuronal activity at different time scales. For a full description of the voltage trace, and sampling rates higher than 1 KHz, an Unscented Kalman Filter (UKF) is used to fit a set of parameters of a given mathematical model. The UKF predicts the voltage of the neuron for the next sampling time and this information is used to control, in real time, the voltage of the measured neuron.

For the study of spike rate variability, I introduce the use of fixed and adaptive linear models. These models are able to predict the neurons firing rate in response to stimuli. The models were then used to control the neuron's spike rate.

Finally, I introduce two new methods to measure how the spiking activity of a given neuron can be modified by synaptic inputs. One method is able to describe the change in the period of a periodically firing neuron at different firing rates and under non-stationary conditions. The second method generates a model that is well suited to study neuronal response to bursts of seizure-like activity.

Contents

Acknowledgements **i**

Dedication **iii**

Abstract **iv**

List of Figures **viii**

1 Introduction **1**

2 Modeling **3**

3 Adaptive Noise Cancellation **9**

3.1 Introduction 9

3.1.1 Motivation 12

3.2 Filtering strategies 13

3.2.1 Linear Filters 13

3.2.2 Second order notch filters 15

3.2.3 Comb filters 17

3.2.4 Adaptive filters 18

3.2.5 Selecting frequencies and bandwidth 22

3.2.6 Filtering the data 24

3.3 Discussion 25

4 Spike Rate Control **29**

4.1 Introduction 30

4.2 Results 33
4.2.1 System identification 33
4.2.2 Proportional integral parameters calculation 38
4.2.3 Auto-tuning of a PI controller 42
4.2.4 Phase response curves 42
4.3 Discussion 44

5 Data Assimilation using Unscented Kalman Filters 50
5.1 Introduction 51
5.2 Methods 53
5.2.1 Mathematical model 53
5.2.2 Unscented Kalman Filter formulation 54
5.3 Results 62
5.3.1 Data assimilation 62
5.3.2 Model Reference Control 65
5.4 Discussion 68

6 Optimal Stimulus Design 71
6.1 Introduction 72
6.2 Methods 75
6.2.1 GACell 75
6.2.2 Calculating the percentage of state space covered 76
6.2.3 Stimulus waveforms 77
6.2.4 Optimization algorithm 79
6.2.5 Unscented Kalman Filter implementation 79
6.3 Results 80
6.3.1 State space coverage by step response 80
6.3.2 Optimization of stimulus waveforms 81
6.3.3 Accuracy of parameter estimation dependent on stimulus waveform used 83
6.4 Discussion 87
6.5 Conclusion 91

7 New methods to estimate Phase Response Curves 92
7.1 Method 1: Parameterized PRC 94
7.1.1 Experiment 95
7.1.2 Predicting the neuron's ISI 97
7.1.3 Calculating the neuron's parameterized PRC 100
7.1.4 Discussion 104
7.2 Method 2: Infinitesimal Transient PRC 106
7.2.1 Experiment 108
7.2.2 Analysis 108
7.2.3 Discussion 117
7.3 Conclusion 119

References 121

Appendix A. Experimental preparation 131

Appendix B. GACell 132
B.1 Current balance equation 133
B.2 Intrinsic currents 133
B.2.1 Sodium current I_{Na} 133
B.2.2 Sodium current I_{NaP} 134
B.2.3 Delayed rectifier potassium current I_{Kdr} 134
B.2.4 A-type potassium current I_{KA} 134
B.2.5 Slow potassium current I_{Kslow} 135
B.2.6 Leak current I_L 135

Appendix C. Linear and Unscented Kalman Filter formulation 136
C.1 The discrete-time Kalman filter 136
C.2 The Unscented Kalman Filter 138

Appendix D. Truncated singular value decomposition method 142

List of Figures

2.1 Shapes and sizes in neurons. 4
2.2 Intrinsic firing patterns of neocortical neurons. 5
3.1 Intracellular current recording exhibiting seizure like activity. 13
3.2 Power Spectral Density of an intracellular current recording exhibiting seizure like activity. 14
3.3 Frequency response of three 60 Hz digital notch filters. 16
3.4 Multifrequency notch filters. 17
3.5 Comb filter. 18
3.6 Block diagram of an Adaptive Noise Cancellation filter. 19
3.7 Selecting the main noise sources. 23
3.8 Power Spectral Density of a signal before and after being filtered. 24
3.9 Intracellular current recording and its filtered versions. 25
3.10 Adaptive filter's phase change. 26
4.1 Current-step response of a neuron. 34
4.2 Calculation of the ISI from a voltage trace. 35
4.3 Alternating current injection in the vicinity of the target ISI. 37
4.4 Closed-loop implementation to control the ISI. 38
4.5 Neuronal response to change in the target ISI rate using open-loop and closed-loop controllers. 41
4.6 Running of the auto-tuning program in a cortical neuron. 43
4.7 Closed-loop spike rate control improves the quality of estimated PRCs. . 45
4.8 Sensitivity in the calculation of τ, a, K_p and the poles. 47
5.1 Illustration of *a priori* and *a posteriori* estimations from Kalman filter. . 55
5.2 Predicted values, errors and gains of Kalman filter over time. 56

5.3 *GACell* response to a random current. 63
5.4 UKF-based data assimilation in a neuron. 64
5.5 Error of UKF estimation of neuron's voltage. 65
5.6 Open-loop behavior of neuron to be controlled. 66
5.7 Model reference control of an irregularly firing neuron to spike regularly. 67
5.8 Model reference control of an irregularly firing neuron to fire in a controlled irregular pattern. 68
6.1 GACell in current clamp, constant current. 76
6.2 State spaces of the GACell, current clamp, constant current. 81
6.3 GACell in current clamp, optimized chirp stimulus. 82
6.4 GACell in voltage clamp, optimized OU stimulus. 83
6.5 State space representation of GACell response to stimulus inputs. 84
6.6 Noise added to stimulus to simulate realistic variability in neuronal response. 85
6.7 Parameter estimation of GACell in response to current step. 86
6.8 Parameter estimation of GACell in response to chirp-like stimulus. . . . 87
6.9 Parameter estimation of GACell in response to optimized Ornstein-Uhlenbeck process used in voltage clamp. 88
6.10 Summary results of parameter estimation with different stimulus waveforms. 89
6.11 Percentage of state space covered as a function of time. 90
7.1 Experimental design to estimate Parameterized PRCs. 96
7.2 Recursive ISI estimation in a neuron model. 99
7.3 Recursive ISI estimation in a pyramidal neuron. 99
7.4 Parameterized PRC in a neuron model. 102
7.5 Parameterized PRC in a pyramidal neuron. 103
7.6 Parameterized PRC in 60 pyramidal neurons: summary results. 104
7.7 Current-step response of a neuron. 109
7.8 Current-step response of a neuron. 110
7.9 Infinitesimal transient PRC estimation in a computational neuron model. 113
7.10 Infinitesimal transient PRC estimation in a cortical neuron. 114
7.11 Out-of-sample comparison between predicted and measured STA. 116

7.12 Infinitesimal transient PRC in 30 cortical neurons: summary results. . . . 117
D.1 Linear transformation. 143

Chapter 1

Introduction

Modeling of neurons has increased our understanding on the computing properties of the nervous system. Models are used to test hypothesis of how it works, identifying where we have gaps in our understanding and providing a platform to design drugs and stimulation parameters for electrotherapy. Accuracy of models depends on their ability to capture the underlying biology.

Modeling the nervous system requires nonlinear models. Fitting data to non-linear systems is difficult and requires complex experiments. Existing neuron models have been obtained under particular experimental conditions, and then the results are generalized under un-tested conditions. For example, in different animal models, at different ages of the animal or when a particular drug is added to the neuron. Having a method to estimate the parameters for each particular neuron would revolutionize the state of the art in terms of neuron modeling. Here, I present different methods to generate models that can accurately predict behaviors from neurons at different time scales.

The experimental data was obtained from neurons in the hippocampal formation from rat brain slices (see appendix A) using a technique called “whole cell patch clamp”. The measurements and closed loop control experiments were performed using a linux based dynamic clamp system called “Real Time eXperimental Interface” (RTXI). The 60 Hz noise and higher frequency coherent oscillations introduced in the measurements were removed by using adaptive filters.

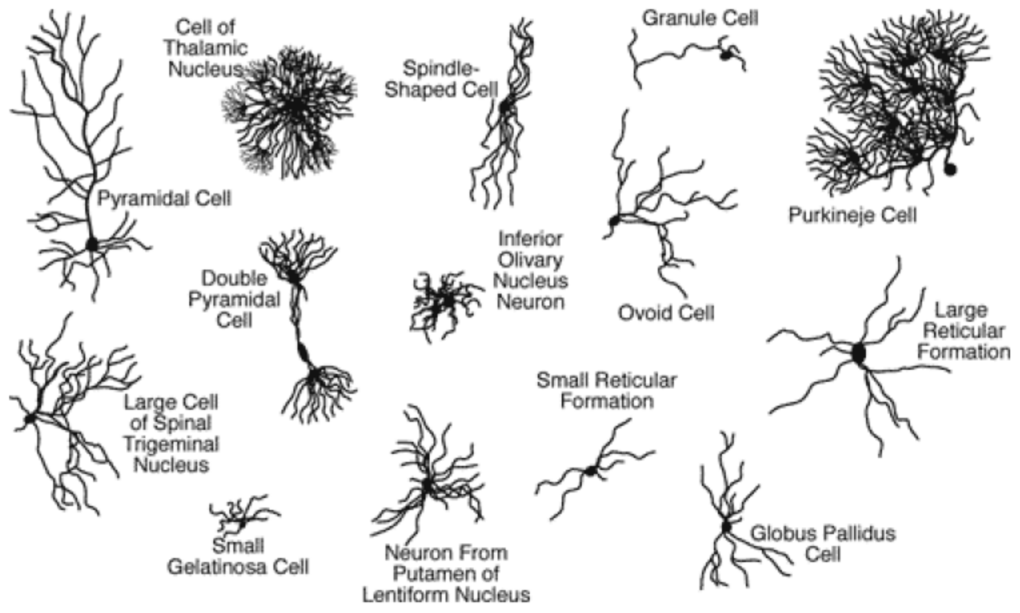
The material is presented as follows: Chapter 2 briefly describes the features that are modeled in neurons. Then, chapter 3 cover the topics of removing noise in neural recordings. The remaining chapters represent the main contributions of this thesis to the field of modeling: In chapter 4, I apply closed loop control to adjust the applied current to a neuron to keep its interspike interval (ISI) constant. This is relevant because some electrophysiology experiments require periodically firing neurons but even regular spiking neurons have variability in their ISIs. Here, the neuron's dynamics are modeled by using a first order difference equation. The time scale of this modeling approach is on the order of hundreds of milliseconds. The obtained model is used to tune a controller to keep the neuron's firing rate constant. Chapter 5 presents the implementation of a more sophisticated model estimation for control, the unscented kalman filter (UKF). The UKF is implemented real-time and is used to estimate the neuron's voltage and the hidden states. It shows the feasibility of implementing UKF in real time for electrophysiology experiments. The time scale of this modeling approach is on the order of hundreds of microseconds and it enables the reconstruction of the measured voltage trace. In this chapter, the UKF is used to predict the neuron's voltage which is then used to calculate the required applied current to make the neuron follow a pre-determined voltage trace. UKF-based estimations can be improved by different factors, like having a good set of initial conditions but also by using an appropriate stimulus to generate a data set enabeling accurate parameter estimation. Chapter 6 presents a theoretical approach to determine the optimal stimulus to maximize accuracy of parameter estimation. This approach is validated by estimating parameters in neuron models using UKF. Finally, chapter 7 introduces two new methods to estimate the firing rate response of neurons to synaptic inputs under transient conditions. The first method measures how the neuron responds to a single perturbation as the neuron's firing rate varies. The second method asks how the neuron responds to a volley of inputs (represented by a noisy input) under transient conditions, as may be observed at the onset of a seizure.

Chapter 2

Modeling

Mammalian neurons exhibit a great diversity in terms of morphology (see figure 2.1), function and electrical activity (see figure 2.2). In terms of morphology, the neurons can be monopolar, bipolar or multipolar [1]. In terms of function, the neurons can be excitatory or inhibitory. In terms of voltage pattern, neurons can be classified as regular spikers (RS), fast spikers (FS) or intrinsic bursters (IB) [2]. There are also cells that fire irregularly. Sometimes it is possible to associate morphology with function. For example basket, Chandelier, medium-sized spiny, purkinje, and spiny stellate cells usually are inhibitory and fast spiking. While pyramidal cells are usually excitatory and regular spikers.

Morphological identification of neurons has been used for over a century [3]. Immunolocalization and, recently, opto-genetics have allowed us to identify cell function based on the presence of proteins involved in neurotransmitter synthesis (anti-Glutamic acid Decarboxylase 67 is an antibody to identify inhibitory cells [4] and pyramidal cells have been identified using transgenic mice expressing Yellow Fluorescent Protein, YFP, [5]). However, there is a gap in our ability to identify neurons by their dynamics. The motivation for my thesis is that modeling and machine learning may provide tools to relate neuron dynamics to cell types. Furthermore, models can also provide a platform to develop new therapies using a more directed approach. This can be useful in developing, for example, new antiepileptic drugs and in tuning deep brain stimulation parameters. However the prediction accuracy of any model depends on how faithfully it can reproduce the behavior of the system. This thesis will predominantly be about how to fit parameters of models to experimental data.

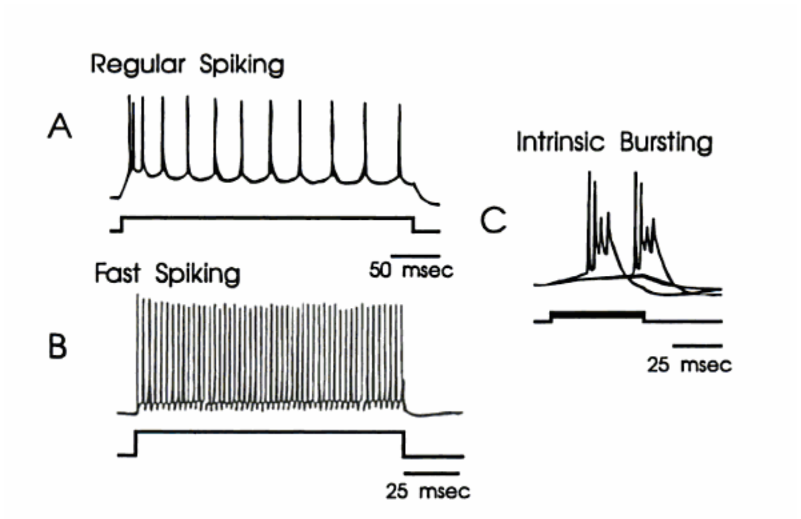

Cell of Thalamic Nucleus
Granule Cell
Spindle-Shaped Cell
Pyramidal Cell
Purkineje Cell
Inferior Olivary Nucleus Neuron
Double Pyramidal Cell
Ovoid Cell
Large Reticular Formation
Small Reticular Formation
Large Cell of Spinal Trigeminal Nucleus
Small Gelatinosa Cell
Neuron From Putamen of Lentiform Nucleus
Globus Pallidus Cell

This diagram illustartes various shapes and sizes of neurons. (It is based on drawings made by Cajal.) [6]

Figure 2.1: Shapes and sizes in neurons.

Today, a neuroscientist has many different mathematical models to describe the neuronal behavior. These models can be used to understand the electrical activity of neurons and their relationship to a stimulus, or pathological activity. However, even a detailed model cannot precisely predict the voltage trace for any particular neuron due to the stochastic nature of experimental conditions. The goal of modeling therefore is to generate a model that can reproduce the statistical properties of the neuron, such as firing rate, resistance, bursting activity or frequency-current relationship, among other features.

Mathematical models can be of different spatial and temporal scales, depending on the question of interest. For example, if the modeler is interested in large scale simulations, (s)he can use reduced models [7], [8] or phase response models [9, 10]. If the focus of the study is related to the absence or presence of a particular ionic current, then a Hodgkin-Huxley-like model is preferred [11]. The trade-off is between biological



A. Regular spiking neurons show initial high frequency spiking and rapid adaptation to a sustained lower frequency. **B.** Fast spiking neurons fire at a high sustained frequency throughout the stimulus. **C.** Intrinsic bursting neurons respond to stimulus at threshold with an all-or-none burst [2]

Figure 2.2: Intrinsic firing patterns of neocortical neurons.

plausibility and computational effort [12].

Neuron dynamics can be modeled by reproducing its voltage trace or the interspike intervals. We can use linear models, or nonlinear models. We can use static or adaptive models. For the voltage trace, on very short time periods, an adaptive linear model can be very accurate and can be used for removing periodic noise, and will be presented in chapter 3. For controlling neuron spike times, a simple linear model of current to spiking rate can be used, as presented in chapter 4. For more accurate control, a higher order model that takes into account the history of currents and spikes can be used, as presented in chapter 7. Nonlinear models can be made to describe how the ISI is affected by multiple stimulus inputs, also presented in chatper 7. Finally, I will also present how nonlinear models of a neuron's voltage, given an arbitrary input, can be used to predict the voltage and control the cell's voltage trace, as presented in chapter 5.

For the study of ionic currents, Hodgkin-Huxley-like models provide models that explain the voltage trace of the neuron. Their original model was used to explain the voltage response of a squid giant axon to electric impulses [11]. Their model broke down the complicated voltage trace into constituent and independent membrane currents. While mammalian neurons exhibit richer dynamics than squid neurons, such as adaptation (changing of frequency), differences in the shape of the voltage and extended spiking, it is possible to extend this model by adding ionic currents found in these neurons responsible for these behaviors.

The conventional approach to generating a model of an ionic current is to first isolate the current. There are 4 methods to isolate currents from patch clamp recordings: 1) *Kinetically*, 2) *Current subtraction via stimulus protocol*, 3) *Isolation solutions*, and 4) by *By pharmacological isolation* [13]. Given the data from the isolated current, a model is fit to describe how the conductance changes as a function of time and voltage.

In cortical neurons [14, 15], there are voltage dependent, calcium dependent and other types of currents that modulate the flow of the main ionic species involved in the electrical activity of neurons (Na^{+}, K^{+}, Ca^{++}, and Cl^{-}). Each ionic current is responsible for a particular behavior. Regular spiking cells are characterized by a transient sodium current I_{Na}, a *Hyper polarizing Inward Rectification*, I_H, *Sub-threshold Depolarization, T Ca^{2+} channels* $I_{Ca(T)}$ and Na^{+} *persistent* , I_{NaP}- [2], currents.

Many Hodgkin-Huxley style models of neurons are now available through a public repository called "ModelDB" [16]. For the simulations presented in this thesis, I chose a single compartment model proposed by Golomb and Amatai [17] (*GACell*). The GACell has several different ionic currents. It accounts for two Na^{+} currents (fast voltage gated and persistent), three K^{+} currents (delayed rectifier, A type current, and slow current), as well as a leakage current. I_{Na} and I_{Kdr} are responsible for the action potential generation [17], the I_{NaP} increases the excitability of the cell at voltages around threshold, which makes the cell fire action potentials at low input currents, as observed in mammalian neurons [18]. I_{KA} reduces the gain of the neurons (firing rate *per* applied current), shaping the slope of the frequency *vs* current curve [18]. The frequency adaptation observed in excitatory cells is modeled with the I_{Kslow} [17]. Finally, the (I_{leak}) is a non-voltage dependent current that determines the passive properties of the cell [18] balancing the other currents set the model's resting potential. The model has 5 states, the voltage and four gating variables which determine the ionic currents.

For parameter estimation, especially when the models are highly nonlinear, many approaches can be used to fit mathematical models to experimental data [19]: hand-tuning, brute-force, gradient descent, simulated annealing and synchronization methods. These methods will be described briefly here, but this thesis will present a new approach, used extensively in engineering fields but not yet widely applied to neuroscience: the Unscented Kalman Filter.

The first approach is *hand-tuning* (example [20]), where the modeler uses his(her) knowledge to fit the model's parameters.

In the *brute-force* approach, the modeler runs a mathematical model with millions of different combination of parameter values and chooses the model that best reproduces the neuron's behavior [21, 22].

In the *gradient descent* method, an incremental approach is used. An initial set of parameter values are selected, the model is integrated and evaluated using a cost function that weights the error between the model and the data. Measures used in the cost function can include attributes like the firing rate, the frequency-current curve, the adaptation time constant, the interspike intervals, and the number of spikes in bursts. An algorithm is then used to perturb the parameter values in the direction that minimize the cost. The process is repeated until the cost reaches a minimum [23]

A problem with gradient descent methods is that they can become stuck in a local minimum, missing the global minimum model. An alternative is to use a *simulated annealing* approach. In this case, the model parameters are perturbed randomly, and perturbation to the model will be accepted if the cost decreases, but also if it increases a small amount. The acceptable amount of increased cost is decreased over the iterations. This is akin to heating the model by accepting some randomization and "annealing" refers to to the decreased threshold over time. This approach, while taking longer than the traditional gradient descent is more robust to finding the global minimum. This method have been used to fit models that reproduce some qualitative dynamics observed in neurons [24].

Another approach is to make a model that can synchronize with the data using minimum coupling. Synchronization of chaotic systems [25] has been applied to parameter estimation in neurons [26, 27]. In this approach, it is assumed that the real dynamics of a neuron is described by a set of differential equations: $\dot{\vec{x}} = \vec{f}\,(\vec{x}, \vec{p})$. Then, a model with a similar structure is proposed to describe the neuron's dynamics: $\dot{\vec{y}} = \vec{g}\,(\vec{y}, \vec{q})$, where $\overrightarrow{x}$ and $\overrightarrow{y}$ are the systems' states and $\overrightarrow{p}$ and $\overrightarrow{q}$ are the systems' parameters. The goal of this approach is to build a cost function such that makes $\overrightarrow{y} \rightarrow \overrightarrow{x}$ as $t \rightarrow \infty$, if $\overrightarrow{q} = \overrightarrow{p}$. This approach is highly dependent on the cost function selection and the sampling rate.

However, there are many difficulties in generating accurate models. Two problems of major concern are 1) there may not be a unique set of ion channels that can produce the same behavior, and 2) neurons are not static but modulate their behavior over time according to environment and internal demands [28] changing the expression of ion channels [29]. It is not clear how to address the uniqueness problem, other than neurons of a certain cell type may not use a single set of ion channel densities to produce a single behavior. Experiments have shown that the density of channels vary dramatically within cell types, but that the different channels are correlated, indicating that the solution may be a manifold of parameters and not a single unique solution [30, 31]. Therefore our goal in modeling may not be to find the unique solution but instead to find a solution on the manifold of possible solutions a population of neurons may use. For the problem that ion channel kinetics are modulated over time, one solution is to make adaptive models that estimate the parameters as a function of time. The Unscented Kalman Filter (UKF) is one such way to fit models.

To reiterate, fitting model parameters to data is not easy and straightforward. One model fit to one data set may not generalize to the neuron recorded under other conditions or to other cells under the same conditions [32]. There may not be a unique combination of parameters that produce the same statistical description of the neuron [21, 30]. And, there may not be a one-to-one mapping between ionic currents and cell behaviors [22], making uniqueness one of the big concerns on this task. Developing a tool to make realistic models of neurons is an unsolved problem in neuroscience. By making systematic approaches to fitting models to data, we may improve the accuracy of model predictions, which may in turn lead to the design of better therapies.

Chapter 3

Adaptive Noise Cancellation

Electroencephalography and other neuronal recordings are increasingly being used outside of the clinical setting and are prone to noise such as power line (60 Hz) and other electromagnetic radiation. Notch and comb filters are often used to remove the periodic noise sources. This is effective when there are only a few frequency bands to remove, however when the sampling rate is high, cascading filters can begin to interact producing undesired distortions in phase and amplitude in the pass bands. An effective alternative that is easy to implement in real time is an adaptive filter. Adaptive filters optimize their parameters at each sampling time to effectively remove undesired components. This chapter is a tutorial on adaptive noise cancellation designed for neuroscientists with limited experience in signal processing. I present a comparison in performance of notch, comb and adaptive filters applied to a seizure recording corrupted by several periodic noise sources.

3.1 Introduction

Electroencephalogram (EEG) recordings has long been used for diagnosing epilepsy. This is usually done in a clinical setting with a controlled environment. Specifically, an environment where noise sources can be minimized. More recently, developing devices that can analyze EEG data in real time to detect or predict seizures has become possible. EEG is also used in Brain Computer Interfaces (BCI), where signals recorded from the brain are used to control a device [33, 34, 35]. As EEG moves out of the clinic and into more real-world conditions, noise becomes a considerable problem corrupting the EEG measurements degrading the signal to noise ratio. Undesired signals corrupting EEG can be classified into noise and coherence interference [36]. Noise is any undesired signal that is random in nature, like “white” noise, which is stationary, or thermal noise, which is non-stationary. Coherence interference is periodic noise. Examples of coherence interference are the electro-magnetic fields from the power line supply at 60 (50 in Europe) Hz and its harmonics, HVAC systems, electric pumps, and refresh rates on computer monitors; at radio frequencies there are sources like radio, television and cell phone signals. Coherence interference is picked up by ground loops, and by inductive and capacitive interactions with recording electronics [37].

Because coherent interference signals are periodic, they can easily be seen in the power spectrum as a series of peaks that stand out beyond the broadband EEG signal. They seem like ideal candidates to filter out using Finite Impulse Response (FIR) or Infinite Impulse Response (IIR) filters, but they can be deceptively difficult to remove. If coherence interference signals are stationary and combined additively to the EEG, it would be easy to filter them out using linear filters. However, these signals have frequencies that can drift over time and the signals can interact, such as modulation of a high frequency oscillation by a lower one. This makes it difficult to design a filter that removes these coherent noise signals. Here, I present a method to remove the coherence interference present in measurements using adaptive linear filters. For brevity I will use the term "‘noise”’ to refer to coherence interference in the remainder of this chapter.

The magnitude of the electrical activity of neural processes is in the order of μV for EEG recordings recorded at the scalp and on the order of mV for signals measured intracranially. These signals are usually amplified 100-10000x gain before being digitized and recorded. The amplitude of the signals are the same or even smaller than the noise signals. Shielding the recording electrode and recording system to prevent the undesired signals from entering the recordings is the best filtering strategy. This can be done by using dedicated ground systems, radio-frequency absorbing materials, and Faraday cages. However, even with the best prevention, noise continues to be a problem. Therefore, there are many conditions when an electronic filter may still be necessary.

An ideal filter selectively removes the noise without affecting the signal. If the noise signal can be measured separately it is possible to subtract this signal out. This can be done by placing a reference electrode in close proximity to the recording electrode and amplifying the difference; known as common mode rejection. If noise sources cannot be measured simultaneously and independently along with the corrupted signal, then the noise needs to be estimated from the signal. If the filtering can be done offline, designing a filter in Fourier space to remove the noise components and then inverse Fourier transforming the filtered signal back to the time domain. Similarly, filtering can be done with wavelets [38] or other time-frequency Fourier transform-based tools [39] .

However, when the user needs to filter in real time, either for observation or control, the available tools are more limited. Commonly used linear filters are Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters. While these filters are the workhorse of the signal processing field, they have many disadvantages. Many of these problems have become more pronounced now that sampling rates for EEG signals have increased. Until the early 80's and even early 90's most EEG was recorded using pen and paper. The pens had significant momentum that acted as low pass filters, and it was not possible to record signals much above 100 Hz. In the 90's EEG signals became digitized and this hardware limitation became obsolete. Slowly, scientists have been broadening the recording bands and discovering interesting spectral characteristics in EEG at high frequencies [40, 41, 42, 43], and that EEG signals in these bands can be controlled by a patient for effective BCI [44]. When sampling at less than 200 Hz, the there are only a few harmonics of 50/60 Hz, which can easily be removed by cascading a few notch filters. However, when the sampling rate is increased to 5000 Hz, there are over 40 harmonics, and noise sources with fundamentals higher than 100Hz become relevant.

In this chapter I will provide a brief overview of linear filter design, the problems with linear filters when there are many noise components, and then introduce adaptive filters and demonstrate them on some noise corrupted seizure recordings. There are many excellent textbooks that provide an introduction to linear filter design and I refer the reader to these sources if they are interested in a more in depth description [45, 46, 47, 48].

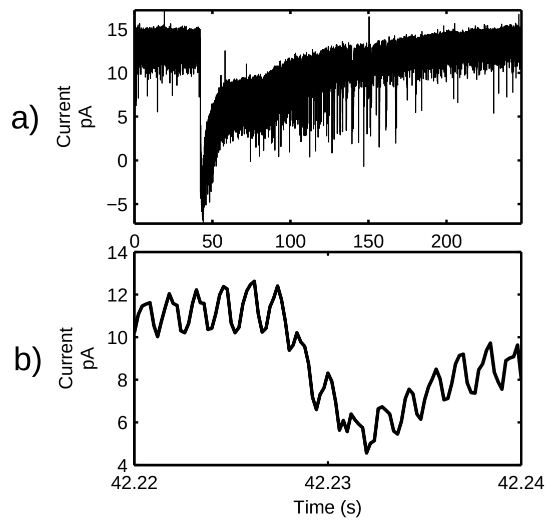
I will discuss the effects, advantages and disadvantages of the use of notch, comb and adaptive filters. The main goal is to show the effectiveness and simplicity of implementing adaptive filters to run in real time. To illustrate the performance of the filter, I use intracellular current recordings that were performed in an animal model of epilepsy (pyramidal cells from rat brain slices in a bath that induces seizure-like activity). This algorithm could be implemented in a closed-loop experimental system such as the open-source Linux-based Real Time eXperiment Interface (**RTXI**) [49, 50].

3.1.1 Motivation

Figure 3.1 shows a measurement of seizure like activity measured from a brain slice. This signal happens to be an intra-cellular recording recorded in voltage clamp to show current flow in the cell, but it is representative of a signal recorded from the brain under noisy conditions. This signal, has low magnitude and it is corrupted by noise. While there is a clear slow neuronal component, the magnification of the data in time (panel **(b)**) shows a high frequency large amplitude component obscuring the low amplitude biological signals. From the power spectrum we can estimate the amplitude of the different noise signals. The corresponding power spectrum of the signal is shown in figure 3.2. By inspection of the power spectrum, we can see that:

1. the components with highest power are the harmonics of 734 Hz,
2. there are side bands to the 734 Hz peak which indicates this frequency is modulated by a lower frequency signal (approximately 10 Hz),
3. 60 Hz its harmonics are present, and higher order harmonics do not exactly lay at a frequency multiple of 60.

Even though the noise sources in this recording are specific to our noise sources, this time series exemplifies the expected and unexpected noise components that are present in every neuro-recording: harmonics that does not lay at an integer multiple of the fundamental, coupled electromagnetic radiation of different sources like monitors and lamps, non-linear interactions like modulations appearing as lateral bands in the vicinity of the frequency of the carrier, etc. Here, we will show how an adaptive filter can effectively remove those components in real time.



(a) shows the entire recording and **(b)** is a close up in the vicinity of 42.23 s. The main oscillation has a frequency of 734 Hz.

Figure 3.1: Intracellular current recording exhibiting seizure like activity.

3.2 Filtering strategies

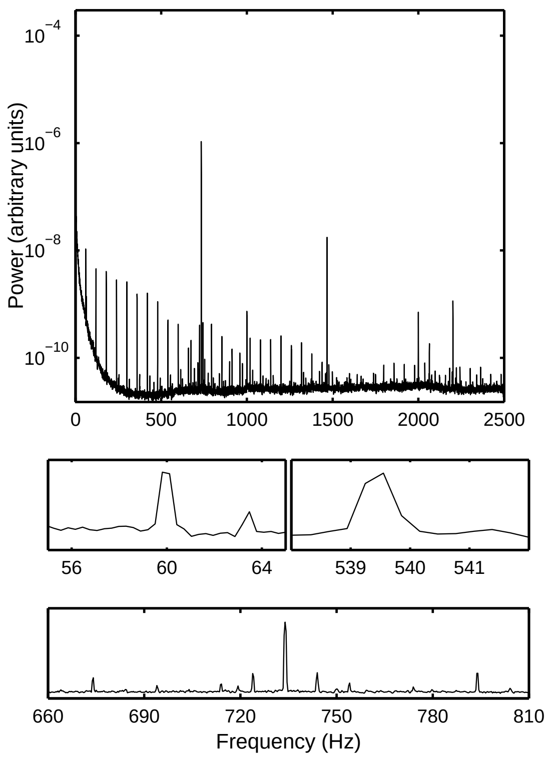
3.2.1 Linear Filters

Linear filters in the time domain iteratively calculate the filtered output as a weighted sum of past values of the original signal u as well as the filtered signal y. These filters take the general form as follows:

$$y_k + a_1 y_{k-1} + ... + a_p y_{k-p} = b_0 u_k + ... + b_q u_{k-q}, \tag{3.1}$$

where k is the time index, $u = u_1, ..., u_n$ is the measured signal, $y = y_1, ..., y_n$ is the filtered output, the coefficients $a_1, ..., a_p$ and $b_0, ..., b_q$ are their respective weighting factors, and q and p indicate the number of points used from the original data and the filtered data. Filter design is an entire field dedicated to determining the a and b coefficients to remove the noise from u to generate a "noiseless" signal y.

The goals of filter design are easier to visualize in the frequency domain. The z-transform is the discrete equivalent of the Laplace Transform. It is used for visualizing



Top panel show the power as a function of frequency of the signal. Middle left panel shows a close up in the vicinity of 60 Hz, middle right panel in the vicinity of 540 Hz and lower panel in the vicinity of 735 Hz.

Figure 3.2: Power Spectral Density of an intracellular current recording exhibiting seizure like activity.

digital filters. The z-transform of the filter 3.1, is:

$$H\left(z\right)=\frac{b_0+b_1z^{-1}+...+b_qz^{-q}}{1+a_1z^{-1}+...+a_pz^{-p}}, \tag{3.2}$$

where z is the complex frequency variable that depends on the sampling frequency, and $H\left(z\right)$ is called the "transfer function" of the filter, characterizes the relationship between the input and output of the filter.

From the function H you can calculate the effect of the filter on the amplitude and phase of the filtered signal as a function of frequency. The numerator and denominator are polynomials, the solutions to these polynomials, where the equation equals zero, are called ”zeros” for the solution to the numerator and ”poles” for the solutions to the denominator. The placement of the zeros and poles determines the stability of the filter (we usually don't want a filter that can go off to infinity with a bounded input) as well as the frequencies that are removed and amplified. Ideally the gain of the filter is 1 at all the frequencies where there is little noise (band-pass) and almost zero at the frequencies corrupted by noise (band-stop). The zeros determine what frequencies the filter will not pass (nulls), while the poles indicate frequencies that the filter will amplify. By placing zeros near frequencies that we desire to remove and poles flanking them to restore the signal back to passing frequencies in nearby bands containing signals we want, we are able to design the filter. Switching from band-stop to band-pass or vice-versa is never a perfect step. The more points used in the history (p's and q's, the more accurate these transitions will be. The trade-off is that the filter becomes more computationally intensive. There will always be some error in the amplitudes caused by the switch between these two amplitudes. Other problem with these filters is that the frequency and bandwidth of the undesired signal can change over time, but H is static, *i. e.* the position of their nulls does not follow the frequency components of the undesired signal. A common strategy to overcome this problem is to design nulls with a wider bandwidth. However, this approach does not discriminate between desired signal and noise, hence, desired components of the signal are removed.

3.2.2 Second order notch filters

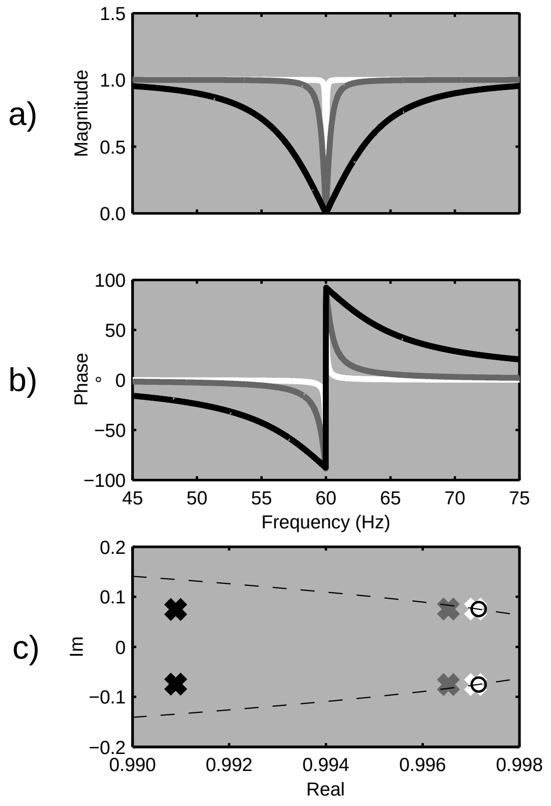
The simplest linear filter is a second order notch filter where the maximum order of the polynomials on $H(z)$ is two. For this filter it is easy to set the notch at a specific frequency. The transfer function of a second order notch filter is given by

$$H(z) = \frac{1 - 2cos(2\pi f_c)z^{-1} + z^{-2}}{1 - 2\zeta cos(2\pi f_c)z^{-1} + \zeta^2 z^{-2}}, \tag{3.3}$$

where f_c is the frequency to be removed, in Hz, and ζ controls the bandwidth. ζ must be between 0 and 1 to have stable filters. Setting $\zeta = 0$ (denominator equal to one) results in a notch filter that has a wide bandwidth, while using $\zeta = 1$ does not affect the amplitude of the applied signal.

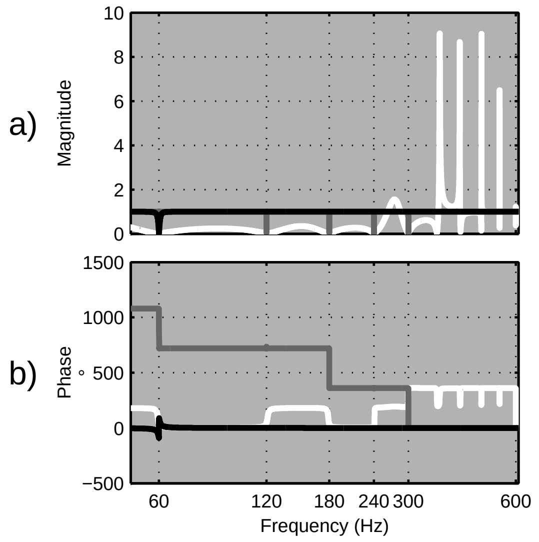
Examples of three second order notch filters all designed to remove 60 Hz at a sampling frequency of 5000 Hz, but different bandwidths are shown in figure 3.3. Notice that the filter has a discontinuity in phase at the center frequency of the notch and that the wider bandwidth the smoother the transition. We can see in panel C that the distance between poles and zeros determine the bandwidth.

A second order notch filter is a good solution when the coherence interference is stationary and its frequency and bandwidth are known. When there are higher order harmonics, a cascade of these filters can be implemented to remove the harmonics. However, the poles add ripple in the bandpass, and a cascade of notch filters can lead to undesired frequency responses, as shown in figure 3.4.



Magnitude **(a)** and phase **(b)** response of three 60 Hz notch filters with bandwidths of 0.1 (white), 1.0 (gray) and 10.0 Hz (black). Panel **(c)** shows the corresponding poles ('x') and the three common zeros ('0').

Figure 3.3: Frequency response of three 60 Hz digital notch filters.



Magnitude **(a)** and phase **(b)** response of three notch filters with bandwidth of 0.001 Hz and cutting frequencies of 60 Hz (black), 60 Hz and 4 harmonics (gray) and 60 Hz and 9 harmonics (white).

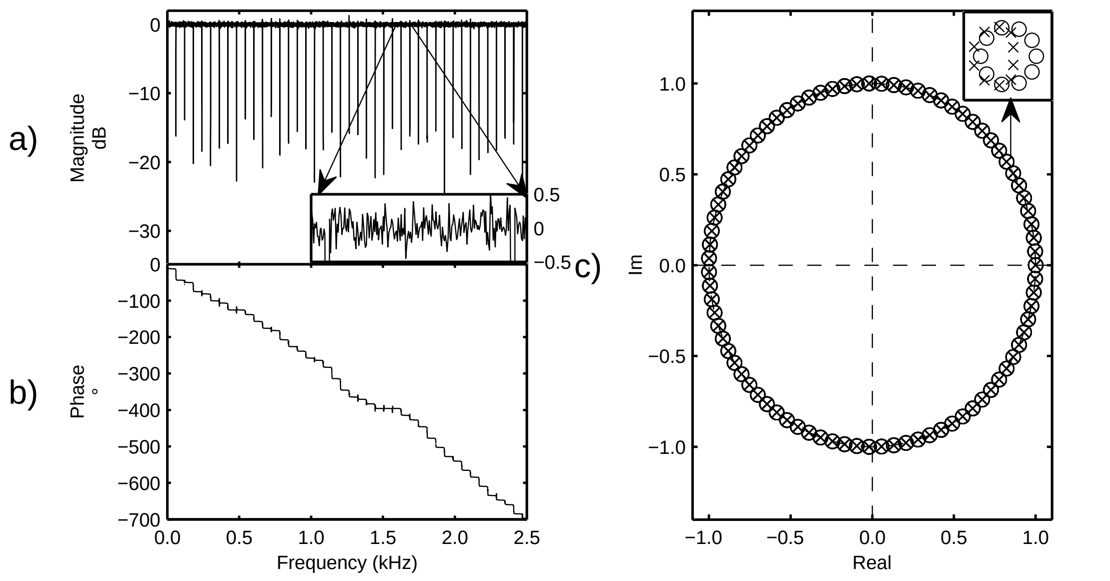
Figure 3.4: Multifrequency notch filters.

3.2.3 Comb filters

If the noise is limited to one fundamental and peaks repeatedly at its harmonics (frequencies that are integer multiples of the fundamental), it is possible to design a notch filter that is periodic in frequency to remove these noise sources [46]. It evenly distributes the zeros in the transfer function across the spectrum to the Nyquist frequency nf_c, where $n = 1...\lfloor Ny/f_c \rfloor$. This filter may be very high order, but many of the coefficients in the transfer function are zero and the effective number of zeros and poles is manageable for real time applications. The advantage of a comb filter is that the interactions between all the zeros and poles are accounted for in the design of the model, so it behaves better than applying a series of individually designed filters.

The magnitude and phase response of a comb filter is shown in 3.5. Notice that comb filters can be designed with narrow bandpass. The phase delay of the filter “jumps” at at each notch. While the gain in the bandpass is close to one, at closer inspection it can be seen that it is not perfect.

The disadvantage of the comb filter is that the notches must be evenly distributed across the spectrum, therefore it only works at frequencies where the sampling rate divided by the fundamental of the noise is an integer. If there is only one noise term then the sampling rate could be set at a multiple of the noise. But, when there are multiple noise sources, the sampling must be set at a the lowest common denominator of the frequencies, which may not be practical.



Magnitude **(a)** and phase **(b)** response of a comb filter designed to remove 60 Hz noise and its harmonics at a sampling frequency of 5 kHz. Panel **(c)** shows the position of the poles and zeros of the filter (each cluster has ten zeros and ten poles!)

Figure 3.5: Comb filter.

3.2.4 Adaptive filters

Notch and comb filters have constant parameters and are relatively easy to design and implement for noise that is stationary over the duration of the experiment. But, in the real world, noise sources are constantly changing in amplitude, phase and even frequency. Adaptive filters adjust their coefficients over time to minimize the noise.

The adaptive filter is designed to minimize an error signal. For example, it can minimize the difference between an estimation of the noise and the real noise. The filter can update the coefficients of the filter at each time step to track the noise and adjust the zeros and poles of H accordingly. The filter can be designed to solve the Least-Mean-Square (LMS) solution to the error minimization. A LMS adaptive filter has two inputs: the signal corrupted with noise and a reference, an estimation of the noise. The objective of the filter is to adjust the amplitude and phase of the reference signal to maximize the correlation with the measured signal. The filtered output then is the difference between the reference and the measured signal.

Adaptive noise canceling applied to sinusoidal interferences was originally proposed in [51]. A block diagram of this algorithm is shown in figure 3.6. The goal of the filter is to recover the noiseless signal s by removing the noise n_0 from the measurement $d(k)$. The noise term is assumed to be uncorrelated to the noiseless signal, therefore the goal of the feedback is to maximize the correlation between the estimated noise signal and the measured signal. If the noise is described by a periodic signal $n_0 = A\sin(2\pi f_n + \phi_n)$, where f_n and ϕ_n are the frequency and the phase of the noise, we will search for a reference signal, $n_1 = B\cos(2\pi f_n)$, that is an estimation of the noise term present in d. The estimated noise signal is weighted at each sampling time to match the actual noise signal and then subtracted out. The weight is determined by adaptive filter adjusts by maximizing the correlation with the measurement using the error between measurement and the weighted reference as feedback. This algorithm iteratively minimizes the mean square value of the error signal [52].

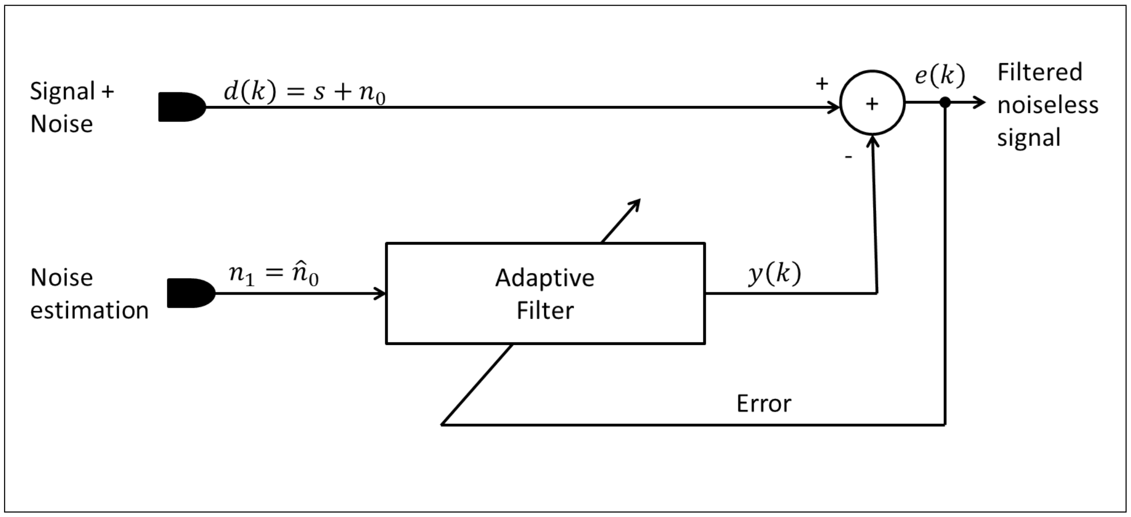


Figure 3.6: Block diagram of an Adaptive Noise Cancellation filter.

The feedback ($e(k)$) shapes the weights of the adaptive filter based on the LMS algorithm: Defining a M-length vector $\vec{w}$, (which can be initialized with zeros), the noiseless signal ($e(k)$) is obtained from:

$$e\left(k\right)=d\left(k\right)-y\left(k\right), \tag{3.4}$$

where

$$y(k)=\overrightarrow{w}^{T}\left(k\right)\overrightarrow{n_1}\left(k\right). \tag{3.5}$$

Hence,

$$e\left(k\right)=d\left(k\right)-\overrightarrow{w}^{T}\left(k\right)\overrightarrow{n_1}\left(k\right), \tag{3.6}$$

and the goal is to determine $\overrightarrow{w}\left(k\right)$ such that the mean square error of $e\left(k\right)$, (MSE), is minimized. MSE is calculated from [52]:

$$\xi\left(k\right)=E\left[e^{2}\left(k\right)\right], \tag{3.7}$$

$$\xi\left(k\right)=E\left[d^{2}\left(k\right)\right]-2\overrightarrow{p}^{T}\overrightarrow{w}\left(k\right)+\overrightarrow{w}^{T}\left(k\right)\mathbf{R}\overrightarrow{w}\left(k\right), \tag{3.8}$$

where

$$\overrightarrow{p}=E\left[d\left(k\right)\overrightarrow{n_1}\left(k\right)\right], \tag{3.9}$$

$$\mathbf{R}=E\left[\overrightarrow{n_1}\overrightarrow{n_1}^{T}\right]. \tag{3.10}$$

$\xi\left(k\right)$ can be seen as a quadratic vectorial function of $\overrightarrow{w}\left(k\right)$ with a unique global minimum. This minimum represents the optimal weighting of the reference noise $\overrightarrow{w_0}\left(k\right)$ to maximize correlation with the measurement. When the filter is first turned on, the weights may be initialized at random $\overrightarrow{w}\left(k\right)$, the global minimum is then achieved following the steepest descend path at each iteration, based on the instantaneous tangent of $\xi\left(k\right)$:

$$\overrightarrow{w}\left(k+1\right)=\overrightarrow{w}\left(k\right)-\frac{\mu}{2}\nabla\xi\left(k\right) \tag{3.11}$$

where μ is a positive constant that is called the *step-size parameter* . When the statistics of $\overrightarrow{p}$ and $\mathbf{R}$ are unknown, the instantaneous squared error $e\left(k\right)$ can be used to estimate $\nabla\xi\left(k\right)$:

$$\xi\left(k\right)\approx e^{2}\left(k\right), \tag{3.12}$$

$$\nabla\xi\left(k\right)\approx2\left[\nabla e\left(k\right)\right]e\left(k\right). \tag{3.13}$$

Since $e\left(k\right)=d\left(k\right)-\overrightarrow{w}^{T}\left(k\right)\overrightarrow{n_1}\left(k\right)$, $\nabla\xi\left(k\right)\approx-\overrightarrow{n_1}\left(k\right)$, hence

$$\overrightarrow{w}\left(k+1\right)=\overrightarrow{w}\left(k\right)+\mu\overrightarrow{n_1}e\left(k\right), \tag{3.14}$$

where μ determines stability and speed of convergence. The algorithm is stable for $0<\mu<\frac{2}{\lambda_{max}}$, where λ_{max} is the largest eigenvalue of $\mathbf{R}$. The speed of convergence, *i. e.* from the initial condition to the global minimum $\overrightarrow{w}\left(k\right)$, is proportional to $\frac{1}{\mu\lambda_{min}}$, where λ_{min} is the minimum eigenvalue of $\mathbf{R}$ [52].

It can be shown ([51, 53]) that an approximation of the transfer function to calculate $e\left(n\right)$ from $d\left(n\right)$ in an LMS filter is given by the second order filter transfer function:

$$H\left(z\right)\approx\frac{1-2cos(2\pi f_n)z^{-1}+z^{-2}}{1-2\left(1-\frac{\mu MB^2}{4}\right)cos(2\pi f_n)z^{-1}+\left(1-\frac{\mu MB^2}{2}\right)z^{-2}}. \tag{3.15}$$

When the step size μ is close to zero, 3.15 is equivalent to the transfer function of a second order notch filter (see 3.3), because if $\zeta=1-\frac{\mu MB^2}{4}$, then $\zeta^2=1-\frac{\mu MB^2}{2}+\left(\frac{\mu MB^2}{4}\right)^2$. But, if μ is small, then $\zeta^2\approx1-\frac{\mu MB^2}{2}$.

The corresponding bandwidth BW at -3 dB, in Hz, is given by

$$BW=\frac{\mu MB^2}{4\pi T}, \tag{3.16}$$

where T is the sampling period, in seconds. If the user selects the length $M$of the filter, as well as the desired bandwidth, BW, and the amplitude of the reference signal B is set to an arbitrarly value, let say one, then μ can be determined from 3.16.

For a signal where there are n harmonics we use n references, each one with a magnitude of one (parameter B in 3.16). Then, we use n M weighting filters, each one with an independent μ calculated for each reference. The estimate of the noiseless signal $(e(k))$ is obtained from:

$$y(k)=\sum_{i=1}^{n}\sum_{j=1}^{M}w_{ij}n_{1ij}, \tag{3.17}$$

$$e\left(k\right)=d\left(k\right)-y\left(k\right), \tag{3.18}$$

$$\overrightarrow{w_i}(k+1) = \overrightarrow{w_i}(k) + \mu_i \overrightarrow{n_1^i} e(k)\,. \tag{3.19}$$

$\mathbf{w}$ is an $n \times M$ matrix containing the n references filtered through n M-taps weighting filters, $\vec{w_i}$ is the M size i^{th} row vector of $\mathbf{w}$ and w_{ij} is the $(i,j)^{th}$ component of $\mathbf{w}$. $\mathbf{n_1}$ is an $n \times M$ matrix where the last M values for each one of the n references are stored at each sampling time k, and $\vec{n_1^i}$ is the M size i^{th} row vector of $\mathbf{n_1}$. $\vec{\mu}$ is a vector of length n containing the independent weighting factors for each reference, being μ_i its i^{th} component.

In order to calculate μ, we use a fixed value of $M = 80$ taps, set the magnitude of the reference to be 1 and determined the bandwidths from the data (as detailed in the next section). The number of taps should be long enough in order to account for an entire cycle of the slowest noise component we wish to be removed. For example, if the sampling frequency is 5000 Hz, and the slowest noise component to be removed is 60 Hz, an entire cycle of the reference requires 84 samples.

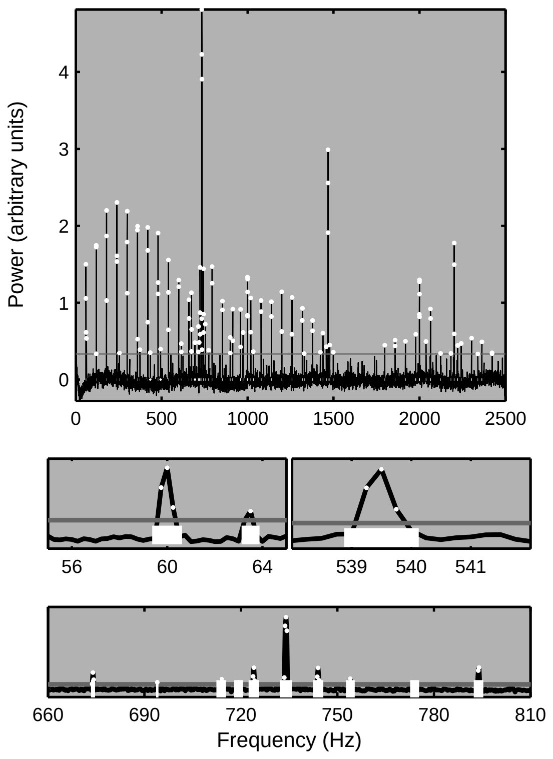
In this adaptive filter, there is one parameter that must be selected by the user μ. This parameter determines the speed of convergence, tracking, notch attenuation and bandwidth. It is possible to use another adaptive feedback algorithm in parallel to the adaptive filter to optimize this parameter automatically [54].

3.2.5 Selecting frequencies and bandwidth

If the noise sources are known, it is possible to set up filters with frequencies and bandpass widths to remove all the noise sources. Often the noise sources are not known, and even those with known sources, such as 60 Hz, may not behave as expected. It can be seen in figure 3.2, that the harmonics of 60 Hz may not be exactly at multiples of 60 [1] , and their amplitudes do not decay monotonically with frequency. Therefore, identifying the frequencies and their bandwidths of the different noise signals may be done empirically. Because the power drops off with frequency, it is not possible to set one threshold, above which a signal is considered noise and below which the signal is likely to be our signal of interest. This can be dealt with by detrending the spectrum, assuming that the noise signals make a small fraction of the entire spectrum. This can

[1] Frequency of the power supply may drift due to changing load conditions [55]

be done by fitting the power spectrum of the time series with a high order polynomial (I chose 10). This fit polynomial is then subtracted the power spectrum 3.7. The noise signals are identified as those that greater than two times the standard deviation of the detrended spectrum. Each noise source usually shows up in a few contiguous frequency bins. We determine the frequency as the average frequency and the bandwidth μ by the number bins that are above threshold (3.16).

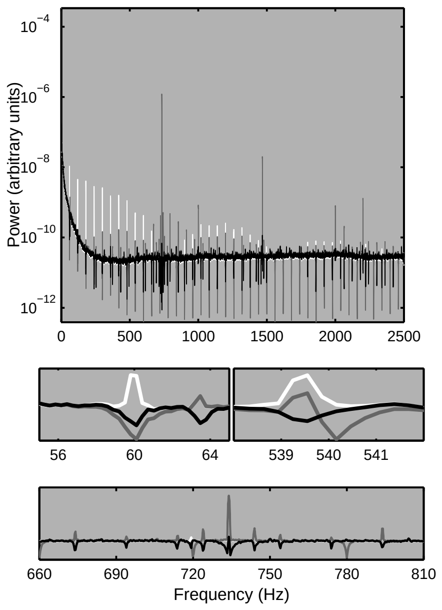


An alternative approach is to use another sensor to pickup just the noise. Then, this signal would replace the estimation of the noise (signal n_1 in figure 3.6.)
Black line shows the detrended spectrum of the original signal. Gray line is the threshold value used to identify the point noise sources, white dots indicate the corresponding frequencies and the white line shows the corresponding bandwidth calculated for each frequency. Top panel show the power as a function of frequency of the signal. Middle left panel shows a close up in the vicinity of 60 Hz, middle right panel in the vicinity of 540 Hz and lower panel in the vicinity of 735 Hz.

Figure 3.7: Selecting the main noise sources.

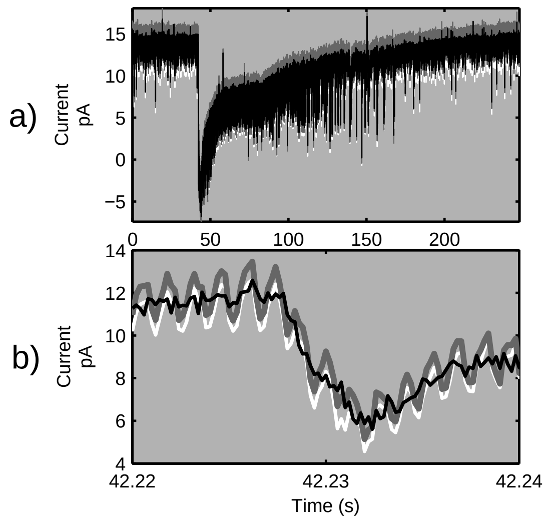
3.2.6 Filtering the data

Here I present the results of filtering the data shown in figure 3.1 through two adaptive filters. The first filter removes the harmonics of 60 Hz using a bandwidth of one Hz. In the second filter, the frequency components were selected empirically, as described previously. The spectrum of the original and filtered signals are shown in figure 3.8, and their corresponding time series in figure 3.9. As expected, filter 1 fails to remove harmonics that do not exactly match an integer multiple of the fundamental. Notice that the strongest component has a fundamental of approximate 734 Hz which filter 1 fails to remove. Filter 2 removes all the harmonics that were selected. These results show that the adaptive filter successfully removes the frequency components of the signals used as references of noise and making a good selection of harmonics improves the overall performance of the filter.



Top panel show the power as a function of frequency of the signal. Middle left panel shows a close up in the vicinity of 60 Hz, middle right panel in the vicinity of 540 Hz and lower panel in the vicinity of 735 Hz. The original signal is plotted in white. Gray line shows the filtered signal removing 60 Hz and its harmonics. Black line is the filtered signal where noise sources are identified empirically.

Figure 3.8: Power Spectral Density of a signal before and after being filtered.



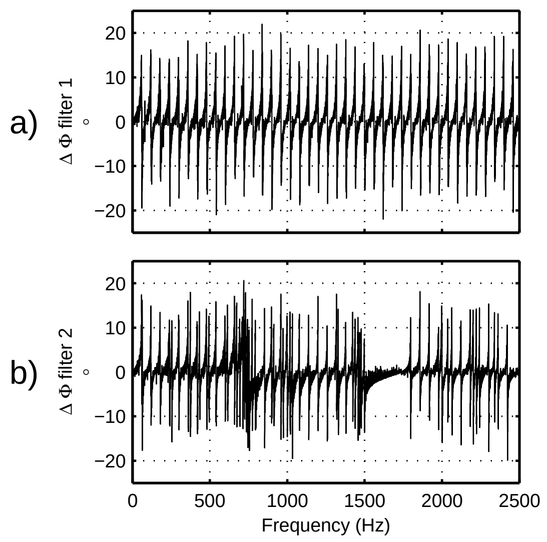
(a) shows the entire recording and **(b)** is a close up at 42 s. The original signal is showed in white. Gray line shows the filtered signal removing 60 Hz and its harmonics. Black line is the filtered signal when the main noise sources were removed.

Figure 3.9: Intracellular current recording and its filtered versions.

Notice that the time series of the filtered signals seems to have no delays with respect to the original signal (see figure 3.9), which indicates a negligible effect in phase change induced by the filters. To quantify the phase change, we windowed the time series to calculate the Fourier Transform of the original data and their filtered versions. At each window, the phase was calculated for each signal: original and filtered, and the phase change was obtained by subtracting the phase of the filtered signal from the phase of the original signal. Then, the phase change was averaged across the windows and is shown in figure 3.10. Notice that the maximum effect of phase change occurs at the notch frequencies. The maximum phase change caused by the adaptive filter is much lower than that caused by the second order notch filters.

3.3 Discussion

Electroencephalogram recordings include signals originating from neural source as as well as those from noise sources, as shown in figure 3.2. A common source of noise



(a) shows the phase change induced into the original data by the filter that removes the harmonics of 60 Hz. **(b)** shows the phase change generated for the filter where noise was identified epirically 3.7.

Figure 3.10: Adaptive filter's phase change.

is line noise, which occurs at 60(50) Hz and its harmonics. This noise introduces coherence interference with desired signal. Linear filters, such as second order notch filters, can be easily designed to remove periodic signals such as these, if there are relatively few to remove and the noise sources are approximately time invariant over the duration of the experiment. However, in the real world, even line-noise varies in frequency and amplitude and the harmonics do not occur at exact multiples of the fundamental. Furthermore, there are usually other noise sources which are unique to each recording environment. As EEG is sampled at higher and higher frequencies as hardware advances, there is an increased number of noise bands that must be removed. Using a cascade of second order notch filters to remove the noise bands can result in poor bandpass characteristics due to interactions from the ripples from each filter in the pass band (see figure 3.4).

Even comb filters, where the interactions are accounted for, are not flat at the passband. Comb filters are also limited to eliminating frequencies that can be evenly divided into the sampling rate. If there is a single noise source, adjusting the sampling rate to design a comb filter may be feasible, but if there are multiple noise sources with different fundamentals, this may not be a solution.

Adaptive filters based on the Least-Mean-Square algorithm can be very effective at removing noise signals once the frequencies at which the noise bands are determined. The main assumption is that the noise present in the signal is additive and uncorrelated with the neuronal activity. The algorithm compares the M tap weighted reference noise signal $y(k)$ with the measured signal and adjusts the weights to maximize correlation between the two signals. The noise-free signal is then determined by subtracting the reference from the measured signal. This approach can be used to filter multiple harmonics and can be used for post-processing or it can be implemented in real time.

μ is a critical parameter that determines the stability and speed of convergence of the filter. We ensure stability calculating μ from the equivalent transfer function of the adaptive filter. This approach relates μ to the bandwidth of the notch.

If the frequency of the noise sources are known, but the phase and amplitude is not, the adaptive filter can be designed to remove those frequencies. However, in most cases, we do not know the frequencies of the noise a-priori. However, with a short recording and inspection of the power spectrum, the noise source frequencies and bandwidths can be identified empirically and an adaptive filter designed. This empirically determined adaptive filter design is relatively simple to implement and can be done in real time, it has great filtering characteristics that remove power at the desired frequencies with good phase delay characteristics. However, if noise source frequencies emerges and then disappear, the initial screening would not capture this transient noise components. In this case, the most convenient approach is to use a second sensor to capture "just the noise". Then, the adaptive filter will remove the components that have the same spectral characteristics than the ones captured by the second sensor. The problem with this approach is that the second sensor could capture some dynamics of the desired signal. Then, the adaptive algorithm would remove noise and signal. Finding the best locating for the second sensor is the best strategy to find a balance between the levels of noise and signal that would render acceptable results.

Adaptive filters can also be applied to follow particular bands in the EEG. Alpha, beta, gamma, or delta bands are associated with particular cognitive states. If the filter is set to follow a particular band, then the signal $y(k)$ from 3.17 becomes the estimation of that band and the error from 3.18 is just calculated to update $\vec{w}$.

Chapter 4

Spike Rate Control

Some electrophysiology experiments require periodically firing neurons. One example is when measuring a neuron's phase response curve (PRC) where a neuron is stimulated with a synaptic input and the perturbation in the neuron's period is measured as a function of when the stimulus is applied. However, even regular spiking cells have considerable variations in their period. These variations can be categorized into two types: jitter, which characterizes the rapid changes in interspike intervals (ISIs) from spike to spike, and drift, which is a slow change in firing rate over seconds. The jitter is removed by averaging the phase advance of a synaptic input applied at a particular phase several times. The drift over long time scales results in a systematic change in the period over the duration of the experiment which cannot be removed by averaging. To compensate for the drift of the neuron over minutes, I designed a linear proportional-integral (PI) controller to slowly adjust the applied current to a neuron to maintain the average firing rate at a desired ISI. The parameters of the controller were calculated based on a first-order discrete model to describe the relationship between ISI and current. The algorithm is demonstrated on pyramidal cells in the hippocampal formation showing ISIs from the neuron in an open loop (constant applied current) and a closed loop (current adjusted by a spike rate controller). The advantages of using the controller can be summarized as: (1) there is a reduction in the transient time to reach a desired ISI, (2) the drift in the ISI is removed allowing for long experiments at a desired spiking rate and (3) the variance is diminished by removing the slow drift. Furthermore, I implemented an auto-tuning algorithm that estimates in real time the coefficients for each clamped neuron. I also show how the controller can improve the PRC estimation. The program runs on Real-Time eXperiment Interface (RTXI), which is Linux-based software for real-time data acquisition and control applications.

4.1 Introduction

The goal of this chapter is to show that a classical control engineering approach can be used to control the interspike intervals (ISIs) of a neuron. Even in neurons that tend to spike periodically when a constant current is applied, the ISIs show variations over time. These variations can be categorized into two types: jitter, which characterizes the rapid changes in ISIs from spike to spike, and drift, which characterizes the slow changes in the firing rate of the neuron over the time scale of seconds. Using control engineering concepts, I propose that it is possible to calculate the required current to be injected into the neuron in order to make it fire at a desired ISI using a dynamic clamp. We propose the use of a linear proportional-integral (PI) controller, where the parameters are calculated based on a linear description of how the ISIs change in response to steps of current close to the current required to make the neuron fire at the desired ISI.

One example of an experiment where regularly firing neurons are required for long periods of time is in the measurement of a neuron's phase response curve (PRC). The PRC measures how a synaptic input of a neuron perturbs the neuron from its period depending on the phase the synaptic input is applied. The PRC is useful in predicting how a network of neurons will synchronize. However, for an accurate measure of a PRC, it is necessary to minimize interactions between synaptic inputs, therefore only one synaptic input is applied and several cycles are allowed to pass for the neuron to return to normal before applying another input. The response of the neurons is also noisy; therefore neurons are stimulated at the same, or nearly the same, phase many times. These experiments can take several minutes requiring the neuron to fire at a consistent firing rate for the entire experimental duration to measure an accurate PRC. Even in very regular firing neurons, there is too much variability over the duration of an experiment to measure the PRC without adjusting the current applied to the neuron. Removing the long-term drift can improve the measurement of PRC and can be done through closed-loop control.

A controller compares the output of the process with a desired value (target) and depending on the difference (error) the controller modifies the input to the process in order to force it to achieve the target. The target can be a numerical value or a trajectory. Good closed-loop performance of the controlled process means that errors are corrected relatively quickly, that there is on average no error and the closed-loop system must be stable. To tune a controller to provide good performance some knowledge of the natural response of the process to be controlled must be known. When the desired quality attributes of the controller performance are known as well as the response of the system to a change of input, it is possible to determine the controller parameters to make the process achieve success. Examples of quality attributes of control may be how fast the target value should be reached when this target is changed, the maximum allowed overshoot in response to perturbations or changes in target, how insensitive to perturbations the system should be, etc.

Proportional-integral derivative (PID) controllers are most popular for their ease of implementation. They are widely used in different applications in the chemical, automotive, aeronautics and other industries [56]. In a PID controller, the error between the output of a system, such as a neuron's ISI, and the target value is used to calculate the input to the process, such as the current applied to the neuron. The PID output is based on the weighted sum of three measurements based on the error, the proportional error (P), its integral over time (I) and its derivative over time (D). The weighting of each of these factors depends on the dynamics of the process to be controlled and the desired behavior of the closed-loop system. Therefore, the first step in the design of a controller is to obtain an open- loop model that describes the input-output relationship of the process to be controlled. Then, the three parameters of the controller that satisfy the closed-loop quality attributes can be calculated.

Here, the goal is to control the ISI (output) of a neuron by the injection of current through an electrode using a dynamic clamp. The ISI is a discrete time variable calculated as the difference in the timings between peaks of action potentials from the measured voltage when current is applied to the neuron. While both voltage and current are continuous variables, we will only consider the time between action potentials and only change the current during action potentials, leaving it constant for one period of the neuron; therefore, we treat the neuron as a discrete point process.

To design a controller, it first must be determined what order the system is. For linear continuous processes where the time evolution of the output given an input is described by differential equations, the order of the system is defined as the highest order of the differential equation. In a zero-order system, the response in the output of the system to a change in the input is instantaneous and solely dependent on the amplitude of the input (i.e. it is not dependent on the history of the input). In a first-order system, when a step input is applied, the output goes from its initial value to its final stable state value following a monotonic, exponential change. First- order systems can be characterized with three parameters: (1) the gain, the amplitude of the change in the output given a change in the input, (2) the time constant, how long it takes to achieve a steady state response and (3) the dead time, the time between when a change of the input is applied and a change in the output is seen. Both the time constant and dead time are always positive but gain can be either positive or negative. Higher order systems have more complicated time responses (see [57] for a review in modeling).

The interaction between a controller and a system is through the mathematical operation called convolution [58]. The convolution, which requires an integral, is difficult to solve in the time domain; therefore, control design is done in the frequency domain by taking a Laplace transform (for continuous systems described by differential equations) [45] or the z-transform (for discrete systems described by discrete difference equations) [59] where the integral becomes simple algebra. In frequency space, the model for the system is called the transfer function. The difference is that convolutions in the discrete time domain are calculated as the summation of the product of the system and the applied signal instead of the integral of the product. Several Control Engineering and Signals and Systems textbooks go into detail on generating linear models and controller design using the Laplace transform and z-transforms in detail. We recommend [56, 59, 57] for a good introduction.

To model the neuron, we apply a step of current to the neuron and we fit a discrete difference equation to capture the change in the ISI and how long it takes to settle to a steady value. A neuron's response to a current step is nonlinear (i.e. doubling of the size of the current step may not produce a doubling in the firing rate), but we assume that if the model is fit using small current steps near the target firing rate of the neuron, the neuron's response will be nearly, or locally, linear. Because a PID controller is rather robust, even a very approximate estimate of the gain and time constant will produce decent control.

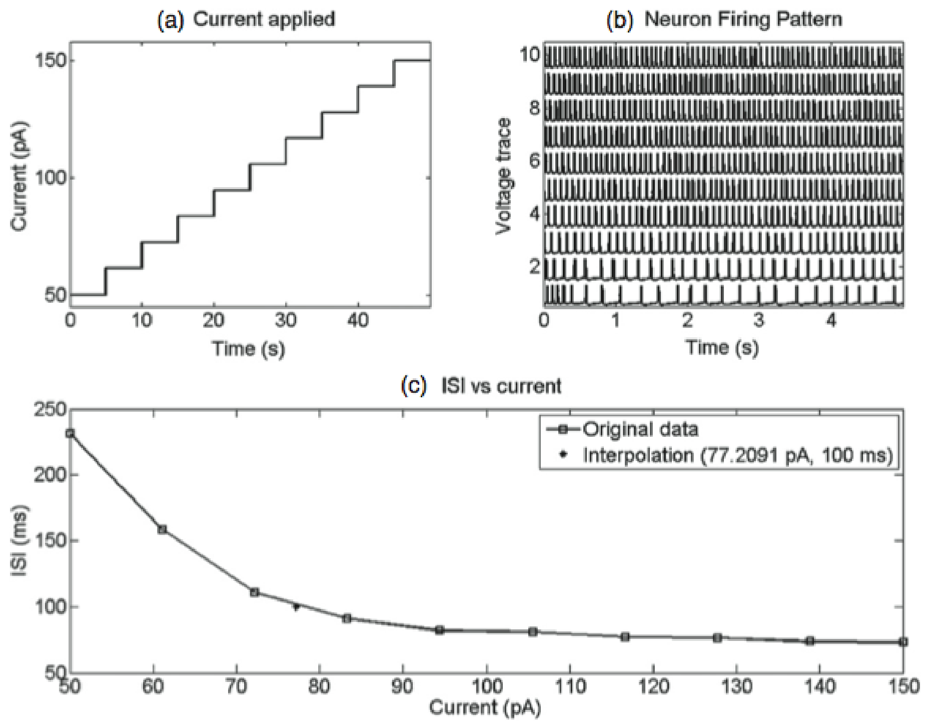
Because the goal is to cancel out the slow drift in the neuron's ISI rather than the jitter, and since I am only interested in keeping the neuron at a fixed target rate (rather than a moving one), it is only necessary to fit a first-order model to a neuron that captures the slow change of the ISI. We will use this simple model of the ISI to current relationship to compute the parameters of a controller with the goal that the system responds to changes in current rapidly (on the order of tens of spikes) and monotonically (without oscillations). We decided not to use the derivative term of the PID control because of the high variability between spikes and the inaccuracies in the modeling. For that reason, we just use the proportional and integral of the error (P and I) parameters, excluding the derivative term (D).

In this chapter I will first show how the parameters of the PI controller are derived to obtain critical damping of the neuron's response (i.e. with no oscillations). Then, I will demonstrate the controller on pyramidal cells in the hippocampal formation using a dynamic clamp. The performance of the controller will be compared to the variability of the ISI with open-loop application of a constant current. I will also demonstrate a short algorithm implemented to automatically determine the PI coefficients. Finally, I will illustrate the utility of the controller by measuring PRCs of neurons with open-loop and closed-loop control.

4.2 Results

4.2.1 System identification

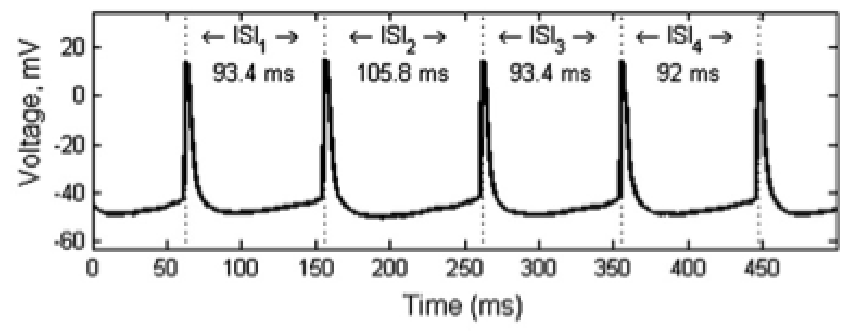
The first step in the design of a controller is to obtain a model that describes the dynamics of the process to be controlled. In this case, it is necessary to characterize the causal relationship between the ISI and the applied current; therefore I applied steps of current and measured the ISIs from the voltage traces, as shown in figure 4.1. The injected current steps applied ranged from the minimum current that causes regular firing of the neuron to a maximum where the cell is close to the saturation in its firing rate. The ISI is the difference in times between the neuron's action potentials; hence it is a discrete variable, as illustrated in figure 4.2. While the current applied to the neuron is continuous, we only change it at the onset of the action potential and therefore we treat it as a discrete value sampled at the same time the ISI is measured. The current for the nth ISI affects $ISI[n+1]$.

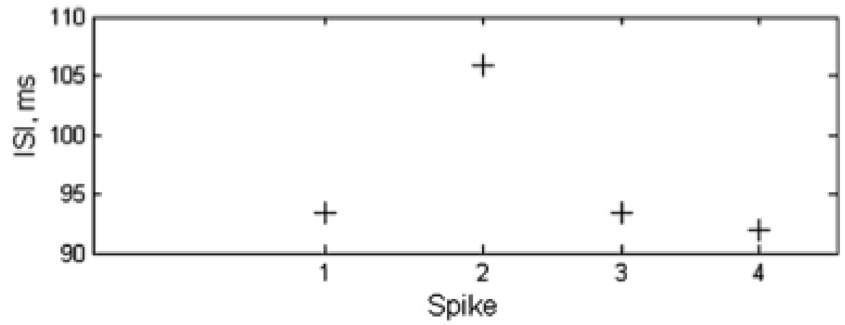


(a) Each 5 s the current injected into the neuron is increased. **(b)** 5 s voltage trace for each current step is plotted on a separate line. Firing rate increases with increasing current, and the corresponding effect is an increase in the frequency as indicated by the reduction in timing between spikes. **(c)** The mean ISI was calculated for each 5 s of data and plotted as a function of the corresponding applied current. Data are interpolated (using a cubic spline) to infer the current required to make the cell spike at 100 ms.

Figure 4.1: Current-step response of a neuron.

When the current applied to the neuron is increased, the change in the ISI is not proportional, as shown in figure 4.1 (c). The change in the ISI is greater at low currents than at high currents, which indicates that the system is nonlinear. However, if the goal is to control the ISI at a particular value, we can assume that the system is linear in response to small perturbations of the current in the vicinity of the desired ISI. I chose to control the neuron around the target ISI of 100 ms. A first approximation of the current necessary to fire at this ISI can be estimated by interpolating the ISI-current





(a) The voltage trace and the difference between peaks. **(b)** Plot of the ISI as a function of spikes.

Figure 4.2: Calculation of the ISI from a voltage trace.

curve and solving for the current at the desired ISI, as shown in 4.1 (c). For this neuron, 77 pA of applied current should produce an average ISI of 100 ms.

If we treat the ISI response of a neuron to current input as a first-order process, then the difference equation that relates the changes in ISIs on the nth spike ($\Delta ISI\left[n\right] = ISI\left[n\right] - ISI\left[n-1\right]$) as a function of changes in current ($\Delta I\left[n\right] = I\left[n\right] - I\left[n-1\right]$) is given by

$$\Delta ISI\left[n\right] = a\Delta ISI\left[n-1\right] + K\Delta I\left[n\right], \tag{4.1}$$

where a determines how quickly the ISI reaches its final value and K is the gain. K has the dimensions of ms/pA and is calculated as

$$K = \frac{\Delta ISI_{ss}}{\Delta I} \tag{4.2}$$

where ΔISI_{ss} is the difference in the steady state ISI for a given change in current. The coefficient a determines how quickly the ISI settles to the steady state firing rate and is related to what might be called the time constant of the system, τ , by the following equation:

$$a = 1 - \frac{1}{\tau}. \tag{4.3}$$

The coefficient τ determines how many spikes occur following a change in the applied current before the neuron reaches 66% of the way to the steady state ISI. τ is often thought of as a time constant, but because this is a discrete time system this instead determines the number of spikes required for the system to settle to the steady state ISI and I will use 'time' loosely. The settling time is defined as the time after the application of a step input for the process to reach and stay within a stated plus and minus tolerance band around its final value. For a first-order system, the time to reach 5% of the steady state value is approximately three time constants [57] and after five time constants the systems has reached more than 99% of its final value. Therefore, we estimate how many spikes it takes for the neuron to settle to a steady state and divide that time by 5 to estimate τ.

Controller design is easier in the frequency domain than in the time domain; therefore we calculate the z-transform of our model of the neuron and the controller. The z-transform is a frequency space representation ($X[z]$) of the system $x[n]$ and is defined as $X[z] = \Sigma_{n=-\infty}^{\infty} z^{-n} x[n]$, where z is defined as the complex number $z \equiv Ae^{j\phi}$ [45]. The z-transform of equation 4.1 is

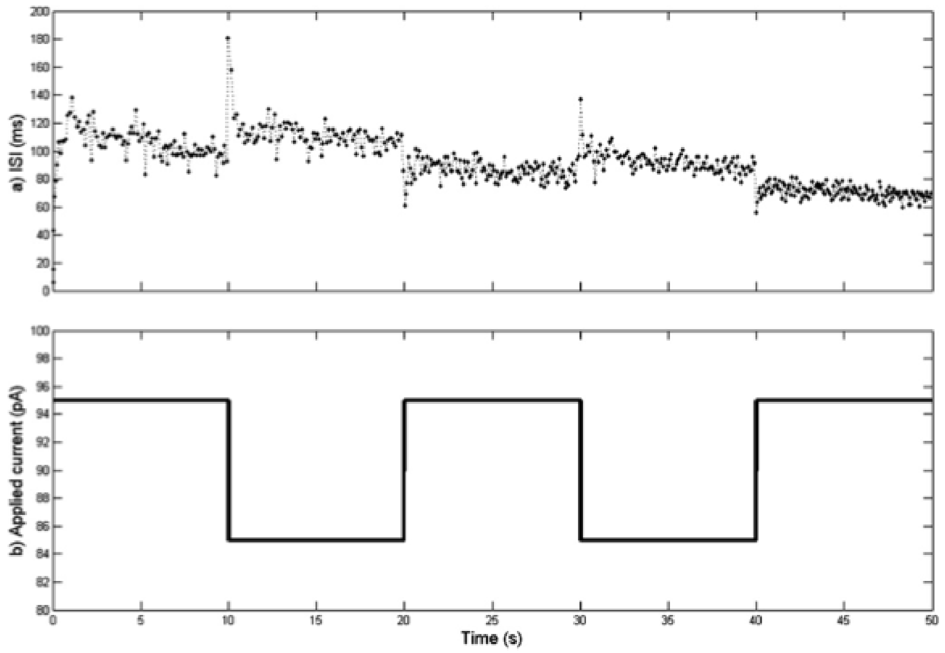
$$\Delta ISI\left(z\right) = a \Delta ISI\left(z\right) z^{-1} + K \Delta I\left(z\right), \tag{4.4}$$

where z^{-1} on the right-hand side of the equation is the z-transform of the time shift. The transfer function of the neuron $N(z)$ is given by solving for the output divided by the input and is expressed by the following equation:

$$N(z) = \frac{\Delta ISI_{ss}}{\Delta I}\left(z\right) = \frac{Kz}{z-a}. \tag{4.5}$$

The transfer function characterizes the frequency response of the neuron's ISI response to a unit impulse of current applied at discrete time $n = 0$ with zero initial conditions.

The coefficients a and K from equations 4.2 to 4.5 are fit for ISI measurements from a neuron following a step in current. Because the step response is noisy, I apply several steps up and down in the current bracketing the target ISI, as shown in figure 4.3. It can be seen that there is a transient (overshoot) in the ISI following the current step and then it tends to a final state. The gain in neurons is negative, *i.e.* an increase in current decreases the ISI and vice versa. As the steps in the applied current are not symmetric around the exact value of current to produce an ISI of 100 ms, the final ISIs after each step of current are different. Figure 4.3 also illustrates the two noise components present in the ISI: drift and jitter. The drift appears here as a different stable state ISI when positive (and negative) current is applied and the jitter is always present as variability in ISIs spike after spike.



(a) The ISI in each block of the applied current shows two regions: there is an overshoot in the first two or three spikes and, after that, the ISI tends to a final ISI following a first-order dynamic. **(b)** Corresponding injected current.

Figure 4.3: Alternating current injection in the vicinity of the target ISI.

The response of a first-order model cannot generate overshoots. Its response evolves monotonically until it reaches a stable state. The ISI response obtained experimentally exhibits a transient with overshoots that last a few spikes. I performed system identification with higher order models and found that a fourth-order model could explain most of the neuron's response to a current step. However, as the goal of this method is to remove the long-term drift, a first-order model was sufficient to account for the short transient time and the slower remaining dynamics were ignored without much deficit. It is also easier to derive the controller's parameters and implement a first-order fit to the neuron's response in real time.

The gain is calculated using equation 4.2. For example, in the data shown in figure 4.3, 85 pA are applied to the neuron from time 10 and 20 s resulting in an average ISI of 110 ms after a transient. When the current is increased to 95 pA at 20 s, the ISI stabilizes at 93 ms after the transient. Therefore the gain is $\frac{110-93}{85-95} = -1.7ms(pA)^{-1}$. The transient in the response can be seen immediately after a change in current as an overshoot in the ISI for two or three spikes. The ISI eventually reaches a final value of around 110 ms. The time constant can be calculated by counting the total number of spikes required to reach the average ISI (including the initial two or three spikes in the transient) and dividing this number by 5, because after five time constants the ISI reaches more than 99% of its final value. Only a very rough estimate of this parameter is necessary to tune the controller for satisfactory results. It takes approximately six spikes for this neuron to settle; therefore I estimate $\tau = 1.2$ $spikes$.

4.2.2 Proportional integral parameters calculation

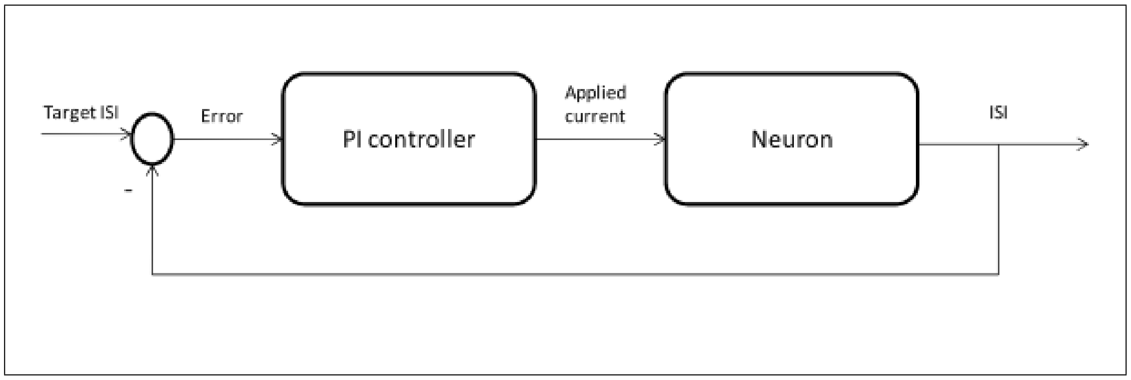


Figure 4.4: Closed-loop implementation to control the ISI.

A PID controller has three basic actions of control: a proportional, an integral and one derivative of the error. The proportional action generates an output that is proportional to the size of the error. If the error changes, the proportional action also changes to reduce the error. However, when the error remains at a constant value, the proportional component does too, regardless if the system had reached the target or not, which leads to a steady state error. A proportional controller may decrease the error but will always result in a persistent error. The persistent error can be removed by adding an integral term. By integrating the error, any small difference between the target and the actual ISI will accumulate until the controller compensates for the error. A derivative of the error can also be incorporated in the controller for predictive correction of the error. However, this requires an accurate model of the open-loop system to be useful, which we do not have. Furthermore, the neuron is very noisy with nearly zero correlation from spike to spike (as can be seen in the bottom left panel of figure 4.5), rendering the derivative useless. Therefore, I decided only to use the proportional and integral components of the controller.

Figure 4.4 is a schematic showing the closed-loop controller and the neuron. The equation of the PI controller is

$$\Delta I[n] = K_p e[n] + K_i \sum_{i=0}^{n} e[n-i], \tag{4.6}$$

where $e[n] = ISI[n] - Target_{ISI}[n]$ is the error in the ISI of the nth spike, K_p represents the proportional gain and K_i the integral gain. The z-transform of the controller is

$$PI(z) = K_p + \frac{K_i z}{z-1}. \tag{4.7}$$

The input/output (current/ISI) transfer function of the neuron and the controller connected together is $\frac{\Delta ISI}{\Delta i} = \frac{PI(z)\times N(z)}{1+PI(z)\times N(z)}$. By substituting equations 4.5 and 4.7 into this equation we obtain:

$$tf(z) = \frac{\left(K_p + \frac{K_i z}{z-1}\right)\frac{Kz}{z-a}}{1+\left(K_p + \frac{K_i z}{z-1}\right)\frac{Kz}{z-a}}. \tag{4.8}$$

This can be rearranged as

$$tf(z) = \frac{K(K_p+K_i)z^2 - KK_p z}{[K(K_p+K_i)+1]z^2 - (KK_p+a+1)z + a}, \tag{4.9}$$

where $K = \frac{\Delta ISI_{ss}}{\Delta I}$ and $a = 1 - \frac{1}{\tau}$, and τ is the time constant.

The dynamics of the closed-loop system is characterized by the denominator of the transfer function (which is known in the control literature as the characteristic equation). The roots of the characteristic equation determine the temporal response of the system. Because roots of the characteristic equation are the zeros of the denominator, they are also known as the poles of the transfer function:

$$0 = K(K_p + K_i) + 1]z^2 - (KK_p + a + 1)z + a. \tag{4.10}$$

In order to avoid oscillations in the response of a second-order system, yet still have a reasonably fast response time, it is necessary to find the parameters K_p and K_i that make the poles equal. This condition is known as critically damped. The poles of the closed-loop system are calculated using the following equation:

$$p_{1,2} = \frac{(KK_p + a + 1) \pm \sqrt{K^2K_p^2 + 2K(1-a)K_p + (1-a)^2 - 4aKK_i}}{2[K(K_p + K_i) + 1]}. \tag{4.11}$$

To force the poles to be the same, we need the portion under the square root to be zero:

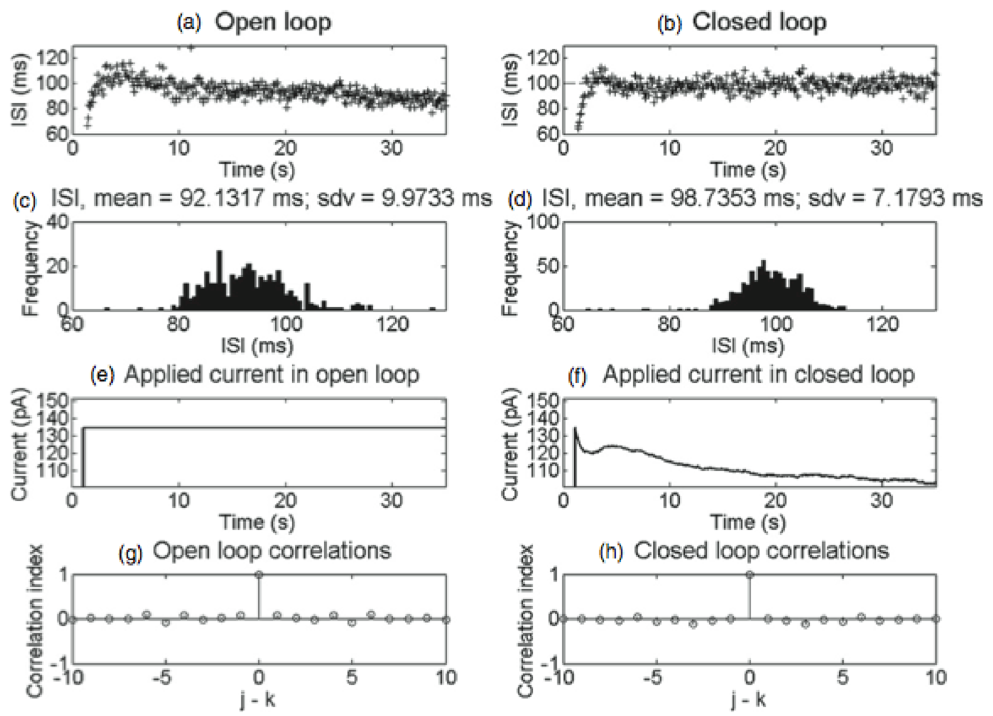
$$K^2K_p^2 + 2K(1-a)K_p + (1-a)^2 - 4aKK_i = 0. \tag{4.12}$$

It is possible to solve for this equation given K and a for a neuron and fix one of the proportional or the integral gains so that it remains only one variable to solve for. The goal is to eliminate slow drifts in the firing rate and not to change with jitter on each cycle; therefore I set $K_p \ll K_i$ so that the response is dominated by the integral and therefore is smooth to noise shocks on each cycle. I arbitrarily set $K_i = 100K_p$ and substitute it into the above equation and solve for K_p. Therefore

$$K_p = \frac{1}{K}\left(201a - 1 \pm 20\sqrt{101a^2 - a}\right). \tag{4.13}$$

There are two solutions to this equation. I use the negative root of $101a^2 - a$ (the other root changes the sign of the gain). The temporal response of the system can be obtained calculating the inverse z-transform of equation 4.9.

To test the controller performance, I compare the ISI histograms obtained in the same neuron under two conditions: (1) when a constant current is applied and (2) using the controller to adjust the applied current (this is a different neuron from the one used in the previous figures, for this reason the required current to produce an ISI of 100 *ms* is different). The use of the controller renders four positive outcomes that can be appreciated in figure 4.5.



(a) In the open-loop configuration, ISI experiences a significant drift over the ∼30 *s* recording after a constant current was applied. The error in the ISI toward the end of the trace (30 to 35 *s*) is 10 *ms* on average for a target interval of 100 *ms*. Autocorrelations (bottom graph) of first lag are nearly zero (see the bottom row). This indicates that the error from one cycle is almost completely independent of the previous cycle. **(b)** With closed-loop control, the ISI converges quickly to the target rate of 100 *ms*. The mean ISI, after the initial transient, is statistically indistinguishable from the target rate. To maintain the neuron at the target ISI, the current injected into the cell is varied. Standard deviations of the error in the open loop and closed loop are similar, indicating that the closed loop only reduces the drift in the ISI rate and not the variability from spike to spike. The autocorrelation at the first lag is nearly zero. If feedback gain (from the proportional feedback coefficient) is too high, the first lag of the autocorrelation will be negative, indicating a ringing of the controller.

Figure 4.5: Neuronal response to change in the target ISI rate using open-loop and closed-loop controllers.

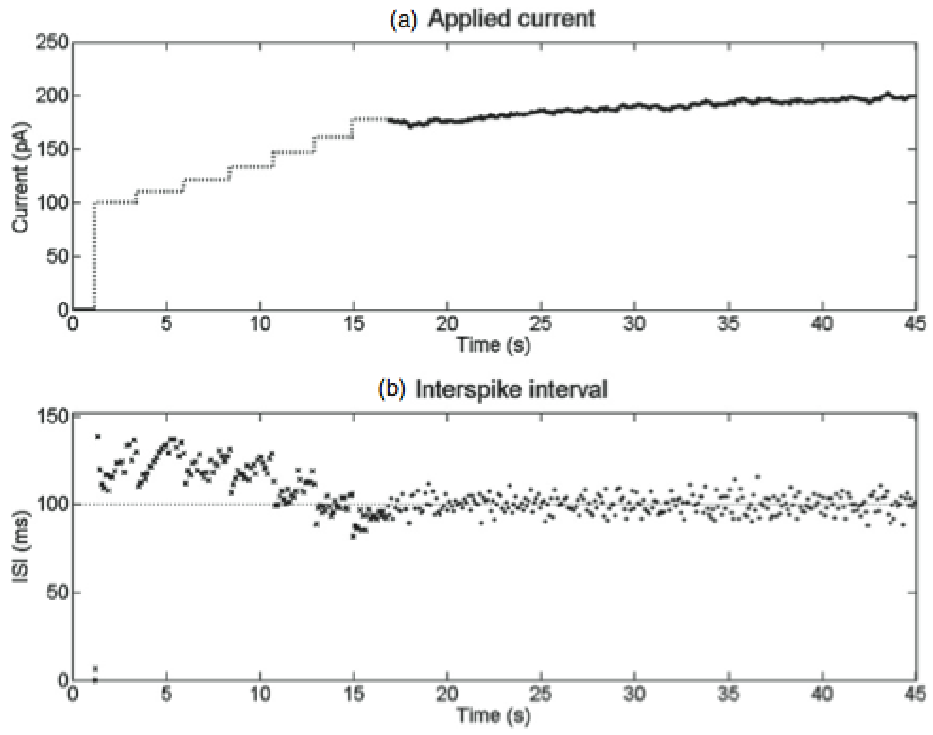
- **There is a reduction in the transient**. The use of the controller helps the cell reach the target in a shorter time than with the application of a constant current.
- **The drift is canceled out**. In an open loop, the mean of the ISI was 92 *ms* but the use of the controller is moving the mean to the target, which is 100 *ms*.
- **Variance is diminished**, as indicated by the reduction of the standard deviation over the entire time from 10.0 to 7.2 *ms*. This drop in variability is from the removal of the drift and not from a decrease in the jitter.
- **No autocorrelation is introduced**. If the neuron is not tuned well then the controller and the neuron will oscillate as indicated by a significant value in the autocorrelation at the first lag. Even despite rough estimates of the gain and time constants, no appreciable autocorrelation was introduced.

4.2.3 Auto-tuning of a PI controller

I implemented an 'auto-tune' function in RTXI to characterize the neuron's response and calculate the proportional and integral coefficients for the controller. The auto-tune function starts by stepping the current injection until the target ISI is reached, as shown in figure 4.6, that shows the algorithm implemented in a cortical neuron. The current is applied until 20 spikes are detected to calculate the mean ISI. If the mean is greater than the target, the current applied to the neuron is increased. This procedure is repeated until the mean ISI is equal to or shorter than the target. When this condition is met, the function calculates the gain (K) and time constant (τ) of the neuron, which it uses to estimate the controller gains (K_p and K_i). Once the PI coefficients are established, the auto-tune function switches to closed-loop control.

4.2.4 Phase response curves

Finally, I demonstrate the utility of the controller in measuring PRCs from pyramidal cells in the rat hippocampus. A PRC quantifies how a perturbation to a neuron affects the timing of the next spike as a function of when the stimulus is applied [60]. I perturb the neurons with transient stimuli to simulate synaptic inputs. Because network activity can be seen as a series of synaptic perturbations from firing cells onto target



(a) Initial choice of current makes neuron fire slower than the target ISI. After 20 spikes, the mean ISI is calculated and if it is greater than the target ISI, the applied current is increased by 20%. This is repeated until the mean ISI is shorter than the target ISI. Once the current is stepped to a value that produces ISI shorter than target ISI the model and controller parameters are calculated. The dotted line represents the open-loop tuning current injected for system identification. Once parameters for a controller are estimated, the programs switch to a closed loop mode and are plotted as a solid line. **(b)** Below is the temporal evolution of the ISIs, where 'x' represents the ISI on the system identification. Once the program switches to the closed loop, the ISI (represented with dots) is kept at the target value (dotted line).

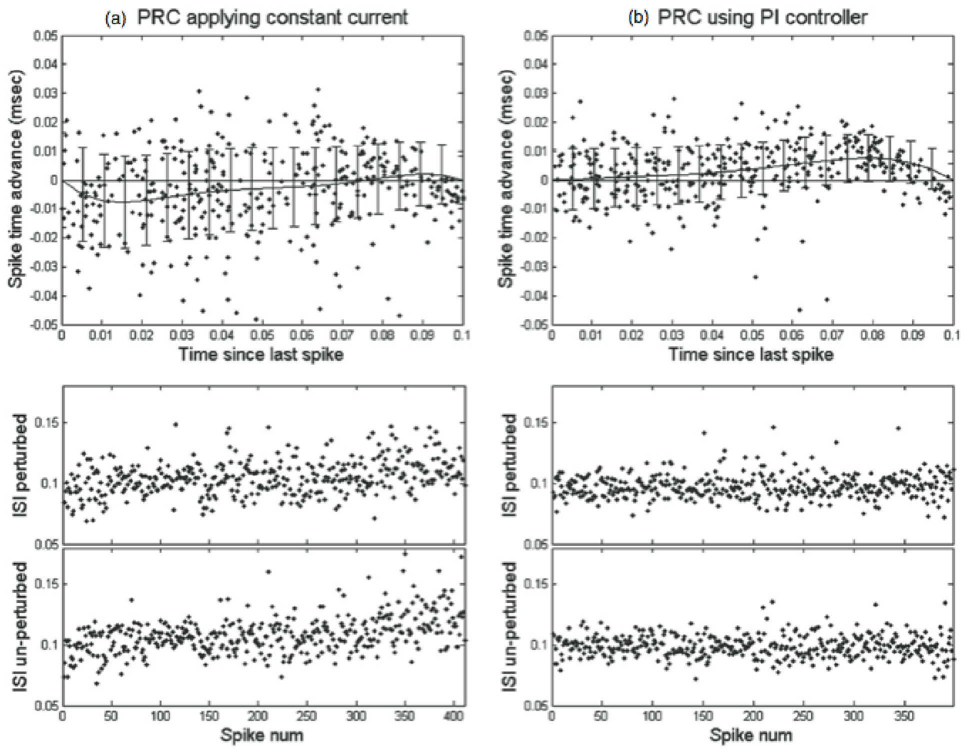
Figure 4.6: Running of the auto-tuning program in a cortical neuron.

neurons, PRCs can be used to predict synchrony in neural networks. However, one of the requirements to calculate PRCs is that the neuron should spike at a regular frequency to identify changes in its period caused by the perturbation. I use the controller to stabilize the ISI of the neuron while I measure its PRCs. As an example of the utility of the spike rate controller, I compare PRCs measured with and without the use of the controller.

I measured PRCs in the same patched neuron under two experimental conditions: (1) while injecting a constant current and (2) using the controller to maintain the neuron at a target ISI. The results are shown in figure 4.7. In both cases the PRC has a sinusoidal-like shape. However, in the open-loop experiment, the ISI drifts around 30% from beginning of the experiment until its end. In the closed-loop scenario, after a settling time the ISI remains in the target value. The effects of the synaptic inputs used to calculate the phase change are not canceled out by the controller. Another indication of the improvement achieved using the controller to measure PRCs is that the error bars indicate reduction of ISI variability in the closed-loop experiment due to the removal of the drift or from decreased spike variability.

4.3 Discussion

Designing a linear controller requires a linear model that relates the causal dependence between the parameter to be controlled and the input to the system to achieve the goal. As the goal is the cancellation of the long-term ISI drift on patched neurons, I require a model that relates the ISI to the applied current. I obtain a linear model using a step current response in the vicinity of the current that makes the neuron fire at a desired ISI. The model I fit to the neuron's dynamics reflects the drift in the ISI in the vicinity of the target value. When current is applied to make a neuron fire, the response in ISIs has two components: a rapid variation that occurs spike after spike termed jitter and a long-term drift in the order of seconds. The long-term drift, obtained as a step response experiment (figure 4.3), after a quick transient of usually two or six spikes, seems to behave as a first-order process, where the gain can be obtained as the ratio of the change in the ISI versus the change in the current, and the time constant of the process can be obtained by calculating the required time (in spikes) that the neuron



PRCs use a constant current **(a)** and adjust the current to remove the long-term drift on ISI **(a)**. Dots in top panels show the time advance as a function of the timing of the perturbation in the previous cycle, and the solid line corresponds to the fitted PRC function with error bars indicating the standard deviation. Middle panels show the ISI as a function of spike number of perturbed cycles while the bottom panels show the ISI of the unperturbed cycles immediately preceding.

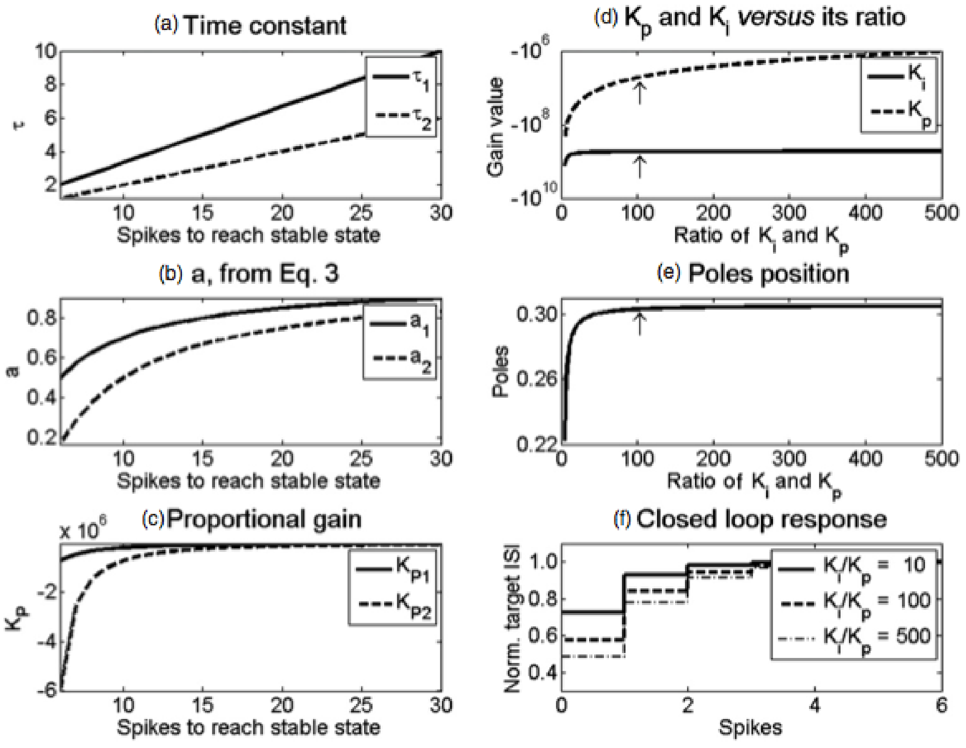
Figure 4.7: Closed-loop spike rate control improves the quality of estimated PRCs.

spent recovering from the initial overshoot to its final mean ISI including the spikes in the transient. For this reason, I assume the process to be a first-order discrete system.

To calculate the number of spikes required to achieve a stable state, I consider the first spike that crosses the mean ISI after the transient. In a first-order system, after five time constants the system achieved more than 99% of its stable state. The value of 5 renders a K_p absolute value bigger than when using a factor of 3 to calculate the time constant. The overshoot and oscillations observed in a closed loop (figure 4.5) can be caused by misestimates of the time constant or in the order of the model used to describe the neuron. The sensitivity to the estimation in the time constant can be seen in figures 4.8(a)-(c). When τ is estimated as 1/5 the time to the first crossing of the ISI over the target ISI, K_p is bigger than when it is estimated at 1/3. Panel (c) also shows that the estimation in K_p is most sensitive to misestimates where the stable state is achieved between six and nine spikes. However, K_p is rather robust because most neurons we have encountered require more than nine spikes to achieve a steady state (see figure 4.8(c)).

I chose $\frac{K_i}{K_p} = 100$ arbitrarily with the intention that the controller will smooth over the high variability from spike to spike and base the control signal on an average over many cycles. The use of different ratios between K_i and K_p generates almost the same closed-loop performance, assuming that the neuron is well characterized as a first-order system, as illustrated in figure 4.8(f). Solving equation 4.11 for different ratios of K_i to K_p shows that the value of K_p remains stable for ratios greater than 10 (figure 4.8(d)) and that the position of the poles is almost unaffected (figure 4.8(e)). The corresponding closed-loop responses for three different controllers when the ratio was set to 10, 100 and 500 are almost the same, as illustrated in figure 4.8(f), which indicates the robustness of the closed-loop performance for different ratios between K_i and K_p.

Robustness and simplicity were the main arguments considered to use a PI controller. PI controllers are a subset of PID controllers. PID controllers calculate the input applied to the process to be controlled based on the error between the target and the actual value of the variable to be controlled, and the total input is a weighted sum of the error, its integral and its derivative. If only the term that is proportional to the error is included, the controlled process will have stable state error. For example, Netoff *et. al.* used an adaptive proportional controller to keep the ISI at a desired value [61] and



Panels **(a)** - **(c)** show respectively the values of τ, a and K_p as a function of the number of spikes to reach a steady state. τ is either estimated as one-third time to a steady state (solid line) or one fifth (dashed line). The coefficient a was calculated using equation 4.3 and K_p from equation 4.13. Panel **(d)** shows the dependence of K_p and K_i on the ratio K_i/K_p (from equation 4.12). The arrows indicate the obtained gains using $K_i = 100$. Panel **(e)** shows the dependence of the poles on the ratio K_i/K_p. The poles were calculated using equation 4.11. The arrow indicates $K_i/K_p = 100$. Panel **(f)** shows the temporal response of the closed-loop Kp controller (i.e. the number of spikes to reach a steady state) for $K_i/K_p = \{10, 100, 500\}$. In panels **(c)** - **(f)** the value $K = -1.7 \times 10^{-9}$ $spikes/A$ was used as gain for the open-loop system.

Figure 4.8: Sensitivity in the calculation of τ, a, K_p and the poles.

their results show a stable state error in the ISI. The controller presented here responds faster and with smaller error.

When the process to be controlled shows high variability, it is not recommended to include the derivative term in the controller, because that term can introduce oscillations and instability. Figures 4.5 and 4.7 show that when the PI controller is implemented, the control goals are achieved (it keeps constant the ISI and reduces the ISI variance) without paying the price of having dramatic changes of current spike after spike or having a high correlated process between spikes.

Because the neuron's activity changes over time, the estimate of the coefficients at the beginning of the experiment may not provide sufficient control after some duration. The PI controller is rather robust, and should work over a large parameter range of the neuron, but if cell's dynamics drift and the user notices that the performance of the controller is deteriorating, the auto-tune algorithm can be re-run to estimate the parameters for the controller again. In our experience, the neuron goes into depolarization block and stops firing before the PI controller becomes ineffective.

The use of the controller allows to measure changes in a phase as a function of the timing of the synaptic input and the undesired long-term drift is removed, keeping the neuron in experimental conditions that enable it to be considered as a regular oscillator, which is one of the requirements for the calculation of PRCs.

The controller may be improved by increasing the accuracy of the model that determines the effect of current on the ISI. This improvement would likely require the incorporation of nonlinear modeling elements and an increase in the order of the system. A better model might enable us to implement a more sophisticated control strategy but it is doubtful that this would improve the quality of the closed- loop performance. This chapter shows that the use of a simple PI controller is enough to remove the long-term drift on the ISI, to decrease the variability on the ISI and to keep the neuron in experimental conditions to calculate PRCs without undesired effects as having a highly correlated relationship between spikes or having abrupt changes in current between spikes.

An important remark in the presented methodology is that I use the z-transform to calculate the controller parameters. In most engineering settings, the z-transformed is performed in the discretized version of a continuous process after it is digitalized using a digital to analog converter (DAC). In DACs, the discretization process is done at regular intervals (sampling time). The ISI is a discrete process per se but the sampling time varies at each point, because each sampling time is the actual ISI. As the z-transform operates in discrete time series regardless of its origin, there is no problem in using it to avoid the convolution in the calculation of the controller parameters.

Chapter 5

Data Assimilation using Unscented Kalman Filters

Parameter and state estimation of neurons is an unsolved problem because of their highly nonlinear behaviors. One solution used extensively in the engineering fields is data assimilation, where at each sampling time, the data is compared to predictions, which is then used to increase the accuracy of the model. Data assimilation has been used to make predictions in nonlinear and chaotic systems, like weather forecasting. One of these data assimilation techniques is called Unscented Kalman Filter (UKF). Here, I implemented the UKF in a real-time computing environment to estimate dynamics and predict the voltage of a neuron. In this chapter I present the equations and assumptions used to make the filter stable. I show that the prediction errors reach steady state after approximately one second of data assimilation. Then the voltage estimation is used to make the neuron follow an arbitrary voltage trace. The differences between the predicted and target voltages were used to calculate the current necessary to make the neuron follow a voltage trace determined by a computational neuron model. To our knowledge, this is the first time that the UKF has been used to assimilate data from neurons in real-time. This method can be applied to detect changes in dynamics of neurons over the duration of an experiment and perhaps across neurons. The ability to track parameters of a neuron over time may provide insight into the basic mechanisms of how homeostatic changes in ionic currents play a role in diseases like epilepsy, Parkinson's disease, and schizophrenia.

5.1 Introduction

Fitting highly nonlinear systems with many hidden states, such as Hodgkin-Huxley style equations, to experimental data is a difficult and an unsolved problem. Further complicating this problem is that biological systems are constantly changing over the duration of an experiment. Models are composed of states, which change to describe the dynamics of the neuron, and coefficients which are fixed. Ideally, if the model is complete, one set of coefficients should be sufficient to fit an entire data explaining all the behaviors of the neuron over the duration of the experiment. These changes in the dynamics of the neuron over time may be considered a nuisance, or may be indication of homeostatic changes within the neuron over the duration of the experiment. Instead, time varying coefficients may explain the data better, tracking the changes of the neuron over the duration of the experiment. Furthermore, if the model is going to be used in closed-loop experiments, where the model will interact with real neurons, estimating the coefficients must be done in real-time.

There are many different methods to fit mathematical models to experimental data [19], such as "hand-tuning" [20], *brute-force* parameter exploration [21, 22], gradient descend [23], simulated annealing [24], and synchronization-based methods [26, 27]. One successful approach for fitting coefficients in nonlinear models widely used in industry is the Unscented Kalman Filters (UKF). The UKF, and other versions of Kalman filters, are used extensively for weather forecasting [62]. In this chapter we chose the UKF to fit parameters of a model neuron to neuronal data. The UKF approach uses the model to select combinations of coefficients that best predict and voltage trace given the history. An advantage of the UKF approach is that it can be implemented in real time to improve estimation accuracy over time [63, 64, 65].

Only recently the UKF method has been introduced for fitting models in neuroscience [66, 67, 62]. The UKF formalism has been shown to successfully fit the model to reproduce the voltage and hidden states by adjusting parameters. When the initial parameters were selected randomly, the UKF adjusted parameters so that they converged to the actual values used in the model to generate the data [67]. The UKF does an excellent job when the same model is used to fit the data as was used to generate the data, but with randomized coefficients. However, problems arise when using the UKF to fit models to real neuronal data.

In this chapter we implement the UKF in real-time to estimate the states of a computational model of a neuron to match the intracellular voltage recording from rat hippocampal naurons. Here, we chose a single compartment model proposed by Golomb and Amatai [17] (*GACell*). The model equations are reproduced in the appendix B. The *GACell* has several different ionic currents. Their interaction mimics some dynamical features observed in cortical neurons. It accounts for two Na^+ currents (fast voltage gated and persistent), three K^+ currents (delayed rectifier, A type current, and slow current), as well as a leakage current. The model has 5 states, voltage and four gating variables (hidden states). By adjusting the coefficients, the *a priori* estimates of the errors reduced over the experiment, indicating that the model fit improved over time. The model was validated by using it to control the voltage of the neuron to follow the voltage trace from a reference neuron model.

Our ultimate goal is to achieve a model that can predict the behavior of the neuron's response to an arbitrary stimulus waveform. Unfortunately, the models we achieve are accurate in predicting the voltage when assimilating data, however when the model is run independently, we find that it does not accurately reflect the dynamics of the neuron, indicating that the UKF is accurate for predicting short-term behaviors, but the coefficients achieved in the model do not likely represent the underlying dynamics of the neuron. This indicates that the shortcoming may be in the model we selected to fit to the data. The significant contribution of this chapter is demonstrating that UKF for a complex Hodgkin-Huxley style neuron can be implemented in real-time to predict voltage and used to control the voltage trace of the neuron. We will describe details of how to implement the UKF equations for Hodgkin-Huxley style to ensure stability of the algorithm. Finally, we will discuss limitations of this method for parameter estimation.

5.2 Methods

5.2.1 Mathematical model

The *GACell* is a Hodgkin-Huxley based model. Its entire description can be found in [17] and the equations are reproduced in the appendix B. Here we briefly present a few relevant details of the model. The current balance is given by

$$C\frac{d}{dt}V\left(t\right)=-\sum_{i=1}^{6}I_i+I_{App}. \tag{5.1}$$

The six currents in the model are the fast Na^+ current (I_{Na}), the persistent Na^+ current (I_{NaP}), the delayed rectifier K^+ current (I_{Kdr}), the A type K^+ current (I_{KA}), the slow K^+ current (I_{Kslow}), and the non-voltage dependent leakage current ($I_{leakage}$). I_{Na} and I_{Kdr} are responsible for the action potential generation [17], the I_{NaP} increases the excitability of the cell at voltages around threshold, which make the cell fire action potentials at low input currents, as observed in mammal cells [18]. I_{KA} reduces the gain of the neurons (firing rate *per* applied current), shaping the slope of the frequency *vs* current curve [18]. The frequency adaptation observed in excitatory cells is modeled with the I_{Kslow} [17]. The last current, ($I_{leakage}$) is a non-voltage dependent current that determines the passive properties of the cell [18] making the voltage to balance at the observed resting potential.

Each one of the voltage dependent currents in the *GACell* has the following structure [68]

$$I=g_{max}m^ah^b(V-E), \tag{5.2}$$

where g_{max} is the maximal conductance of the modeled current (mS/cm^2), m is the probability of the activation gate to be open, h is the probability of the inactivation gate (if present) to be open, a is the number of activation gates per channel, b is the number of inactivation gates per channel, V is the membrane voltage (mV) and E is the respective reverse potential (mV). The *GACell* has four voltage dependent gating variables: h, n, b and z which are the I_{Na} inactivation, I_{Kdr} activation, I_{KA} inactivation, and I_{Kslow} activation, respectively.

5.2.2 Unscented Kalman Filter formulation

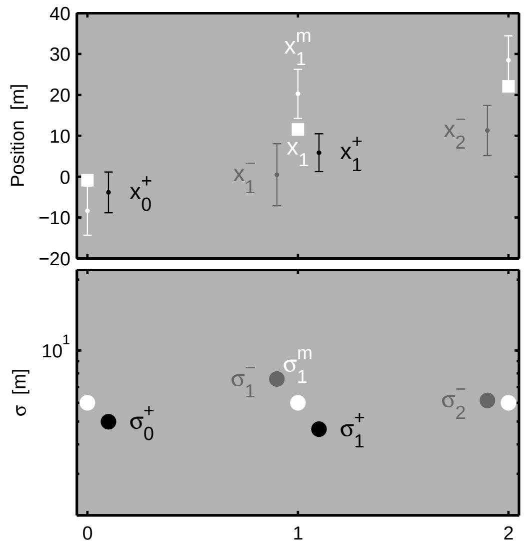
Here we describe the formulation of the UKF (as described in [65]) adapted to the *GACell*. We also describe the assumptions and changes that we implemented (appendix C reproduces the linear and unscented kalman filter formulation presented in [65]).

The UKF can be used to estimate states, parameters and inputs from noisy measurements. It fuses experimental data with a mathematical model in order to infer its results [63], [64], which is called *data assimilation* [69]. The UKF is an extension of the Linear Kalman Filter (LKF), which is used in linear systems where the distribution of possible values that each estimation can adopt is gaussian [65].

We will define "states" as variables that must be estimated by the model. When initiating the filter, an initial value and its respective variance is estimated for each state. The filter then propagates the mean and the variance for each state through the model. Then, it compares the estimation with the measured data to improve the estimation of the states. The weighting of the model and the data is determined based on the statistics of the estimations and the measurements. Ideally, after some iterations, the filter estimates converge to the real underlying values of the states.

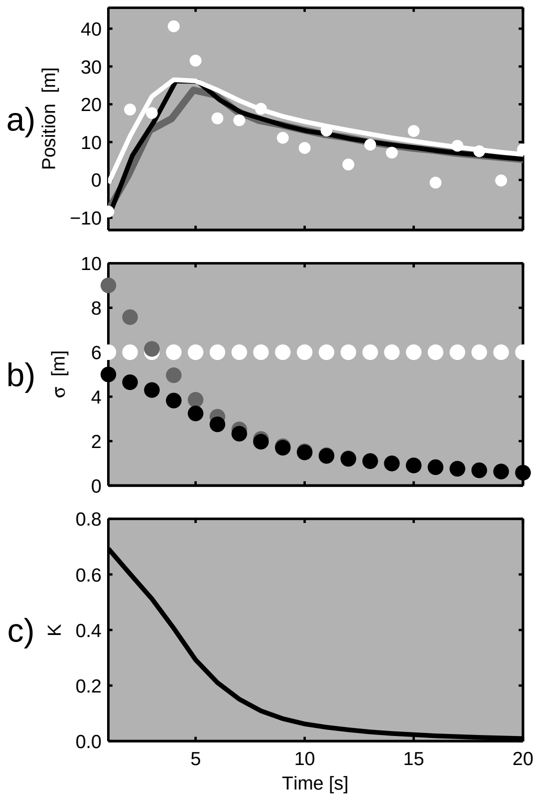
The linear KF fails in non-linear systems because normal distributions propagated through nonlinear systems are no longer normal. This problem is overcome propagating a few points per estimation (sigma points) with the same mean and variance of the corresponding estimation [63, 64]. The sigma points are propagated trough the system, its mean is calculated and the normal KF formulation can be used. If the mathematical model used is a good description of the process, the noise of the system is well characterized and the UKF is tuned correctly, the estimations converge rapidly to accurately predict the behavior of the process.

Each state to be estimated is initialized with an initial guess (x_0^+). The uncertainty around each state is assumed to be normally distributed with a standard deviation of σ_0^+. The variance of each state is grouped in a $n \times n$ matrix $\mathbf{P}$, where n is the number of states. Then, a set of $2n$ points are generated for each state. These points are called "sigma points". The mean and variance of each set of sigma points has the same mean and variance than the corresponding mean and variance of the states, as shown in figure 5.1. The sigma points are then propagated through the model. Their mean, after propagation, corresponds to the neuron's states at some point into the future and it is called the *a priori* estimate of the neuron's states, denoted by x_1^-. (The *a priori* estimation(s) of the system states are indicated by the superindex - and the *a posteriori* by the superindex +). The first sample of the neuron's voltage is taken, x_1^m. Then, the model prediction and the sampled data are combined using a weighted average to generate the *a posteriori* estimate of the neuron's state, x_1^+. If the measurement is noisy it will weigh the model more, if the model's predictions are inaccurate it will weigh the data more. This process is repeated iteratively, propagating estimations and variances through the model and updating them to minimize the prediction errors. Ideally, if the UKF method is used to estimate the parameters of the model they will converge to the optimal parameters that best fit the model tot he data, as shown in figure 5.2.



This figure illustrates the *a priori* prediction using just the model and the *a posteriori* estimate by weighted average of the measured data and the model prediction. At time 0, an initial guess x_0^+, with its respective standard deviation σ_0^+, is propagated through the equations that describe the system to generate an *a priori* prediction of the states x_1^- (with its respective standard deviation σ_1^-). Then, the measurement x_1^m and its standard deviation σ_1^+ are combined using a weighted average to generate the *a posteriori* estimation x_1^+, which is closer to the real value x_1. The process is iterated thorough all the sampled points.

Figure 5.1: Illustration of *a priori* and *a posteriori* estimations from Kalman filter.



Panel **a** shows the predicted values of the Kalman filter and the data over several samples. Dots indicates the measurements, white line is the actual state, gray line represent the *a priori* estimations and the black line the *a posteriori* estimations. Panel **b** shows the corresponding standard deviation of the estimations (color indicate corresponding values to state estimates shown in panel **a**). Panel **c** shows the Kalman gain, which determines the weighting used to average the *a priori* estimation and the data point in estimating the *a posteriori* value.

Figure 5.2: Predicted values, errors and gains of Kalman filter over time.

State space representation of the model

The state of the *GACell* neuron at a given time can be described as a vector $\overrightarrow{x}$:

$$\overrightarrow{x} = \begin{bmatrix} V & h & n & b & z \end{bmatrix}^T. \tag{5.3}$$

Then, the state space representation of the *GACell*, its inputs, noise sources and measurement can be written as:

$$\dot{\overrightarrow{x}} = \overrightarrow{f}\left(\overrightarrow{x}, \overrightarrow{u}, t\right) + \overrightarrow{w}, \tag{5.4a}$$
$$\overrightarrow{y} = h\left(\overrightarrow{x}, t\right) + \overrightarrow{v}, \tag{5.4b}$$
$$w \sim \left(0, \mathbf{Q}\right), \tag{5.4c}$$
$$v \sim \left(0, \mathbf{R}\right). \tag{5.4d}$$

$\overrightarrow{f}$ is the nonlinear function described by the *GACell* (6.1). Its dimension, d, is 5, and represents the number of states in the system. $\overrightarrow{u}$ is an unidimensional vector (scalar): the applied current to the cell (I_{app}). The output of the system is the measured voltage (scalar), hence y=x_1. $\dot{\overrightarrow{x}}$ is calculated as follows:

$$\dot{x_1} = u - I_{Na} - I_{NaP} - I_{Kdr} - I_{KA} - I_{K,slow} - I_L, \tag{5.5a}$$
$$\dot{x_2} = \left(h_\infty - x_2\right)/\tau_h, \tag{5.5b}$$
$$\dot{x_3} = \left(n_\infty - x_3\right)/\tau_n, \tag{5.5c}$$
$$\dot{x_4} = \left(b_\infty - x_4\right)/\tau_b, \tag{5.5d}$$
$$\dot{x_5} = \left(z_\infty - x_5\right)/\tau_z. \tag{5.5e}$$

The function for each ionic current are fully described in [17] and in the appendix B.

The noise in each state variable is described by the vector $\overrightarrow{w}$, which is determined by the covariance matrix **Q**, where all their elements are zero but the diagonals. Normally, this matrix is fixed. However, the noise in the states of the *GACell* depends on the firing activity. When neurons are close to threshold and the gating variables are beginning to open, the variance in the states increases greatly over that seen when neuron is at rest. Therefore we implemented a matrix where the noise in each state is estimated at each time step to account for the state dependence of the noise. The noise terms were calculated as follows:

$$\begin{aligned} Q_{1,1} &= (x_1 + 110mV)\, V_s, & (5.6a)\\ Q_{2,2} &= x_2\,(1 - x_2)\,/Q_s, & (5.6b)\\ Q_{3,3} &= x_3\,(1 - x_3)\,/Q_s, & (5.6c)\\ Q_{4,4} &= x_4\,(1 - x_4)\,/Q_s, & (5.6d)\\ Q_{5,5} &= x_5\,(1 - x_5)\,/Q_s. & (5.6e) \end{aligned}$$

5.6a was used to to calculate the variance of the voltage. In theory, the variance of the voltage should be dependent on the variance of the gating variables, but we made a simplifying equation that calculated the variance of the voltage a simply linearly increasing as the farther it is from the potassium reversal potential around $-110mV$ with a slope of $V_s = 0.2$. The variance of the gating variables was calculated assuming that the actual opening of the channels follows a binomial distribution where the probability of opening is determined by the gating variable and the number of channels. Therefore the variance of the current is $p(1 - p)/Q_s$. We arbitrarily set the $Q_s = 400$.

R is the variance of the measurement and we estimate that to be $R = 1\ mV^2$.

Setting initial conditions

We initialize the state of the filter with the following values:

$$\overrightarrow{x}_0^+ = \begin{bmatrix} y_0 & 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix}^T, \tag{5.7}$$

where y_0 is the first voltage measurement from the neuron recording.

We estimated the variance in the error of our initial states in the covariance matrix $\mathbf{P}$, where all their elements are zero and the diagonal is

$$\overrightarrow{D} = \begin{bmatrix} 16mV^2 & 0.1 & 0.1 & 0.1 & 0.1 \end{bmatrix}^T. \tag{5.8}$$

Setting $D_1 = 16mV^2$ means that we believe the actual voltage measured from the neuron has a standard deviation of 4 mV.

A priori estimations

Here we describe our implementation to calculate the *a priori* estimations.

Sigma points calculation. Sigma points are slight perturbations to the current estimate of the state that are each propagated through the system equations to predict the state at the next sampling time. For the UKF algorithm, $2n$ sigma points are required per estimate. The mean and variance of the sigma points are determined by the current estimate and the errors of these values. One sigma point is created for each state dimension by pushing the state plus or minus one standard deviation of the estimated error for that state. Therefore, to calculate the sigma points, we need an accurate estimate of the errors. The estimate of the variance of each parameter is stored in the matrix $\mathbf{P}$. However, numerical instabilities can occur if we calculate the square root of the matrix $\mathbf{P}$. Therefore, we force $\mathbf{P}$ to be symmetric by making $\mathbf{P} = \left(\mathbf{P} + \mathbf{P^T}\right)/2$ then calculate the square root using the Cholesky Matrix Square Root Algorithm, as suggested by Simon [65]. This function is implemented in RTXI for real-time computing using the C++ Template Numerical Toolkit for linear algebra (TNT) [70].

Given the matrix $\mathbf{P}$ we can now determine the sigma points $\overrightarrow{\hat{x}}_{k-1}^{(i)}$ as follows:

$$\overrightarrow{\hat{x}}_{k-1}^{(i)} = \overrightarrow{\hat{x}}_{k}^{-} + \overrightarrow{\tilde{x}}^{(i)},\ i = 1, \ldots, 2d, \tag{5.9}$$

where

$$\overrightarrow{\tilde{x}}^{(i)} = \left(\sqrt{nP_{k-1}^{+}}\right)_i^T,\ i = 1, \ldots, d, \tag{5.10}$$

and

$$\overrightarrow{\tilde{x}}^{(d+i)} = -\left(\sqrt{nP_{k-1}^{+}}\right)_i^T,\ i = 1, \ldots, d. \tag{5.11}$$

If the errors are large, some of the calculated sigma points could be outside the range of physiologically realistic parameters. For example, voltages must be within the range of the K^+ and Na^+ reversal potentials $[V_K; V_{Na}]$ and the gating variables must be within the range of $[0; 1]$. To ensure this, after estimating the sigma points we test that all are within these hard constraints [71]; values out of that range are set equal to the closest limit , $[V_K; V_{Na}]$ for voltage, or near the limits, $\left[1e^{-12}; 1 - 1e^{-12}\right]$ for the gating variables.

Sigma points propagation. Once the sigma points are calculated and set to be within physiological parameters, then they are propagated through the system equations

C.4:

$$\overrightarrow{x}_k^{(i)} = f\left(\overrightarrow{x}_{k-1}^{(i)}, \overrightarrow{u}_k, t_k\right). \tag{5.12}$$

Once the propagation is done, we again test and enforce the constraints.

***A priori* estimations.** The *a priori* estimation $\widehat{x}_k^-$ is obtained by the averaging of the $2d$ sigma points that were obtained from equation C.7:

$$\overrightarrow{x}_k^- = \frac{1}{2d}\sum_{i=1}^{2d} \overrightarrow{x}_k^{(i)} \tag{5.13}$$

***A priori* covariance estimation**. Next, we update the error covariance matrix P_k^-. This is done using the sigma points and the noise terms in the states, determined in the matrix $\mathbf{Q}$, which is updated on each time step according to the equation 5.6. The updated *a priori* covariance is calculated as follows:

$$P_k^- = \frac{1}{2d}\sum_{i=1}^{2d}\left[\left(\overrightarrow{x}_k^{(i)} - \overrightarrow{x}_k^-\right)\left(\overrightarrow{x}_k^{(i)} - \overrightarrow{x}_k^-\right)^T\right] + Q_{k-1} \tag{5.14}$$

A posteriori estimations.

The *a posteriori* state estimation is calculated by combining the new measured data and the model's prediction. The weighting matrix of the *a priori* estimate and the sampled data points is called the Kalman gain. The Kalman gain is determined by the estimated errors of the *a priori* and the measured data point. Once the *a priori* point is calculated, the last step is to update the *a posteriori* estimate of the variance of the states P_k^+.

Create sigma points. The sigma points for the *a posteriori* estimate are estimated in equations C.6a, C.6b and C.6c, but around the new *a priori* point $\overrightarrow{x}_k^{(i)}$.

Estimate the measurements. The measurements from the real neuron are limited to the observable variables, in this case the voltage. In the algorithm, all the states are estimated. To compare the model to the data, we must use a function that predicts the measurement from the states. In this case it is a linear function, a vector containing one for the voltage state and zeros for the hidden gating variables. There is a predicted measurement value $\widehat{y}_k^{(i)}$ corresponding to each sigma point x_1, calculated as follows:

$$\overrightarrow{y}_k^{(i)} = h\left(\overrightarrow{x}_k^{(i)}, t_k\right) \tag{5.15}$$

Combine the estimated measurements. The predicted measured value for the model is then calculated by averaging these points:

$$\overrightarrow{\hat{y}}_k = \frac{1}{2d} \sum_{i=1}^{2d} \overrightarrow{\hat{y}}_k^{(i)} \tag{5.16}$$

Estimate the error in the model. The covariance matrix P_y is an estimate of the error in the model's prediction. It is generated by calculating the variance of the predictions provided by each sigma point around the averaged prediction from all sigma points:

$$P_y = \frac{1}{2d} \sum_{i=1}^{2d} \left[\left(\overrightarrow{\hat{y}}_k^{(i)} - \overrightarrow{\hat{y}}_k \right) \left(\overrightarrow{\hat{y}}_k^{(i)} - \overrightarrow{\hat{y}}_k \right)^T \right] + R \tag{5.17}$$

Then, we estimate P_{xy} which determines the effect of each state perturbation to the predicted measurement. This matrix will help determine how to perturb the state estimate given the error between the predicted value and the measured value. It is calculated by measuring the covariance between the predicted measurements and the state perturbation:

$$P_{xy} = \frac{1}{2d} \sum_{i=1}^{2d} \left[\left(\overrightarrow{\hat{x}}_k^{(i)} - \overrightarrow{\hat{x}}_k^{-} \right) \left(\overrightarrow{\hat{y}}_k^{(i)} - \overrightarrow{\hat{y}}_k \right)^T \right] + R_k \tag{5.18}$$

Calculate *a posteriori* estimations. Given the estimate of the size of the errors in the states, we can calculate the Kalman gain from the ratio of covariances

$$K_k = P_{xy} P_y^{-1}. \tag{5.19}$$

The *a posteriori* estimate of the cell's voltage is calculated by a weighted average between the best estimate of the cell's voltage given the propagation of the model and averaging the sigma points $\overrightarrow{\hat{y}}$ and the measured value $\overrightarrow{y}_k$ using the Kalman gain:

$$\overrightarrow{\hat{x}}_k^{+} = \overrightarrow{\hat{x}}_k^{-} + K_k \left(\overrightarrow{y}_k - \overrightarrow{\hat{y}}_k \right) \tag{5.20}$$

Finally, the estimate of the variance for each state is updated given the estimate of the errors and the Kalman gain:

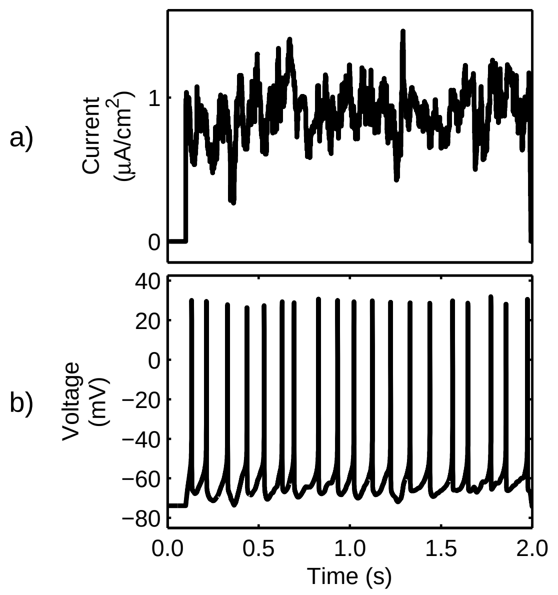
$$P_k^+ = P_k^- - K_k P_y K_k^T \tag{5.21}$$

5.3 Results

The UKF is used to fit the *GACell* to voltage measured from pyramidal neurons in cortical slices prepared from rats. The voltage was recorded using whole cell patch clamp techniques in current clamp mode. First, we show the response of the *GACell* to the same stimulus waveform used to stimulate the neuron. Next, we implement the UKF algorithm in the real-time dynamic clamp. Then we use the UKF to estimate the voltage and the gating variables by fitting the *GACell* model to the data in real time. Finally, we use the UKF prediction to control the neuron to follow a pre-defined waveform.

5.3.1 Data assimilation

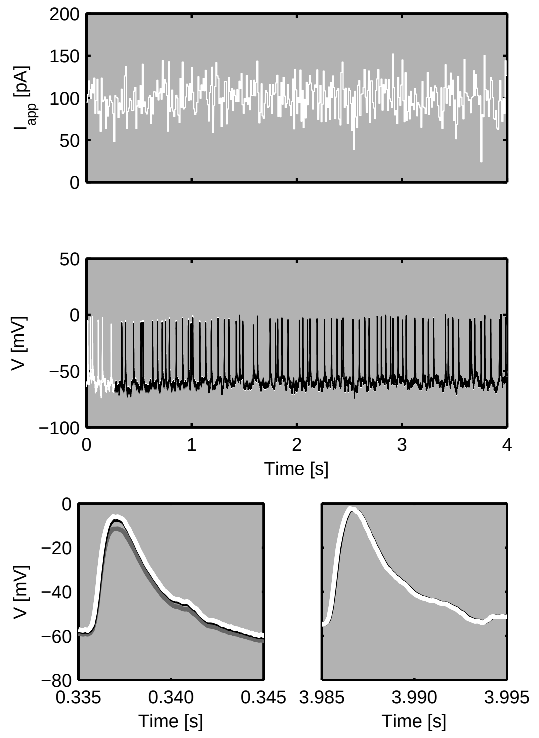
In figure 5.3 is plotted the response of the *GACell* to a random current waveform. As can be seen, the resting potential of the *GACell* is -74 mV and the peak voltage during an action potential is around 30 mV.



(a) Applied current and **(b)** the corresponding voltage trace.

Figure 5.3: *GACell* response to a random current.

Initially, the model's behavior is significantly different from the real neuron's. The white line of panel b on figure 5.4 shows the voltage trace recorded from a neuron. Notice that the resting potential of the neuron (-65 mV) is almost 10 mV more positive than the resting potential in the *GACell*. Furthermore, the recorded neuron has a voltage of $\sim$ 0 mV at the peak of each action potential, which is 30 mV more negative than the action potential peak in the *GACell*.



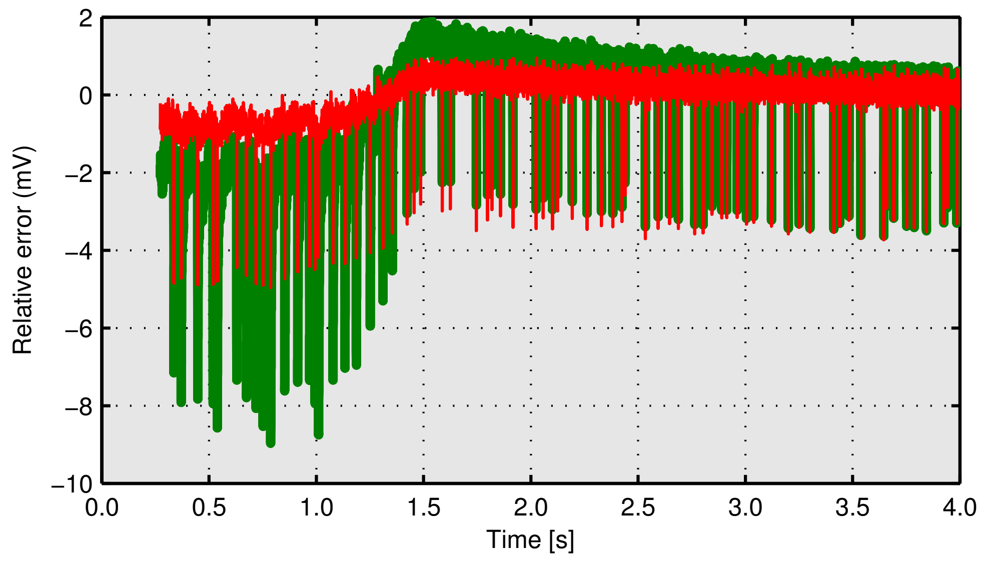
Top panel shows the applied current. Middle panel shows the corresponding voltage trace. White line represents experimental data, gray line shows the *a priori* estimation and black line corresponds to the *a posteriori* estimation. Lower panels are expansions of the first and last spike shown in middle panel.

Figure 5.4: UKF-based data assimilation in a neuron.

While the data was being acquired, at time $t = 200\ ms$ the UKF was started to estimate the states in the model and to make a prediction of the voltage for the next sampling time (*a priori* estimation). Then, at each sampling point, the error between the filter 's prediction using the *GACell* and measured voltage is used to improve the selection of sigma points and the voltage trace prediction.

The same figure 5.4 shows the *a priori* (gray line) and the *a posteriori* estimations (black line). Notice that the discrepancy between the estimations and the measurement decreases as the filter takes into account more experimental data. Within one second of data assimilation, the model is tuned so that *a priori* predictions are almost indistinguishable from the data.

Figure 5.5 shows the time evolution of the error of the *a priori* and *a posteriori* estimations when they are compared with the voltage measurement. Notice how the error on the estimations are reduced as more data is taken into account. The *a posteriori* estimation has a lower error than the *a priori* estimation. This phenomena is due to the fact that the *a posteriori* estimation is an improvement over the *a priori* estimation. This improvement is modulated by the error in the *a priori* estimation and the uncertainty associated to the estimations (matrices **P**, **Q** and **R**). In this figure is shown that after one second of data assimilation the error in the estimations reaches stable state (excluding estimations at spike time).



Errors of the *a priori* (green, thick line) and the *a posteriori* estimations (red, thin line).

Figure 5.5: Error of UKF estimation of neuron's voltage.

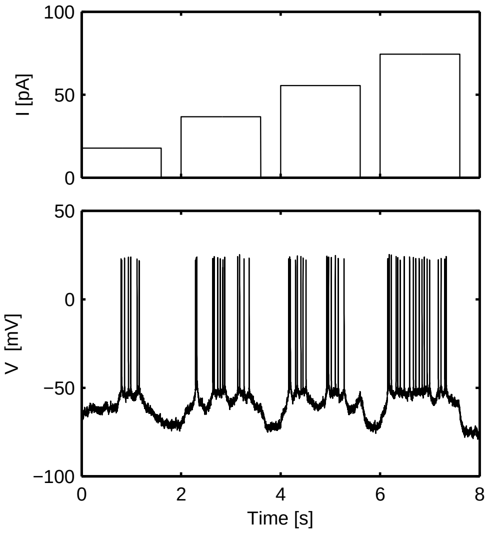
5.3.2 Model Reference Control

Given an accurate prediction of the neuron's voltage on the next time step, it becomes possible to make a control algorithm to make the neuron follow any arbitrary voltage trace. In this case, we selected the voltage trace calculated from the *GACell* as the reference signal ($Y_{target}(k)$). The control current is simply calculated from the difference between the expected voltage on the next time step and the target voltage. The difference is scaled by the cell's capacitance (C) and divided by the cell's resistance (R_m).

$$I(k) = \frac{Y_{target}(k) - \widehat{y}_k^{(i)}}{R_m} C \tag{5.22}$$

C was set equal to 1 $\mu F/cm^2$ and R can be changed by the user. We use R= 0.01.

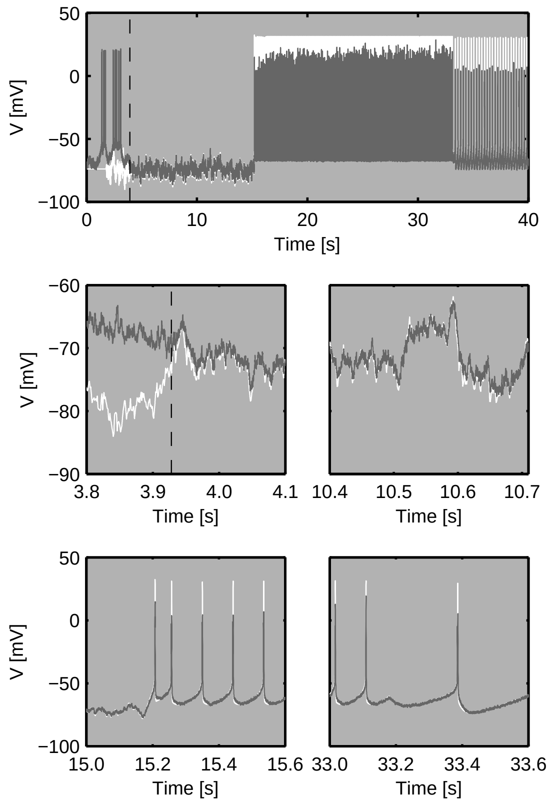
Figure 5.6 shows the response of a patched neuron when a series of applied current steps. This "open loop" response shows typical behaviors seen in excitatory cells: the firing rate is decreased after a few spikes, then it settles to an average firing rate with small variations in interspike intervals with an occasional missed spike.



Top panel shows the applied current steps to patched neuron. Bottom panel shows the neuron's voltage trace in response.

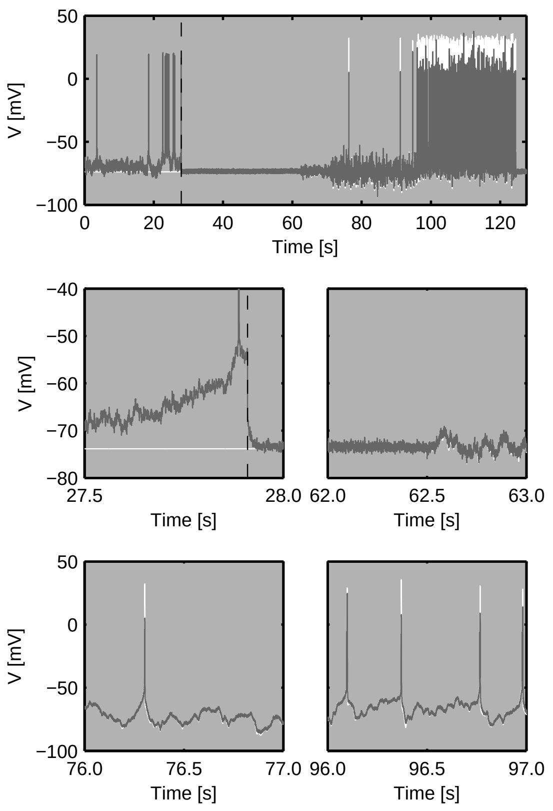
Figure 5.6: Open-loop behavior of neuron to be controlled.

The control goal is to make the neuron voltage follow the voltage trace generated by GACell in response to an arbitrary noisy input. You might think of this as turning the real neuron into the *GACell* To do this, first we simulate the *GACell* with and without noise and recorded the voltage trace. To control the neuron, we use the UKF to predict the voltage at the next sampling time and calculate the difference between the reference voltage and the predicted voltage to determine the control current. Figure 5.7 shows control around a noisy *GACell* response to current step, where it is firing nearly periodically. Figure 5.8 shows control around a noisy *GACell* being driven with a random stimulus waveform. Notice that in both cases the control algorithm successfully made the patched neuron follow the *GACell*'s voltage trace in sub-threshold and super-threshold activity. This approach was tested in approximately 20 cells.



White line shows the voltage trace from the *GACell* model, gray line is the patched neuron. Black dotted line indicates the moment at which the controller is switched on. Top panel shows entire recording. Lower panels show expansions around onset of control (middle left), sub-threshold behavior (middle right), super-threshold periodic firing (bottom left), and change in firing rate (bottom right).

Figure 5.7: Model reference control of an irregularly firing neuron to spike regularly.



Lines labels are same as in 5.7. *GACell* reference is driven with noise inputs of increasing amplitude over duration of experiment. Bottom panels are expansions around onset of control and three different amplitudes of noise driving the *GACell*.

Figure 5.8: Model reference control of an irregularly firing neuron to fire in a controlled irregular pattern.

5.4 Discussion

In this chapter we demonstrate that the UKF implemented in real-time can estimate the voltage and the hidden states of the neuron to accurately predict the voltage into the short time future. The prediction errors were greatest at the peak of the action potential where the derivative of the voltage is the highest. However, within a second, the UKF algorithm was able to substantially decrease the *a priori* prediction errors. The prediction accuracy was quite high despite the fact that the resting potential and action potential peaks in the free running model was significantly different than the real neuron's.

To demonstrate the validity of the prediction, the UKF algorithm was used for a model reference controller. The difference between the predicted voltage and a reference voltage waveform was used to calculate the control current. The UKF based reference control could successfully control the neuron's voltage to follow the voltage generated from a model neuron.

Ultimately, we would like to use the UKF to estimate the parameters of the *GA-Cell* that can accurately reproduce the behavior of the neuron. However, while these experiments demonstrate that we could predict the behavior short times into the future (on the order of milliseconds) we have not demonstrated that the UKF can be used to accurately generate computational models of the recorded neurons. By treating parameters as states, it is possible to use the UKF model to not only estimate the states of the system, but also the parameters as well. Using this approach, we have tried to fit a few representative parameters of the *GACell*, such as the maximum conductance of the ionic currents present in the model to improve the fit and predicted model. We do find that estimating the parameters along with the states using the UKF does increase prediction accuracy. However, we do not believe that these parameters accurately reflect the neuron's actual values. This is because after letting the model converge to a reasonable prediction error, we use the estimated parameters to run a simulation of the *GACell*, the model does not always behave like the neuron. For example, when parameter are fit to a regularly spiking neuron, the model sometimes produces bursts of action potentials or even lack action potentials entirely. This indicates that prediction error can be minimized by altering parameters in ways that do not accurately reflect the neuron's parameters. Of course, a model that could reproduce the neuron's dynamics when free-running would probably be the best model for prediction, this indicates that it is not a unique solution.

There are intrinsic limitations fitting models using only the potential measured from a neuron. Specifically, the voltage recording of the neuron is limited to the soma, and all the spatio-temporal information is lost. Hence, the use of a single compartment model will never be able to perfectly reconstruct the dynamics of a spatially extended cell for which a multi-compartment model is required [72]. The use of a different model, reducing the number of parameters used to fit to focus on the most important ones, and the use of stimulus waveforms optimized to evoke complex responses that constrain the models that can reproduce the behaviors are approaches that may be used to increase accuracy of parameter estimation from data.

Despite the inaccuracies between the fit model and the neuron's actual parameters, a fit model may still be useful for classifying neurons and to detect changes in dynamics of neuron over the duration of an experiment and perhaps across neurons. Simultaneous estimation of all the neuron's currents from a single recording may provide the ability to understand how ionic currents are maintained in proportion to one another [30]. The ability to track parameters of a neuron over time may provide insight into the basic mechanisms of how homeostatic changes in ionic currents play a role in diseases like epilepsy, Parkinson's disease, and schizophrenia. The control algorithm presented here may be useful to make a neuron behave in a particular pre-described pattern that may be useful in determining the relationship between a neuron's behavior and its homeostatic control of its ion channel densities.

Chapter 6

Optimal Stimulus Design

In computational neuroscience it is often a need to fit a computational model to data. This problem is known as system identification. There has been much focus on different methods for fitting models to data; but, just as important is the stimulus waveform used to stimulate the neuron to generate the dataset used in fitting the model. An ideal stimulus waveform will drive the neuron through its repertoire of behaviors in a short period of time. The more complex the behavior, the more constrained the model will be in determining the model parameters that can reproduce the behavior, and hopefully the more accurate the resulting model. In this chapter I propose a method for quantifying the complexity of the neuron's response, by calculating the percentage of state space covered by the neuron in response to the stimulus waveform. Several stimulus waveforms, steps, ramps, chirps and an Ornstein-Uhlenbeck (OU) process are compared. Neuron responses in voltage clamp and current clamp were simulated using the computational neuron model proposed by Golomb and Amatai [17]. Parameters of the proposed waveforms were optimized to maximize the percent state space covered in a two second stimulus. To measure how effective the resulting data is for parameter estimation, we use an Unscented Kalman Filter (UKF) to fit the model to the data, starting with random initial conditions for each of the estimated parameter. Using the step response, the UKF could only determine the parameters within 20% of the actual value, indicating that the model was under-constrained by the data. Using an optimized OU stimulus waveform, the UKF could determine the parameters of the model within 1%, even under realistic noisy conditions. By optimizing stimulus waveforms to maximize the the percent state space covered by a model, our goal is to generate data from neurons that will constrain model parameters and improve accuracy of the fit model.

6.1 Introduction

In whole-cell patch clamp recordings, it is possible to apply current or voltage waveforms to the neuron and measure the neuron's response. In this chapter I propose a method to design stimulus waveforms suited for fitting models to data from patch clamp recordings.

If neuronal dynamics were linear and time invariant (LTI), it would be possible to use tools developed in the engineering discipline for fitting models to data, known as system identification. However, neurons are very nonlinear and therefore we use very nonlinear models, such as Hodgkin-Huxley style models. These nonlinearities make it difficult to use standard system identification tools to fit models to data.

In system identification the goal is to accurately model the output of the system given a known input. LTI systems can be fully characterized by their impulse response, the system's response to a delta function, an infinitely high pulse with area one and width of zero. Given the impulse response, the system's response to any input waveform can be predicted. However, the delta function is a theoretical stimulus and experimentally it can only be approximated. Experimentally there are several stimulus waveforms that are used to characterize LTI systems, such as rectangular pulse, step input, frequency chirp and white noise.

For nonlinear systems, pulses and step inputs may characterize the systems behavior for small perturbations, where the response is locally linear, but they are not sufficient to characterize the full behavior of the system. By definition, a system is nonlinear when doubling the input does not double the output. Neurons are highly nonlinear systems. Subthreshold stimuli may result in nearly linear behaviors, but doubling a subtheshold stimulus current pulse may bring the neuron above threshold resulting in an action potential, a response that is not a scaled subthreshold response. In a carryover from traditional engineering approaches, neurons are often characterized by applying a series of step responses. This is used to measure spike rate as a function of current (F-I curve), or to characterize the opening and closing of ion channels. To characterize some ion channels more complex step waveforms are used, such as varying the holding voltage prior to a step to a target voltage. An advantage to a pulse and step inputs is that the response can be locked to an event, making it easy to characterize the result with a few statistics, such as as spike count, onset time of first spike and spike rate adaptation. However, these stimulus waveforms do not elicit complex enough results to constrain the model, which may result in many parameter combinations that can be used to reproduce the data [21]. The more complex the neuron's behavior the more the data can help constrain the models.

A neuron's step response does not contain a lot of information. After the first few spikes, the neuron usually settles down to a periodic behavior and little new information is generated by recording for longer intervals [73]. Ideally an experiment will continue to provide more information the neuron the longer the neuron is recorded. Complex stimuli can provide much more information about the neuron's dynamics, which can be useful in characterizing the neuron [74, 75, 76, 28, 77, 78]. A neuron's response to a Gaussian white noise stimulus is much richer. However, while the response to white noise is more complicated, is it the optimal stimulus for characterizing the neuron? White noise is not a naturalistic stimulus rarely eliciting behaviors that could be common with more naturalistic conditions. For example, white noise may not produce the spike rate variability observed *in-vivo* where long pauses in the firing may occur. During these long pauses, the effects of slow currents begin to emerge. Presumably the dynamics of neurons are tuned to maximize subtle differences in the dynamics they see normally. Therefore, subtle differences in responses to naturalistic inputs may be important in characterizing the neuron's behavior that may be seen rarely in response to Gaussian white noise.

Essential to a successful fit is a good data set, a measure to compare the two, and an optimization algorithm to adjust parameters of the model until the model fits the data. Optimization my be done in many ways, such as *brute-force* approach, synchronization based methods (SBM) [26, 27], Unscented Kalman Filters (UKF) [66, 67], or simulated annealing [24]. The focus of this chapter is how to generate the data to facilitate the optimization.

We posit that the best stimulus for fitting a model is one that maximizes the behavioral repertoire of the neuron in a short period of time. Not knowing the dynamics of the neuron, we propose to optimize stimulus waveforms on a computational model whose dynamics roughly approximate that of the neuron of interest. The advantage of the computational model over the real neuron is that in the model it is possible to measure the effect of the stimulus on all the hidden states of the neuron, such as the activation and inactivation of the different ionic currents while in a real neuron the response can only be measured from the voltage or current. Our goal in designing a stimulus waveform is to find a stimulus that will evoke all the possible combinations of opening, closing and inactivation of the currents within the duration of the experiment. In other words, we want a stimulus that will push the neuron through its entire state space in a short period of time. Realistically it is not possible to elicit every combination of channel states, and certainly not possible without killing the neuron, but a stimulus that covers more state space presumably will characterize the neuron's behaviors better than one that causes it to cover less state space. We propose that the percentage of state space visited by the model over the duration of the stimulus can be used as a measure to compare and optimize stimulus waveforms. If a stimulus waveform has parameters, for example standard deviation and smoothness, the stimulus can be optimized to maximize the percentage of state space visited for a finite power.

Simulations in this chapter will use a a single chamber model by Golomb and Amatai [17] (*GACell*), a model of an excitatory regular-spiking neocortical neuron, but the theory could be applied to any conductance based model. We applied several different waveforms, steps, ramps, chirps and white noise, and Ornstein-Uhlenbeck processes. Each was applied to the model as either a voltage control or current control signal, simulating either voltage or current clamp experiments and the percent state space covered measured. Each stimulus was optimized to maximize the percentage of state space covered by the GACell's response to a two second stimulus waveform. The resulting response of the neuron was then used by an UKF algorithm to fit coefficients, with random initial conditions, of the model to the data. Success of the different stimulus waveforms were judged by the accuracy of the coefficients compared to original parameters starting from randomized initial conditions.

6.2 Methods

6.2.1 GACell

The Golomb and Amatai (GACell) is a Hodgkin-Huxley based model. It will be described here in brief and a full description can be found in [17]. The current balance equation is as follows:

$$C\frac{d}{dt}V = -I_{Na} - I_{NaP} - I_{Kdr} - I_{KA} - I_{Kslow} - I_{leak} + I_{App}. \tag{6.1}$$

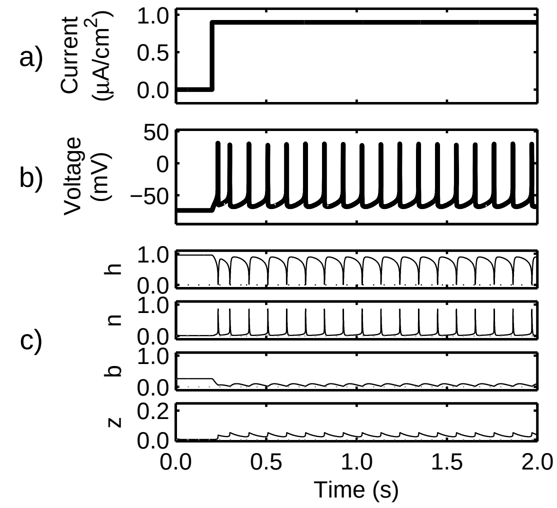
The six currents in the model are the fast Na^+ current (I_{Na}), the persistent Na^+ current (I_{NaP}), the delayed rectifier K^+ current (I_{Kdr}), the A type K^+ current (I_{KA}), the slow K^+ current (I_{Kslow}), and the non-voltage dependent leak current (I_{leak}). I_{Na} and I_{Kdr} are responsible for the action potential generation [17], the I_{NaP} increases the excitability of the cell at voltages around threshold, which makes the cell fire action potentials at low input currents, as observed in mammalian neurons [18]. I_{KA} reduces the gain of the neurons (firing rate *per* applied current), shaping the slope of the frequency *vs* current curve [18]. The frequency adaptation observed in excitatory cells is modeled with the I_{Kslow} [17]. The last current, (I_{leak}) is a non-voltage dependent current that determines the passive properties of the cell [18] balancing the other currents so that the model rests at the neuron's observed resting potential.

Each one of the voltage dependent currents in the GACell has the following structure [68]

$$I = g_{max}m^a h^b(V - E), \tag{6.2}$$

where g_{max} is the maximal conductance of the modeled current (mS/cm^2), m is the probability that the activation gate is open, h is the probability of the inactivation gate (if present) is open, a is the number of activation gates per channel, b is the number of inactivation gates per channel, V is the membrane voltage in (mV) and E is the reverse potential in (mV) of the ion that the channel passes. The GACell has four voltage dependent gating variables: h, n, b and z which are the I_{Na} inactivation, I_{Kdr} activation, I_{KA} inactivation, and I_{Kslow} activation, respectively. Of the 5 states, the voltage is the only observable one in a real neuron, the others are hidden states of the system.

In current clamp simulations, the GACell (6.1) is solved at a given holding voltage V_m to determine the current I_{App} required to hold the cell at that voltage. The GACell is integrated using a time step of 200 μs to emulate data acquisition from a real cell sampled at five kHz, as shown in figure 6.1.



When a constant current of approximate 0.9 $\mu A/cm^2$ is applied to the GACell (panel **a**), it will fire at 100 ms, as indicated in panel **b**. Panel **c** shows the time evolution of the gating variables.

Figure 6.1: GACell in current clamp, constant current.

6.2.2 Calculating the percentage of state space covered

To predict the efficacy of the stimulus waveform, the percentage of the state space covered by the model in response to the stimulus waveform is measured. We divide the state space into voxels and measure how many states the neuron visits. Each voxel represents a unique state of the neuron. Theoretically, once a neuron has visited a particular state, revisiting it following the same path does not contribute new knowledge about the neuron's dynamics. Voxels are generated by dividing each hidden state, *i. e.* the four gating variables, into 30 bins each. This results in $30^4 = 810000$ voxels. The voltage is implicitly included, because it is a function of the four gating variables. The percent state space covered is calculated as the number of unique voxels visited divided by the total number of voxels.

6.2.3 Stimulus waveforms

We applied several different waveforms: steps, ramps, sinusoidal linear chirps and Ornstein-Uhlenbeck (OU) processes. For the chirps and the OU processes free parameters were determined by optimizing the stimulus waveform to maximize the percentage of state space covered within a two second stimulus. Percent state space covered from each stimulus is reported.

Step

A constant current applied to the GACell. At 0.9 $\mu A/cm^2$ the GACell fires at 10 Hz (inter-spike interval of 100 ms).

Ramp

Current ramped from -0.2 to 1.5 $\mu A/cm^2$ over two seconds.

Chirp

A sinusoidal stimulus waveform in which the frequency ramps over time. Because hyperpolarizing and depolarizing inputs to the neuron have different effects, we scaled the stimulus amplitude for the positive phases and negative phases differently. At each discrete time k, the applied input is calculated as:

$$y_k = \begin{cases} A_1 \sin(2\pi f_k t) & \text{for } \sin(2\pi f_k t) > 0 \\ A_2 \sin(2\pi f_k t) & \text{for } \sin(2\pi f_k t) < 0 \end{cases} \tag{6.3}$$

where A_1 and A_2 are the gains for positive or negative phases, respectively (in $\mu A/cm^2$), f_k is the instantaneous frequency (in Hz) and t is the time (in s). The instantaneous frequency is calculated as follows:

$$f_k = \frac{k-1}{N} \times f_{max}. \tag{6.4}$$

In two seconds of data sampled at 200 μsec intervals, there are $N = 10001$ samples. The optimized variables were A_1, A_2 and f_{max}.

Ornstein-Uhlenbeck process

White noise stimuli are often used to characterized LTI systems. In theory, these stimuli would be ideal for characterization of non-linear systems, but the waveforms are not physiologically realistic and may take a very long time to elicit realistic behaviors. Instead, we used a filtered white noise signal known as an Ornstein-Uhlenbeck (OU) process, that is history dependent. The OU process is calculated as follows:

$$\mathrm{d}x_t = \theta(\mu - x_t)\mathrm{d}t + \sigma \mathrm{d}W_t, \tag{6.5}$$

where μ represents the long term mean of the process x_t, θ is the weight given to the history, and σ represents amplitude of the Gaussian white noise process added at each time step, W_t.

The variance of x_t is obtained from

$$\mathrm{var}(x_t) = \frac{\sigma^2}{2\theta}. \tag{6.6}$$

As the resting potential is not equidistant between the sodium (V_{Na}) and potassium (V_K) reversal potentials, we decided to use a different variance of the OU process above and below the resting potential. To do this, we calculated two values of θ

$$\theta_1 = \sigma^2 / \left(2\left(\left(R_p - V_K\right)/w\right)^2\right), \tag{6.7}$$

$$\theta_2 = \sigma^2 / \left(2\left(\left(R_p - V_{Na}\right)/w\right)^2\right). \tag{6.8}$$

This modified OU process normalizes the standard deviation by the distance from the resting potential to the V_{Na} and V_K and scaled by the factor w. Finally, using equation 6.5, the holding voltage is calculated:

$$\mathrm{d}V = \begin{cases} \theta_1(R_p - V)\mathrm{d}t + \sigma\mathrm{d}W_t & \text{for } V < R_p \\ -(n+1)/2 & \text{for } V \geq R_p \end{cases} \tag{6.9}$$

Here, we optimized the parameters R_p, σ, and w. We set V_{Na}= 50 mV and V_K= -100 mV.

6.2.4 Optimization algorithm

Each one of the applied inputs has one or more free parameters that can be optimized to maximize the percentage of space covered by the gating variables. For each input, we chose initial values for each parameter at random. The stimulus is applied to the model, and the percentage of state space covered over the duration of the simulation is calculated. A gradient ascent method using Matlab's *fminsearch* algorithm was used to optimized the parameters to maximize the percentage of space covered [53].

6.2.5 Unscented Kalman Filter implementation

To determine the efficacy of the stimulus for determining model parameters, we fit the GACell parameters to best fit the data. Of course, because the GACell was used to generate the model, the parameters were known, but at the initiation of each fit, we perturb the model parameters randomly. We can then compare the error in the estimated parameters to the real parameters over the duration of the simulated experiment. The model was fit to the data using the UKF. Details of the UKF method for fitting models are presented in [65].

In brief, the UKF fuses experimental measurements with a mathematical model to estimate states, parameters, and inputs from noisy time series measurements of continuous non-linear dynamical systems. The parameters can be time-variant and the model can also have stochastic terms [66]. UKF is an extension of the linear Kalman Filter (KF). The KF is the optimal estimator for states, parameters, and inputs in linear dynamical systems where the possible value for each parameter to be estimated needs to follow a normal distribution [65]. The KF recursively propagates the mean and the variance for each estimation through the set of differential equations that describe the system and after some iterations the filter converges to the real value for each estimation. At the outset of each fit, the filter propagates the initial conditions through and the stimulus through the model to obtain an *a priori estimation* of the neuron's response. Then, the *a posteriori estimation* of the neuron's response is generated using a weighted average of the sampled data and the *a priori estimation*. The weighting is determined by the Kalman gain, which is calculated using the statistics of the prediction and measurement noise properties. This *a posteriori estimation* is used as the initial condition for the next sampling time and it is propagated through the system again with the next stimulus input. As the propagation of normal distributions through nonlinear systems are no longer normal, the KF can not be directly applied to nonlinear systems. Julier and Uhlmann [63, 64] came up with an ingenious approach, calculating a few points, called *sigma points*, per state whose has the same mean and variance as the originating state. Before the calculation of the *a priori* or *a posteriori* estimations, *sigma points* are obtained for each state. Then, each *sigma point* is propogated through the equations. The resulting points and their used to calculate the mean and variance of the *a priori* and *a posteriori* estimations. This approach is called Unscented Kalman Filter (UKF) and its modification to the KF allows it to handle nonlinear systems. If the parameters of the UKF algorithm are properly tuned, the estimates of the parameters and states converge to the real values. For this approach to be successful, the model fit must be able to reproduce the dynamics of the neuron and the noise amplitude in each state and measurement noise measurement must be known [65]. In these simulations, these conditions are possible to meet, but may not be possible using real world data.

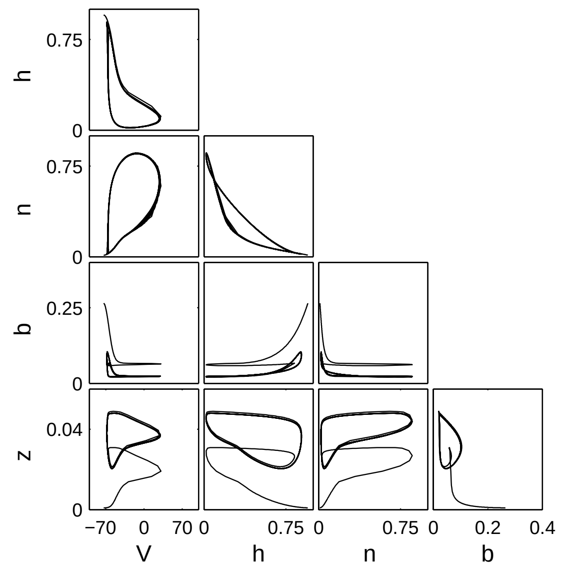
Using the UKF the five states (the voltage and the four gating variables) were estimated along with the maximum conductances for each one of the six currents present in the model (I_{Na}, I_{NaP}, I_{Kdr}, I_{KA}, I_{Kslow}, and I_{leak}, which default values are 24, 0.07, 3, 1.4, 1, and 0.02 mS/cm^2, respectively). The initial values for each maximum conductance was randomly set at either plus or minus 50 % of its default value.

6.3 Results

Current and voltage steps were first applied to the neuron model to measure the baseline state space percentage covered. Then, more complex waveforms were optimized on the model, and then applied, measuring the maximum percent state space covered. Finally, UKF was used to fit the GACell with random initial conditions for maximum currents and states were fit to the data and final error in estimated parameters were measured.

6.3.1 State space coverage by step response

When a constant current is applied to the GACell, there is a transient in the cell's firing rate over the first few spikes and then the model settles to a constant rate. The model reproduces spike rate adaptation seen in excitatory mammalian neurons (see figure 6.1). In figure 6.2 is shown the state space trajectory of the neuron over the duration of the simulation. It can be seen that the trace at the initiation of the simulation is different from the steady state, which means that the model covers new state space ground for the first few spikes, but then as it settles into a steady state behavior, no new state space is being covered. Each dimension was divided into 30 bins, resulting in 810,000 voxels. Step responses measured in both current clamp and voltage clamp, resulted in only 0.05 % of state space covered. As can be seen, the current step waveform generates new information over the first few spikes, but after the neuron settles into a steady state behavior no new information is gained by recording for longer periods.



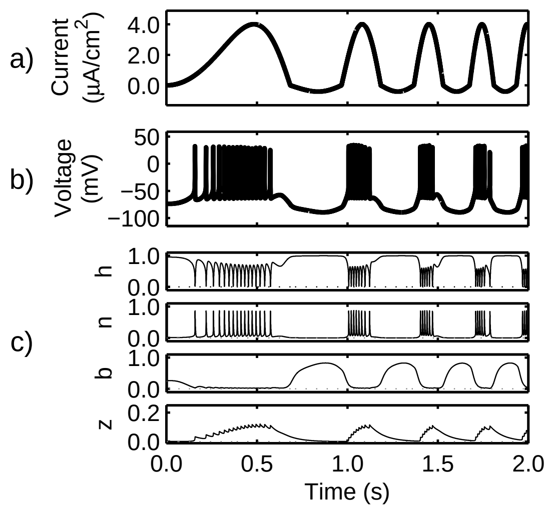
Each one of the 5 states (the voltage trace V and the four gating variables h, n, b, and z) are plotted as a function of each other. This state space represents the time evolution of the GACell when a constant current is applied to it, as shown in figure 6.1.

Figure 6.2: State spaces of the GACell, current clamp, constant current.

6.3.2 Optimization of stimulus waveforms

Next, a chirp-like function and an Ornstein-Uhlenbeck process were optimized to maximize the percent state space covered.

The optimized parameters for the chirp waveform in current clamp are $f_{max} = 2.1374$ Hz, $A_1 = 4.0054\ \mu A/cm^2$, $A_2 = -0.4087\ \mu A/cm^2$. The stimulus waveform and neuron's response, represented by traces of the voltage and gating variables, is shown in figure 6.3. The state space plot representation of the neuron's response, shown by plotting gating variables b vs z is shown on the right hand side of figure 6.5. The chirp response covered 0.2 % in 2 seconds. This represents a 5 fold increase in state space covered when compared to the step response.

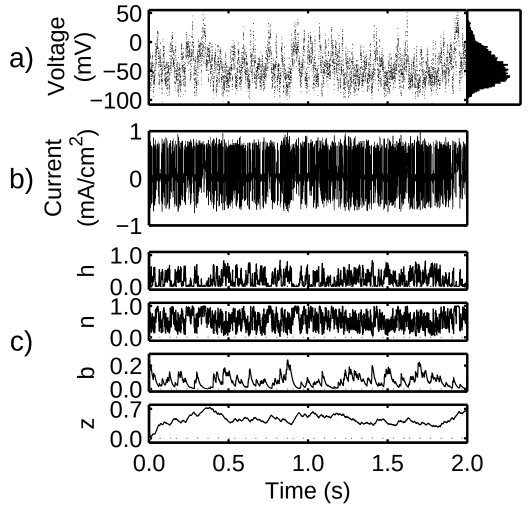


When the optimized chirp like current (panel **a**) is applied to the GACell, its response is shown in panel **b**. Panel **c** shows the gating variables over time.

Figure 6.3: GACell in current clamp, optimized chirp stimulus.

The Ornstein-Uhlenbeck process stimulus was optimized for the parameters determining the resting potential R_p, the history dependence σ, and the normalizing factor w. In voltage clamp, the optimized parameters were: $Rp = -61.8808$ mV, $\sigma = 7.9406$, and $w = 3.1579$. The stimulus waveform, amplitude distribution, voltage response and gating variables are shown in figure 6.4. The state space plot for the response to the OU stimulus is shown on the left hand side of figure 6.5. The stimulus applied in voltage clamp resulted in 1.13 % of the space covered. This is 23 times the coverage of the original voltage step response.

Of the signals optimized and tested, in current clamp the chirp like function results in



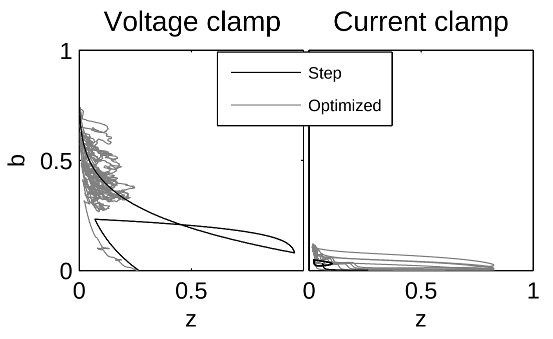
Voltage clamp simulation of GACell where the holding voltage is determined by the optimized Ornstein-Uhlenbeck process (panel **a**). The holding current is shown in panel **b**. Panel **c** shows the time course of the gating variables.

Figure 6.4: GACell in voltage clamp, optimized OU stimulus.

the highest state space covered and in voltage clamp the Ornstein-Uhlenbeck produced the greatest state space coverage.

6.3.3 Accuracy of parameter estimation dependent on stimulus waveform used

We hypothesize that a stimulus that increases the percentage of state-space covered will result in better fit of the model parameters. To test the efficacy of the optimized signals, we use a UKF to estimate the 5 states (voltage and the 4 hidden gating variables h, n, b, and z) and the maximum conductance of each of the ionic currents. There are 6 ionic currents in the model, and we randomly select the initial values for the maximum conductance as either plus or minus 50 % of the actual value. This results in 64 possible "worst case" conditions. To make the the modeling more realistic, white noise was added to the stimulus waveform resulting in interspike variability similar to that observed in experimental data, as shown in figure 6.6.



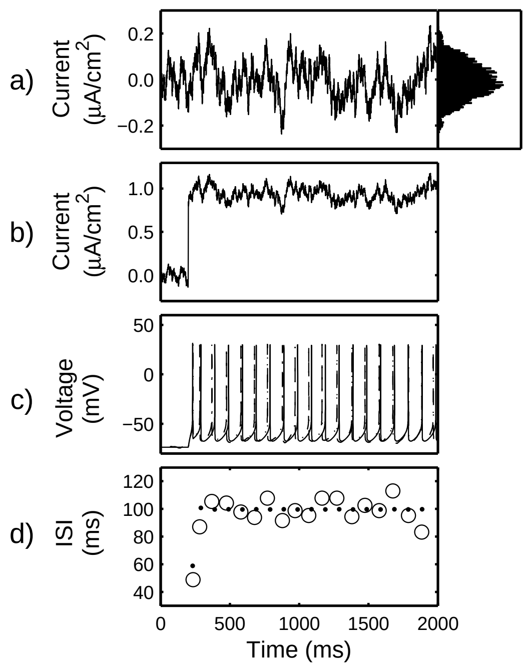
The state space response, represented by plotting the theI_{KA} inactivation gating variable *b* and the I_{Kslow} activation gating variable *z*. Response to the Ornstein-Uhlenbeck process is shown on left, covers 1.13 % of state space, while the response to the chirp, shown on right, only covers 0.2 % of state space. These representations of state space can be compared to lower right panel on figure 6.2.

Figure 6.5: State space representation of GACell response to stimulus inputs.

Figures 6.7, 6.8, and 6.9 show the time evolution of the state and parameter estimations from the GACell using the UKF. Parameters g_{NaP} and g_{leak} were selected as representative of the fit parameters. The g_{NaP} parameter was the fastest parameter to converge and g_{leak} was the slowest, as shown in (6.7). Two factors are used to judge the quality of the signals for estimating the parameters: the final error of the estimated parameters and the time to converge to the within ±5% of the final estimated value.

Figure 6.10 summarizes the parameter fitting results, showing the final error and time to final value for optimized stimuli applied in voltage and current clamp. The worst final error and slowest time to final value resulted from the GACell response to the current step.

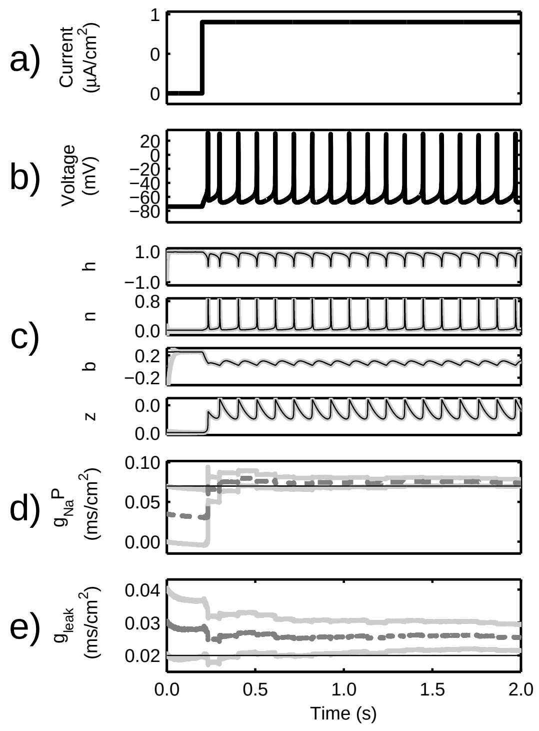
At the initialization, the error for each free parameter was set randomly at ±50% of the default value. The UKF parameter fit using the neuron's response to a 2 second current step recorded in current clamp resulted in a final error of 20%. In voltage clam the step response showed no benefit to the UKF algorithm to fit parameters over the initial value. The optimized chirp-like waveform in current clamp resulted in parameter estimations within 5% of the actual vlaue in less than a second. The best fits were achieved with the optimized OU in voltage clamp resulting in less than a 1% error in less than 0.2 seconds.



A noisy signal with zero mean **(a)** is added to a current step **(b)** to simulate variability in the interspike interval observed experimentally. In **(c)**, the continuous line is a plot of the neuron's step response in the noiseless scenario, and dashed line shows the response to the noisy input. Shown in **(d)** are the corresponding interspike intervals, where (●) corresponds to the noiseless current and (○) for the noisy input.

Figure 6.6: Noise added to stimulus to simulate realistic variability in neuronal response.

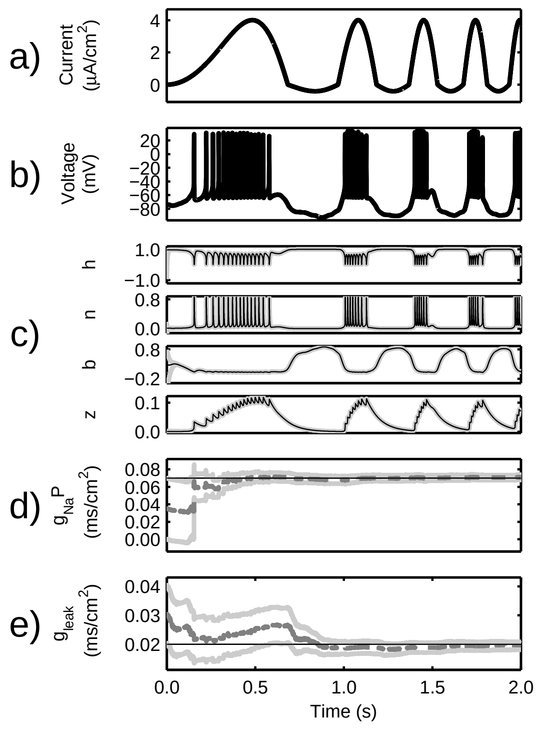
In figure 6.11 is plotted the cumulative percentage of state space covered as a function of time for the different stimulus waveforms. In current clamp, the step response shows a rapid increase in information within the first couple of spikes and then asymptotes as it reaches steady state. The ramp, taking a long time to elicit the first action potential, results in a slow rise in state space coverage, but continues to rise as the firing rate increases. With the chirp input, the state space coverage increases dramatically with each burst (seen as the rapid rises) followed by slow increases during the hyperpolarization phase. Notably, the slow variables are fit during the these hyperpolarization phases, therefore the the information increases slowly but significantly in the first two hyperpolarizing cycles, but later cycles do not contribute much new information. In



Current step **(a)** is applied to the GACell. The UKF was used to estimate the voltage trace **(b)**, and the hidden gating variables **(c)**, as well as the maximum conductance parameters g_{NaP} **(d)** and g_{leak} **(e)**, The thin-black line represents the real value of the parameter, dark gray dashed line is the estimated parameter value, and the light gray line represents the confidence bounds at 1 σ.

Figure 6.7: Parameter estimation of GACell in response to current step.

later phases, the information is gained during the depolarizing phase, as the frequency of the neuron is pushed higher, but this will probably asymptote at a value just above 0.2% of the state space. The OU stimulation in voltage clamp shows a steady increase in state space coverage over time. In two seconds we have covered nearly 1.2% of the state space. We expect that this would asymptote at some value, but well above 1.2%, indicating that the longer this stimulus waveform can be applied, the more information can be gathered about the model's dynamics.



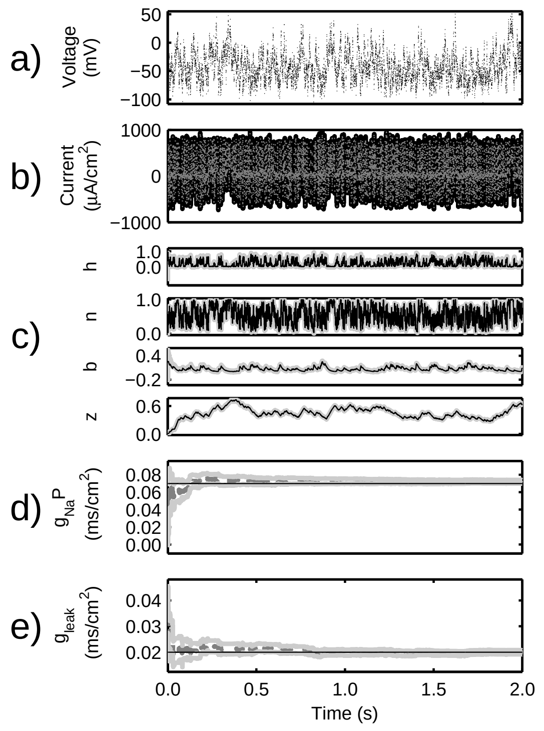
Panels are labeled as indicated in Figure 6.7

Figure 6.8: Parameter estimation of GACell in response to chirp-like stimulus.

6.4 Discussion

Traditionally, neurons have been characterized by applying current or voltage steps and measuring their response. The response to a current step is easily characterizable using measures such as firing rate, accommodation rate, and percent change in firing rate from onset to end. When fitting a model to the data, the simpler the response the less constrained the model will be in fitting the data. This results in a larger parameter space considered a good fit to the data. The data therefore may not not help achieve uniqueness of the model.

In this chapter we hypothesize that a stimulus waveform that elicits complex patterns of behaviors will be better for fitting parameters than a simple one. To quantify the "complexity" of the response, we measured how much state space is covered by the neuron over the duration of the stimulus. Using this methodology, we could optimize

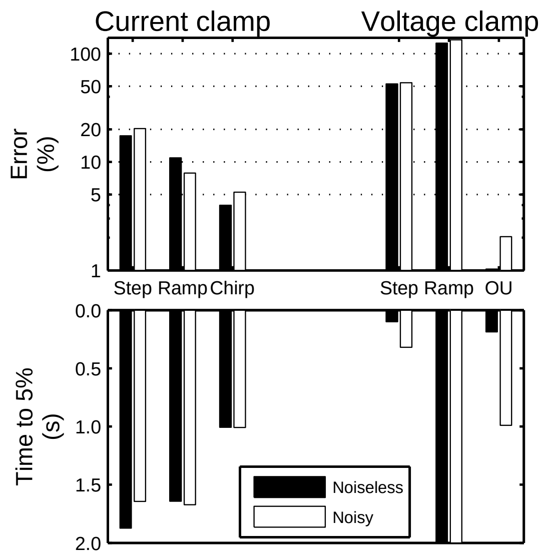


Panels are labeled as indicated in Figure 6.7

Figure 6.9: Parameter estimation of GACell in response to optimized Ornstein-Uhlenbeck process used in voltage clamp.

stimulus waveforms to maximize state space coverage. We then fit models to data using an Unscented Kalman Filter using the original model used to generate the data but with initial guesses that were selected to be distant to the actual values by greater than 50%.

We found that a chirp applied in current clamp and an OU process applied in voltage clamp resulted in more accurate model fits and in less time than the step response. Using the UKF method, it was possible to achieve estimates of the parameters that were within 1% of the actual values for the model in less than 2 seconds of obtained by stimulating the neuron with an optimized OU process. The OU and the chirp methods both show a steady increase in state space coverage with longer recording times while the step response asymptotes as the neuron settles into steady state behaviors after the initial transient. Surprisingly, the ramp has marginal improvement over the step, the evidence



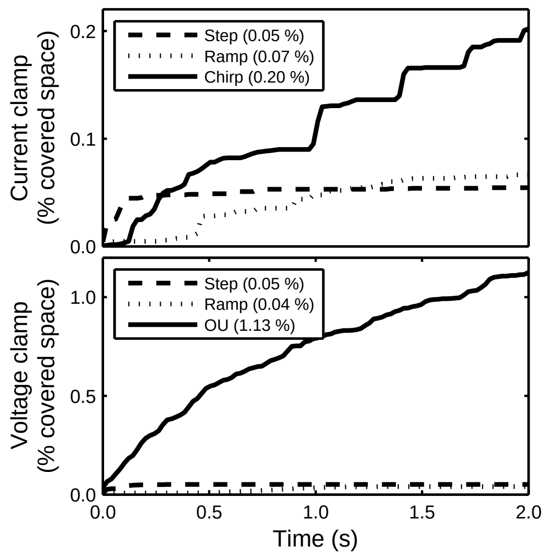
This graph summarizes the results from the UKF parameter estimation based on data collected from the GACell in current clamp (left) and voltage clamp (right) using a step, ramp and the optimized input (chirp-like current in current clamp and Ornstein-Uhlenbeck process in voltage clamp). Results are compared under noiseless conditions (solid bars) and added noise, to emulate in-vitro like conditions (open bars). Top panel shows the final percentage from the real value. Bottom panel shows time it takes to reach and stay within 5 % of its final parameter estimation.

Figure 6.10: Summary results of parameter estimation with different stimulus waveforms.

suggests that this is because ramps do not hyperpolarize the neurons after causing them to spike, which is required to characterize many of the slow currents in the model.

Surprisingly, adding realistic noise to the simulations had only minor deleterious effects on parameter estimation. We attribute the robustness of the UKF to noise because it explicitly models the noise, so that it may be properly accounted for in the model. Therefore, we are hopeful that given a good starting model, fitting models to data can be done under realistic noise conditions.

There are technical problems in implementing these stimulus waveforms experimentally. When using voltage clamp, spatial clamp is poor, resulting in behaviors outside of the vicinity of the electrode [79]. This may result in behaviors that cannot be explained from a recording at a single position within the neuron. In designing the optimized stimulus waveform it is possible to add power constraints in the optimization. However,



Time evolution of the percentage of space covered when a step, ramp, and the optimized input is applied to the GACell in current clamp (upper panel) and voltage clamp (bottom panel). The number that follows the indication of the applied input in each panel represents the final percentage of state space covered after 2 seconds of simulation.

Figure 6.11: Percentage of state space covered as a function of time.

we did not implement this constraint and therefore it is possible that the stimulus could damage the neuron or go beyond the maximum ranges of the amplifier. In practice, when we applied the optimized stimulus waveform to the neuron it killed the cell. In retrospect, this was not surprising because the objective is to maximize the opening and closing of the ion channels in a short period resulting in a massive ionic flux across the membrane and loss of the potential gradients. By adding a power constraint, we can find a balance between perturbing the neuron and killing the neuron.

While our findings suggest that OU stimulus may be be better for characterizing neurons, it does not discount the value of step responses for characterizing a neuron and fitting model parameters. Instead, we posit that the step stimulus is not optimally efficient. When experimental conditions are changing rapidly, as they always do while recording from brain-slices, optimal stimulus waveforms to achieve a data set best suited for your needs is of paramount importance.

The stimulus waveforms developed in this chapter have been optimized given the parameters of the waveform and for the model neuron we used, but we do not suggest that this is a globally optimal stimulus waveform. Instead, this chapter demonstrates a process by which a stimulus waveforms can be compared and optimized.

We used the UKF method to fit the parameters of the model, but this was just one of many methods that could be used. Regardless of the method used for fitting, the model must have the potential to fit the data. Whether the data is contained in the potential behaviors of the model is known as "observability", and it is not easy to determine *a priori*. In this case, we used the same model as used to generate the data, so we were assured of observability. However, when fitting a model to real data observability is not assured. If the model is unable to generate a perfect fit, because it lacks critical currents or spatial representation, it is uncertain what the UKF algorithm will do to achieve a best fit. It should be cautioned that just because a model converges does not mean that it is accurate.

The optimized OU process was designed to be a single stimulus waveform that could elicit all the behaviors of the neuron in a short period of time. However, another consideration in stimulus deign is that correlations between the different gating parameters makes fitting the parameters difficult. An alternative approach to stimulus design is to generate a series of stimulus waveforms that are designed to isolate the individual currents, making it easier to make accurate models of their respective behaviors.

6.5 Conclusion

The input applied to a neuron has a significant impact in the quality of the estimation of parameters and states. The ideal stimulus will force the neuron though its large repertoire of behaviors in a short period of time. Using a model to measure the percentage of state space covered in response to the stimulus provides a quantitative measure of the neuron's behavior. The percent state space covered can be used to optimize stimulus waveforms. In comparing optimized stimulus waveforms to more traditional ones, we find that models can be fit more accurately and in less time using optimized waveforms.

Chapter 7

New methods to estimate Phase Response Curves

Epilepsy, Parkinson's and Alzheimer's are diseases in which synchrony between different cortical areas is affected [80]. The disrupted network functioning could be the result of changes in the dynamics of each brain region, changes in the connectivity among regions, or both. Hence, models that include connectivity and cell dynamics are useful to understand how these pathological behaviors can emerge. But large scale simulations of neuronal networks can be computationally very intensive. There is a perceived trade off between computational effort and accuracy of cell model, but if the model represents the important dynamics of the cell and abstracts away the details that are not, efficiency and accuracy can be very high [12]. A Phase Response Curve (PRC) measures the change in the time of the neuron's next spike after receiving a stimulus [81, 82]. This simple model can be used to make predictions about how networks of neurons will synchronize. PRC models have been used to study the effects of Deep Brain Stimulation on network synchrony in Parkinson's disease [10].

PRCs have also been used to understand the transition between different phases in a seizure. Animal models of seizure like activity show that, both, the neuron's firing rate and synchrony changes over the seizure [83]. At the onset, the neurons fire at high frequency without synchrony. Then, the neural activity transitions into a more coherent and lower firing rate state. It has been proposed that this transition is caused by changes in the phase response curve at different firing rates in the neurons within the network, leading to a shift in synchrony [9]. This is an example where it is necessary to measure a neuron's PRC as a function of its firing rate.

In mathematical models of the neuron, the PRC can be calculated analytically or estimated numerically [84]. The PRC can also be measured experimentally by perturbing a regularly firing neuron and measuring the effect of the perturbation on the spike times [85]. For example, the neuron can be stimulated with a synaptic-like current applied at a random time [81] and the change in the spike time is recorded. The time the stimulus is applied, normalized by the neurons unperturbed interspike interval (ISI), is called the "phase".

However, in experiments the measured spike times are corrupted by jitter and long term drift. Hence measuring PRC experimentally faces, at least, 4 problems: 1) many neurons do not fire periodically, 2) noise and other uncontrolled inputs to the neuron cause jitter in the spike times, 3) the neuron's dynamics and firing rates drift over the experiment, and 4) measuring the PRC at one firing rate is not predicitive of the PRC at another firing rate.

Different methods have been used to remove the jitter [85] and drift [86]. One approach to dealing with the long term drift is to use closed loop control to change the current at each spike time in order to make the cell fire at the same rate over the duration of the experiment [86]. To deal with jitter, synaptic inputs are randomly distributed over the phase and function is fit to the data to estimate the mean phase advance. Different functions can be fit to the data through a least square error minimization, such as 1) a polynomial function [61] or 2) a truncated Fourier series [87]. A bayesian approach has also been proposed to infer the PRC [88] by modeling the jitter and fitting the model to maximizes the likelihood of the observed data. Another approach consists of applying noise instead of a synaptic like current to a regularly spiking neuron. The averaged stimulus preceding a spike is called the "spike triggered average". The spike triggered average is related to the PRC by integration [60]. As the length of the preceding stimulus is different for each spike, people have used different approaches to normalize the data [89, 90]. Interestingly, no one has yet proposed to use noise as stimulus in non-regular spiking neurons. Nor have regularization tools been used to make robust estimations of the spike triggered average.

However, the current methods of estimating PRCs measures the phase advance from the expected period of the neuron. If the expected period of the neuron changes, for example by increasing the applied current, the PRC must be exctracted at the new period.

In this chapter, I introduce two new methods to estimate PRCs under varying conditions and real world experimental problems. The methods were tested, both, in a mathematical neuron model [17], (also see appendix B) and in cortical neurons from rat brain slices. The first method (parameterized PRC) measures how the neuron responds to a single perturbation as the neuron's firing rate varies. This method is able to deal with the jitter, drift and the changing PRC at different firing rates. The second method (infinitesimal transient PRC) asks how the neuron responds to a volley of inputs (represented by a noisy input) under transient conditions, as may be observed at the onset of a seizure. This method does not require regular spiking neurons. It is also able to deal with the jitter, drift and the changing PRC at different firing rates. Both methods are able to model neuron's dynamics by considering the effect of perturbations in one or more preceding ISIs.

7.1 Method 1: Parameterized PRC

This method builds on the system identification approach presented in chapter 4. The goal of the experiment is to vary the neuron's ISI and perturb it at each cycle applying synaptic conductance inputs at different times. Then, the change in the neuron's ISI is calculated as a function of both the ISI and the timing of the applied synaptic conductance.

As the ultimate goal of this method is to determine how the unperturbed period of a neuron is modified by synaptic conductance inputs at different firing rates, the analysis is done in two parts: First, the unperturbed ISI is predicted from both the history of the neuron's ISIs and the inputs. This is done by recursively fitting an Autoregressive model with eXogenous input (ARX) to the ISI as a function of the applied current. The ARX model can then be used to predict the firing rate as a function of the applied current. Because the ARX model is fit recursively, its parameters are able to adapt to non-stationary behaviors of the neuron over the duration of the experiment. The result of subtracting the expected ISI predicted by the ARX model from the measured ISI can be thought as prewhitening. Then, a model is fit to the residual to determine the effect of the synaptic conductance on the ISI. The second part of the analysis seeks to explain the mismatch between the predicted and measured ISI as a function of the history of the linear and nonlinear interactions of two variables: 1) the phase at which the synaptic input was applied and 2) the neuron's firing rate. The following sections describe the design of the experiment and the two steps of the analysis.

7.1.1 Experiment

Traditionally, we have measured PRCs by holding the neuron at a single firing rate over the duration of the experiment. In this method, the ISI is varied by changing the current applied to the neuron at each spike time. This provides a data set in which the PRC can be estimated as a function of both the stimulus phase and the neuron's ISI. First, a constant current (I_{mean}) is applied to a neuron at the beginning of the experiment. This current is chosen from the ISI *vs* I (applied current) curve selecting the current necessary to make the neuron fire approximately at the a desired ISI. At each spike spike i, the current applied is changed and held constant over the next ISI. The applied current over the i'th ISI is $I_{DC}(i)$. The possible values of I_{DC} are selected randomly at each spike time from the set $I_{DC}(i) \in \{I_{mean} \pm 10\%, I_{mean} \pm 5\%, I_{mean}\}$. In addition to $I_{DC}(i)$, a synaptic conductance input (I_{syn}) is added in each cycle. I_{syn} is applied randomly at different points in phase. The entire experiment is depicted in figure 7.1 (results presented are from the GACell model [17]).

$I_{DC}(i)$ can adopt different discrete values. Five arbitrarily chosen current amplitudes were selected over a range large enough to produce a change in ISI greater than the random variability in the neuron's ISI. The value of $I_{DC}(i)$ is randomly selected at each spike time using a Markov process that meets the following conditions: 1) over the duration of the experiment the probability of visiting each current amplitude is equal, and 2) at each spike time, the probability of staying at the same current level is greater than changing to another value. The first condition balances the number of spikes across firing rates. The second condition increases the probability of seeing the same current applied over many cycles, to estimate higher order PRCs accurately. It also induces transient responses so that we may measure the settling of the ISI as a function of both

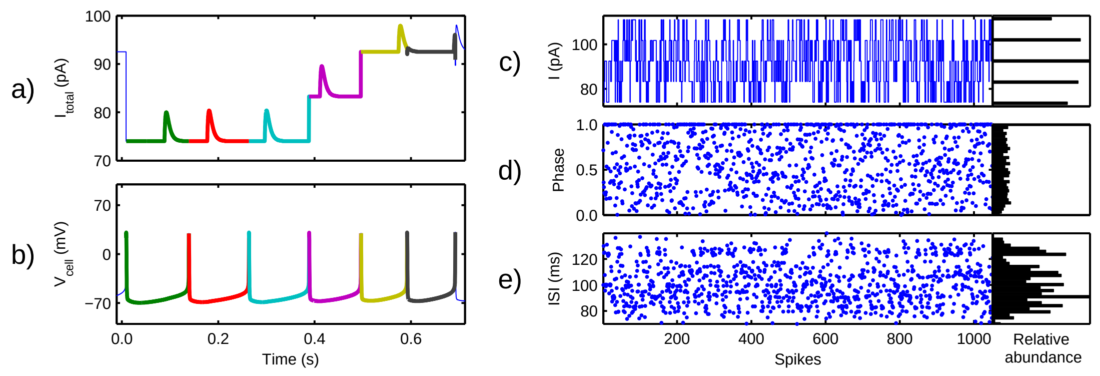


Illustration of the experimental protocol used to estimate Parameterized PRCs. **(a)** shows the total applied current to the neuron during the first 6 spikes. At each spike time, the current is changed and on top of this current, a synaptic current is applied. The time at which the synaptic input is applied is uniformly distributed between zero and the expected ISI. **(b)** shows the corresponding voltage trace. **(c)** shows the total time course of the main current applied to the neuron during more than 1000 spikes. On the right of this figure is shown the corresponding histogram. **(d)** shows point in phase at which the synaptic input was applied and its histogram. **(e)** shows the corresponding ISI and its histogram.

Figure 7.1: Experimental design to estimate Parameterized PRCs.

the timing of the synaptic input and the change in the applied current.

The random sequence for $I_{DC}(i)$ was generated in Matlab and read by the real time experimental interface (RTXI) module. Figure 7.1(c) shows the time course evolution of the applied current as a function of spike time along with a histogram of the amplitudes. Then, a synaptic input is applied at a randomly chosen phase on each ISI, as shown in figure 7.1, panels (a) and (d). The applied current is calculated using a synaptic conductance waveform and membrane potential of the neuron as follows:

$$I_{syn} = G\left(e^{\frac{t}{\tau_{fall}}} - e^{\frac{t}{\tau_{rise}}}\right)(Rp - V)\,, \tag{7.1}$$

where τ_{fall} is the falling time constant (6.23×10^{-3} ms), τ_{rise} is the rising time constant (2.16×10^{-3} ms) of the conductance, Rp is the synaptic reversal potential, which is 0 mV for excitatory synapses or -90 mV for inhibitory synapses, V is the neuron voltage (mV) and G scales the conductance to current.

If the cell fires before I_{syn} is applied, no synaptic input is applied to the neuron and the interval is not used in estimating the shape of the PRC.

7.1.2 Predicting the neuron's ISI

The predicted change in ISI is calculated as a function of the applied current using the ARX model:

$$\begin{aligned} ISI(i) &= -a_1 ISI(i-1) - a_2 ISI(i-2) - \ldots - a_m ISI(i-m) + \\ &\quad b_1 I_{DC}(i-1) + b_2 I_{DC}(i-2) + \ldots + b_n I_{DC}(i-n), \end{aligned} \tag{7.2}$$

where i is index of the current ISI and $i-\alpha$ the preceding ISIs. The number of historical ISIs and applied currents are determined by the variables m and n respectively.

Once the number of parameters (m and n) are selected, the $a's$ and $b's$ can be determined by fitting the model to the data by solving for the least square solution. However, as the ISI drifts, it is prudent to estimate the parameters recursively, updating them at each spike to minimize the mismatch between the predictions and the data. Through recursive estimation of the model, drift in the neuron's dynamics is compensated. The recursive estimation can be repeated multiple times to start with a realistic set of initial conditions each time the estimation is repeated.

The recursive least square estimation of the parameters (*i. e.* the $a's$ and $b's$) of the equation 7.2 is performed following the formulation presented by [65]:

$$y_i = H_k \overrightarrow{x}_i + \nu_i \tag{7.3a}$$

$$\tag{7.3b}$$

where y_i represents ISI. $\overrightarrow{x}_i$ is the vector of parameters

$$\overrightarrow{x}_i = \begin{bmatrix} a_1 & a_2 & \ldots & a_m & b_1 & b_2 & \ldots b_m. \end{bmatrix}^T \tag{7.4}$$

H is a matrix that contains the values of the ISIs and $I_{DC}(i)$ required to predict the ISI, as indicated by equation 7.2. $\overrightarrow{x}_i$ is recursively estimated.

The initial guess for each one of the $a's$ and $b's$ are stored in $\overrightarrow{x}_0$ and their corresponding variance in the matrix P_0

$$\begin{aligned} \overrightarrow{x}_0 &= E(\overrightarrow{x}_i), & (7.5a)\\ P_0 &= E\left[(\overrightarrow{x} - \overrightarrow{x}_0)(\overrightarrow{x} - \overrightarrow{x}_0)^T\right]. & (7.5b) \end{aligned}$$

The recursive estimation is calculated using the kalman filter formalism:

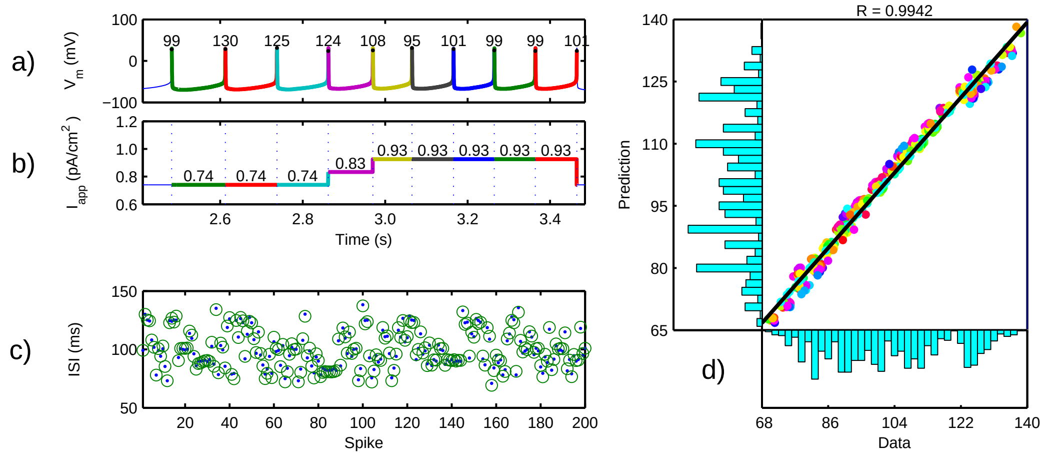
$$\begin{aligned} K_i &= P_{i-1}H_i^T\left(H_iP_{i-1}H_i^T + R_i\right)^{-1}, & (7.6a)\\ \overrightarrow{x}_i &= \overrightarrow{x}_{i-1} + K_i\left(y_i - H_i\overrightarrow{x}_{i-1}\right), & (7.6b)\\ P_i &= \left(I - K_iH_i\right)P_{i-1}\left(I - K_iH_i\right)^T + K_iR_iT_i^T. & (7.6c) \end{aligned}$$

The predicted ISI is calculated from equation 7.6b: predicted ISI $= H_i\overrightarrow{x}_{i-1}$.

Figure 7.2 shows the measured ISIs from the [17] model and the predicted ISI using the recursively fit ARX model. The details of the GACell model is included in B Panel (a) shows the first nine spikes and panel (b) shows the main current applied to the neuron (I_{syn} was applied but for clarity is not shown in this figure). The predicted ISI was calculated as described by equation 7.6. m and n, which determines the number of historical ISIs and applied currents, were arbitrarily set equal to 5. In panel (c) is plotted the predicted ISIs and the measured ISI against spike number, and in panel (d) is plotted the predicted ISI against the measured ISI, the Pearson R correlation is 0.9942.

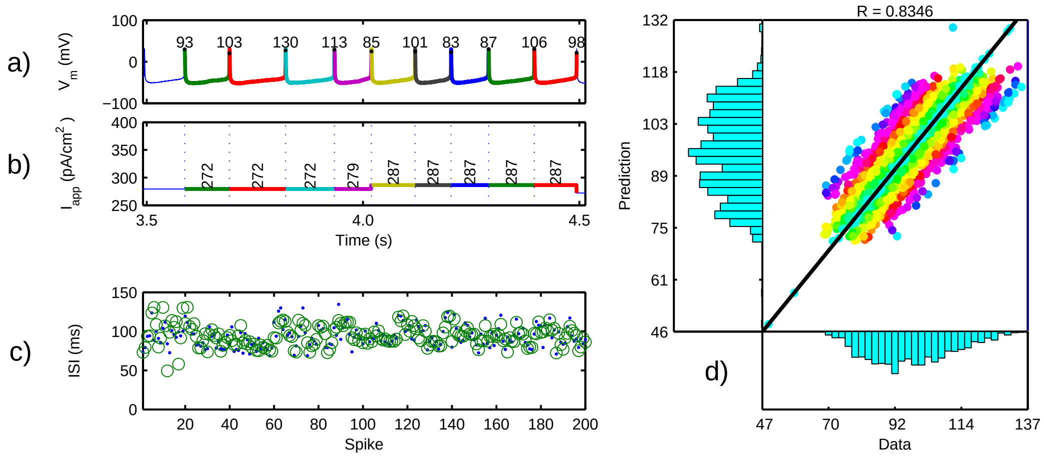
It is interesting to note that while neurons are highly nonlinear, a simple linear ARX model is able to describe more than 99% of the ISI variability as a function of the applied current. The remaining 1% is related to the perturbation applied at different phases at each ISI.

Figure 7.3 shows the actual and predicted ISIs and their correlation from a real neuron. The details of the experimental preparation and patch clamp methods are included in appendix A. In this case, the correlation coefficient between the predicted and measured ISIs was 0.8346. The lower R value reflects noisy inputs and behavior of real neurons that cannot be accounted for by the model. ISI prediction was performed on 60 different cortical neurons. On more than 30 cells, the correlation coefficient was higher than 0.5 (see figure 7.6). The remaining 30 cells did not fire periodically and therefore a linear model could not provide highly accurate predictions.



(a) shows the first nine spikes that a computational neuron model fires when a current, calculated as described in section 7.1.1, is applied to it. **(b)** shows the main current applied to the model. The synaptic input it is also applied, however it is not shown here. The actual (·) and predicted (○) ISIs are shown in panel **(c)**. Panel **(d)** plots one against the other to show the correlation between the two.

Figure 7.2: Recursive ISI estimation in a neuron model.



Panels as presented in figure 7.2.

Figure 7.3: Recursive ISI estimation in a pyramidal neuron.

7.1.3 Calculating the neuron's parameterized PRC

The previous section described how to calculate the predicted ISI as a function of the applied current. In this section, we will relate the residuals of the predicted ISI from the measured ISI to the time at which I_{syn} was applied.

The"spike time advance" (STA) will be used to refer to the difference between the predicted and the measured ISI and "phase" (P) will be used to refer to the ratio of the time at which the synaptic input was applied (relative to the previous spike) to the predicted ISI. Hence, P is bounded between 0 and 1.

In this section, the STA is calculated as a function of both the predicted ISI and the phase of the applied stimulus. As described later, a polynomial function used to fit the STA as a function of the predicted ISI and the stimulus phase. Because this polynomial is two dimensional, it has a large set of independent variables, which can vary by several orders of magnitude in their values. To produce more reliable results than fitting the model using least square solution, the model was fit using the partial least square regression (PLSR). PLSR is also useful when there is a large number of variables and some variables are highly correlated. PLSR assumes that the response of a system is governed by the interaction of latent variables that are hidden in the data (predictor variables) [91]. Here, I use the SIMPLS algorithm [92] to infer the latent variables as a linear combination of inputs to the system.

The PLSR model is fit as follows: If you have a set of n observations $\overrightarrow{y}_{n\times 1}$ and a set of input variables $\mathbf{U}_{n\times m}$ (where each row in $\mathbf{U}$ represents the variables that determine y_i), the SIMPLS algorithm finds a vector $\overrightarrow{\beta}_{m\times 1}$ such that $y_i = U_{i\times m}\overrightarrow{\beta}_{m\times 1}$. *i. e.* the SIMPLS algorithm finds the best linear combination of the predictor variables across observations to describe the response of the system.

Here, the predictor variables used to calculate the STA are the na recursive terms of the STA, *i. e.* the historical values of STA used to calculate the new STA and the product of the phase (P) and the predicted ISI (ISI_{pred}). If $na > 0$, then the first na components of $U_{i\times m}$ are:

$$U_{i,1:na} = STA_{i-1} \quad STA_{i-2} \quad \cdots \quad STA_{i-na}. \tag{7.7}$$

The product of the phase and the predicted ISI are calculated taking the power of each variable from 0 to np and considering the nb historical values. For example, if $np = 2$ and $nb = 2$, then the remaining terms on (U) are

$$U_{i,na+1:m} = \begin{array}{c}[Y_i,\ Y_i^2,\ P_i,\ P_iY_i,\ P_iY_i^2,\ P_i^2,\ P_i^2Y_i,\ P_i^2Y_i^2,\ ; \\ Y_{i-1},\ Y_{i-1}^2,\ P_{i-1},\ P_{i-1}Y_{i-1},\ \cdots P_{i-1}^2Y_{i-1}^2].\end{array} \tag{7.8}$$

where $Y = ISI_{pred}$ and m is the number of predictor variables considered for the calculation of the STA. m is given by

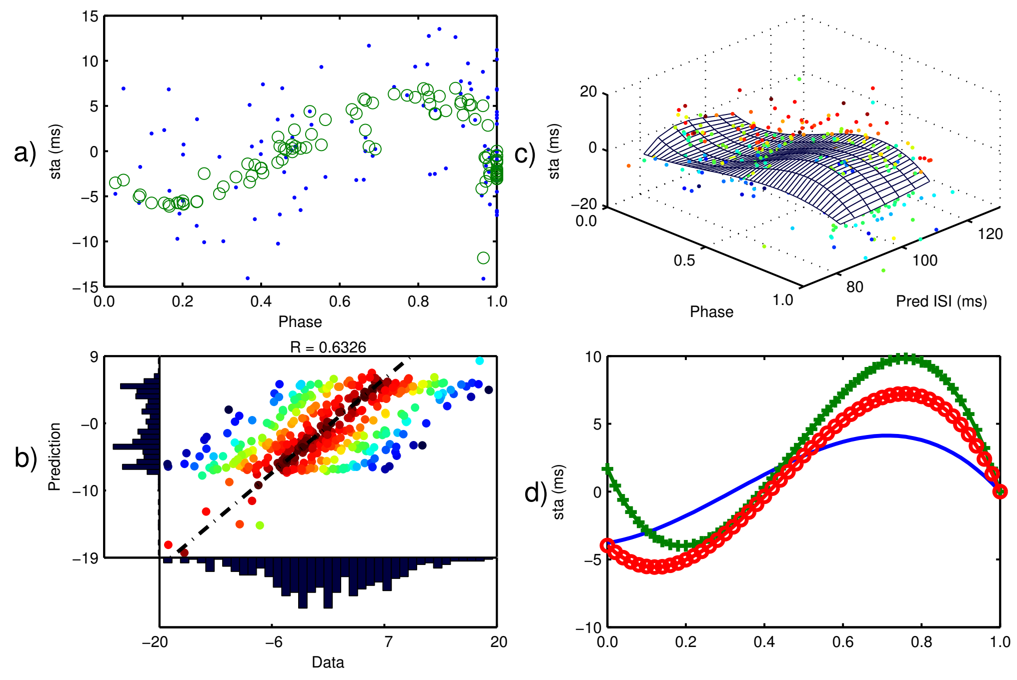
$$m = na + nb \times ((np+1)^2 - 1). \tag{7.9}$$

Figure 7.4 shows the estimated PRC for the neuron model used in figure 7.2. Here, $np = 3$, $na = 0$, and $nb = 1$. This method is able to make predictions with a correlation coefficient of 0.8477 across different firing rates. Also note that the obtained model is able to detect differences in PRCs at different firing rates (Panels (c) and (d)).

Figure 7.5 shows the corresponding PRC for the pyramidal neuron used in figure 7.3. For this model, the parameters were set at np=3, na=0, and nb=1, as it was done with the neuron model. In this experiment, the correlation coefficient of the predicted and observed STA was 0.6326 across different firing rates. Notice how dramatically the estimated PRC changes as a function of the firing rate.

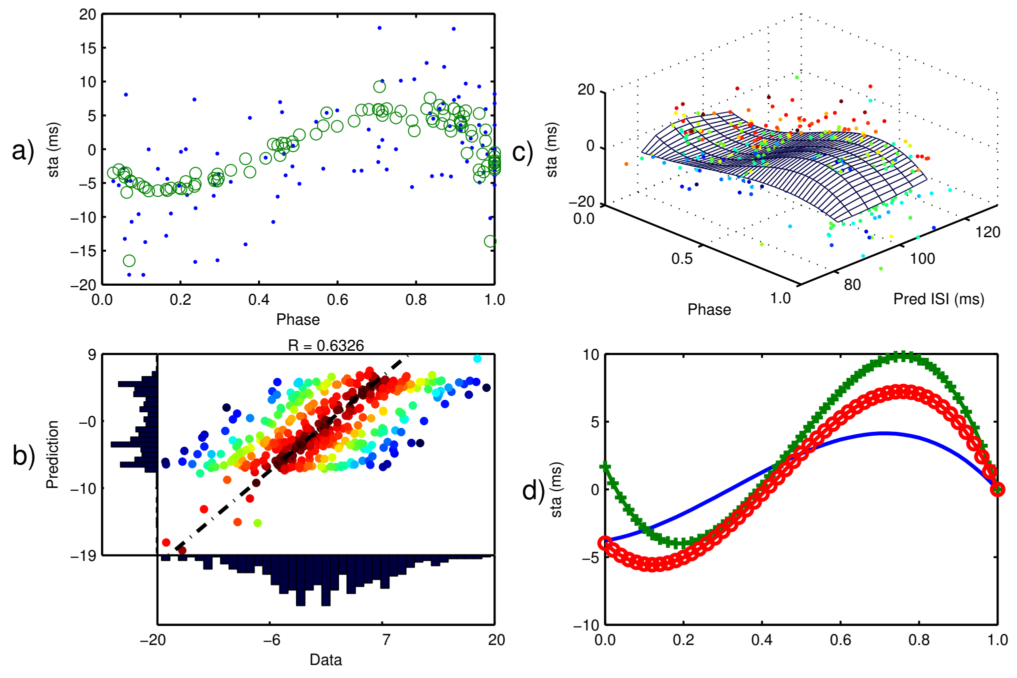
Finally, figure 7.6 summarizes the results obtained from 60 pyramidal neurons. For each neuron, first an adaptive ARX model was fit to the data to infer the unperturbed ISI. The correlation coefficient (R_{ARX}) between the predicted and observed ISI is plotted in blue. The correlation coefficient between the predicted and the observed STA is shown in green (R_{PLSR}). The cumulative prediction ($R_t = R_{ARX} + (1 - R_{ARX})R_{PLSR}$) is shown in red. In more than 30 cells, the parameterized PRC model can account for more than 80% of the variability of the data.

It is important to mention that in more than half of the recorded neurons, the cells missed spikes, meaning than more than 10% of the recorded ISIs had a value at least twice than expected. This atypical behavior was not accounted for in our model, which explains neurons with the lowest total R values shown in figure 7.6.



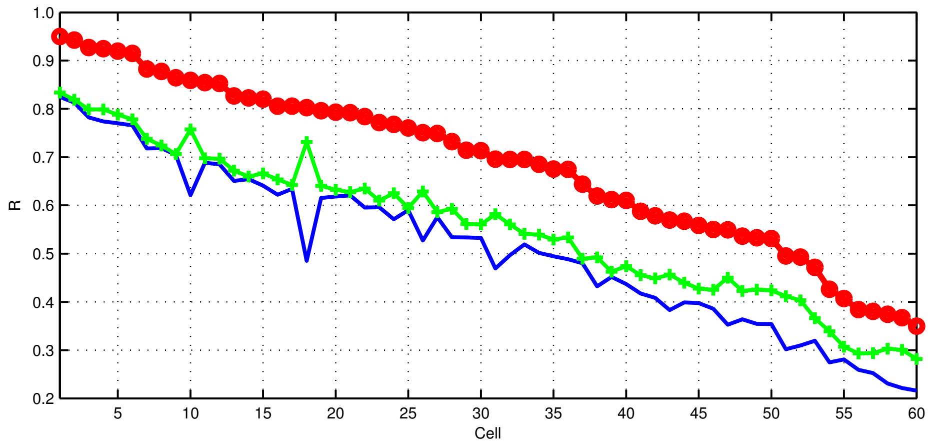
Panel **(a)** shows 100 predicted (○) and actual (·) STA's as a function of phase for the GACell model [17]. Panel **(b)** shows the correlation between the predicted and the actual STA's for the entire set of data. Panel **(c)** shows the predicted (surface) and the actual STA (⋄) as a function of the phase and the unperturbed ISI. Panel **(d)** shows the STA when the unperturbed ISI is 85 (solid red line), 100 ("+" green line) and 125 ("o" red line) ms.

Figure 7.4: Parameterized PRC in a neuron model.



Panels **(a)** to **(c)** as in figure 7.4. Panel **(d)** shows the STA when the unperturbed ISI is 81 (solid red line), 95 ("+" green line) and 108 ("o" red line) ms.

Figure 7.5: Parameterized PRC in a pyramidal neuron.



Blue-solid line indicates the correlation coefficient obtained from the prediction of the ISI as a function of the applied current. Green line ("+") is the correlation coefficient obtained using the model to predict the STA as a function of the phase and the predicted ISI. Red line ("o") is the total variance ($R_t = R_{ARX} + (1 - R_{ARX})R_{PLSR}$) accounted for by the parameterized PRC.

Figure 7.6: Parameterized PRC in 60 pyramidal neurons: summary results.

7.1.4 Discussion

The traditional PRC method quantifies the relationship between the point in phase at which perturbations are applied and the corresponding change in spike time. Here, we introduce a method that allows us to estimate, in the same experiment, the PRC as a function of a second variable. The second variable used in these experiments was the firing rate. The main reason to select the firing rate as a second variable is because in seizures the firing rate of the neuron can vary dramatically and we hypothesize this is important in determining the synchronizability of a network. The approach used here is to induce different ISIs by changing the applied constant current at each spike time, then a synaptic input is applied on each cycle.

To induce the change in the ISI, the applied current is varied around a mean value. The mean current can be calculated from a firing rate *vs* applied current (F-I) curve. The selected value should make the cell fire in the vicinity of the firing rate of interest. The change in current applied at each spike time will enable a linear (ARX) model of the firing rate as a function of the applied current to be obtained, which will in turn be used to prewhiten the data.

The analysis has 2 stages: 1) a recursive linear model is fit to predict the ISIs. 2) the mismatch between the predicted and measured ISI is modeled as a function of both the ISI and the phase.

By fitting the ISI prediction model recursively, the parameters can evolve over the experiment to account for non-stationaries in the neuron. The STA is calculated as the mismatch between the predicted and measured ISIs (the residuals). In both computational models and in real neurons, we found that much of the spike time variability could be explained by these models. It is surprising to us just how well this linear model can predict the spike times of the highly non-linear neuron.

Then, we use a polynomial combination of the phase and the ISI to describe the STA. *i. e.*, this stage fits the residuals of the previous stage to a model. The resulting model corresponds to the neuron's PRC.

The predictive power of this method is quantified by the correlation coefficient of the predicted and measured STA. Figure 7.6 shows that in more than 50 cells (83.3 % of the data), the fitting was able to describe more than 60% of the total variance of the experimental data. The data from the remaining 10 cells come from non-regular spiking cells.

The obtained PRCs have the typical shape of the curves obtained by traditional methods, which implies that this method is, at least, qualitatively similar. The PRCs calculated from the computational model and the cortical neuron (panel d) of figures 7.4 and 7.5) show that the PRC skews toward the right and the peak goes higher as the mean ISI increases (firing rate decreases). This qualitative behavior is also reported by Gutkin, *et. al.* [82] in a computational model of excitatory neurons. They suggest that the displacement of the peak to the right as the ISI decreases is a consequence of slowly adapting potassium currents that cause the cell's firing rate to change. The adaptation current is maximum shortly after an action potential and suppresses the effect of synaptic inputs. The adaptation current deactivates later in the ISI, so the effect of transient inputs reach its maximum close to the end of the ISI. As the ISI decreases the suppressing effects of the adaptation current is present for a larger portion of the phase.

Network simulations indicates that synchrony depends on the firing rate. This modulation of synchrony is mediated by changes in the shape of the PRC at different firing rates. Neurons with PRCs that only have phase advance show increased synchrony with increasing firing rate, while neurons with PRCs with both phase advance and delays decrease synchrony with increasing firing rates [93]. This method for measuring PRCs presented here makes it possible to measure PRCs at different firing rates in the same experiment. It may be useful to experimentally validate how synchrony depends on firing rate.

Furthermore, this method can be used to study the effect of antiepileptic drugs, which may affect neurons at high firing rates more than at low firing rates.

This method can also be extended to relate the STA to other variable besides firing rate. It is also possible to extend this formalism to relate the STA to 3 or more variables. A natural selection for the third variable would be the intensity of the perturbation applied in addition to the phase and ISI of the neuron.

7.2 Method 2: Infinitesimal Transient PRC

Spike time advances to a synaptic stimulus measured experimentally are noisy. Two common sources of noise are synaptic inputs and stochastic fluctuations of the neuron's ion channels [94]. The effects of these sources of noise can be minimized by using long recordings and averaging the response to the stimulus applied many times. Hence, the neuron is required to fire reliably for long periods. However some cells have strongly accommodating currents and therefore can't fire reliably for long periods. This has in the past practically limited the kinds of neurons that we can estimate PRCs from.

Fortunately, there are different ways to produce spikes reliably. When a current above threshold is briefly applied (hundreds of ms), most neurons will fire reliably for a few spikes [95]. Then, by turning off the current, the neuron can recover. Hence, a square wave current can be used to induce a reliable train of spikes. [1]

However, because the interspike interval changes over the first few spikes after the onset of a current step, it is difficult to determine a-priori the expected ISI required to

[1] A square wave is a signal that alternates between a constant amplitude and zero with a given period. The percentage of the period that the signal remains in the non-zero value at each period is called "duty cycle".

determine the stimulus phase, as was needed in the previous method. An alternative approach is to use a continuous stimulus waveform over the ISI and then use linear algebra methods for extracting the PRC given the measured spike time advances. One commonly used stimulus waveform for estimating linear systems is Gaussian white noise (GWN). While this approach works well in theory, in practice it has been found to be numerically unstable. Using regularization methods it is possible to overcome these numerical problems. In this section, we propose to use a GWN stimulus waveform in conjunction with the current step to estimate the transintly changing PRCs.

Algorithmically, this method is summarized as follows:

- A step current is applied repeatedly to cortical neurons and a low-pass filtered white noise current is added to the step current.
- The stimulus is repeated m times and the corresponding voltage trace is recorded.
- On each one of the m cycles there are at least n spikes, resulting in $n - 1$ ISIs.
- The change in the spike time is related to the applied noise through an ill-posed linear system. This system is solved for by using regularization techniques.
- Once the model is fit, it is validated by comparing the predictions with the measured STA.

The rationale of this model is that if the PRC for a neuron exists (which is unknown and is to be determined), the applied noise acts as a transformation matrix of the PRC to obtain the STAs. Hence, as the applied noise and the measured STAs are known, the problem is then to use linear algebra to solve for the PRC. The solution to this system requires regularization techniques because the pseudo-inverse is highly sensitive to noise. This results in a PRC for each ISI following the onset of the current step. Therefore it results in a family of PRCs PRC_k, where k indicates the PRC for the k'th ISI following the current step onset. We will call these PRCs transient PRCs because they change over time. At some point, the neuron's firing rate stabilizes and the ISI's may be combined together to estimate the steady state PRC, PRC_{SS}. The resulting PRCs are independent of the stimulus waveform used. Therefore, they are considered infinitesimal PRCs, iPRCs, which predicts the neuron's response to an infinitesimally short input, or delta function.

The method has been outlined earlier [94], but in practice it has been found that estimating PRCs this way is numerically unstable. Here I introduce the use of regularization techniques to improve the estimated PRCs. I also apply this method to measure PRC's from non-regular spiking neurons, to measure higher order PRC's and to measure the PRC from consecutive ISIs on a train of spikes.

The following sections describe the experiment and the analysis performed to obtain the model. Figures 7.9 and 7.10 show the estimated PRCs by solving the linear equation using the pseudo-inverse and regularization methods.

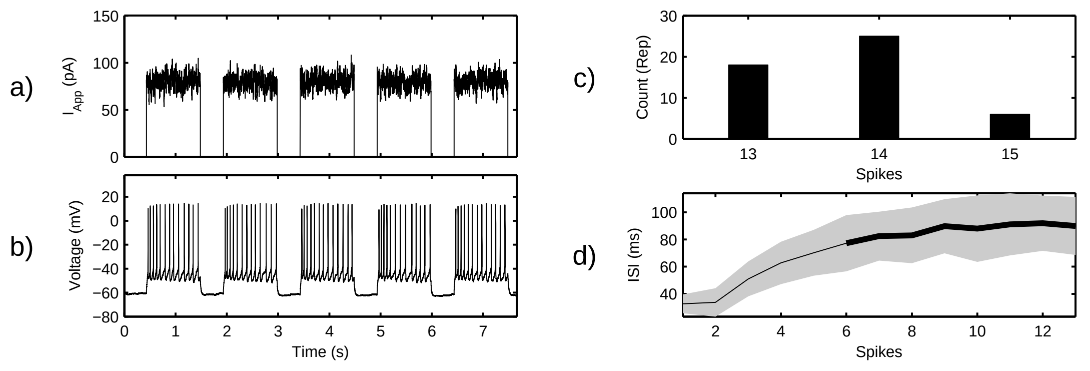
7.2.1 Experiment

In this section, we describe the stimulus waveform and the protocol for measuring the transient infinitesimal PRCs from neurons, as depicted in Figure 7.7. Here, the current is applied to a model neuron [17]. The constant current was selected to makes the cell fire at an average of 90 ms. Then, a low-pass filtered version of white noise is added to the constant current. The amplitude of the noise is set between 1 and 5 % of the constant current. A train of 49 repetitions, with a length of 1.5 s and a duty cycle of 70%, was applied to the model (panel a). Notice that the model fired at least 13 spikes on each one of the 49 repetitions (panel c) and that the average ISI for each one of the 13 individual ISIs across the 49 cycles settles after a transient that lasts a few spikes (panel d).

Figure 7.8 shows the experimental results from a cortical neuron. The mean value of the applied current makes the cell fire at an average of 100 ms (after the transient). The low pass white noise current was added to the constant current and a train of 51 cycles of 1.5 s and a duty of 70% was applied to the neuron. Each one of the 51 cycles had at least 8 spikes. However, in most of the cycles the neuron responded firing 10 spikes (panel c).

7.2.2 Analysis

The goal of the analysis is to determine how the noisy current modulates the neuron's firing rate. This analysis is done by relating the noisy current to the change in ISIs across cycles.



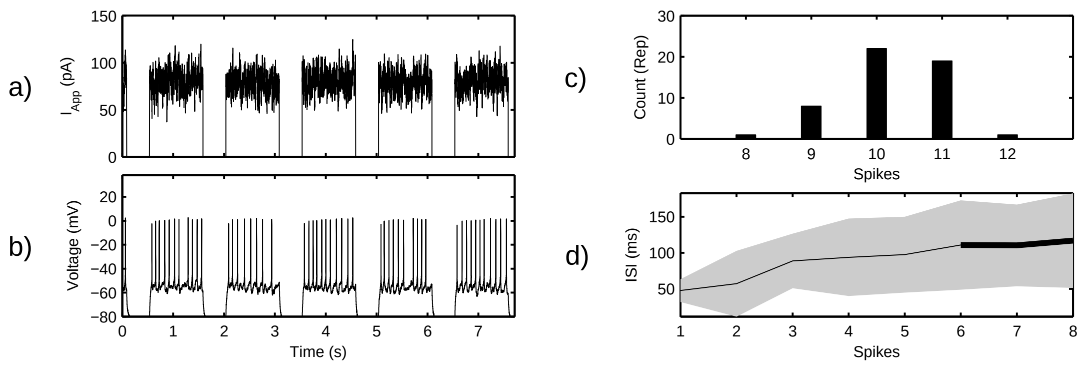
Panel **(a)** shows 5 of the 49 steps of current that were applied to the mathematical neuron model used [17]. Panel **(b)** shows the corresponding voltage trace obtained. Panel **(c)** shows the distribution of spikes across cycles. Black line on panel **(d)** shows the average ISI of each one of the first 13 spikes on the 49 cycles, gray area indicates the $\pm$ 2 standard deviation band and the wide black line marks the last ISIs that are considered stationary used to calculate the steady state PRC, PRC_{SS}.

Figure 7.7: Current-step response of a neuron.

For each experiment, the cell receives m steps of current and it fires at least n times, *i. e.* each cycle has at least $n-1$ ISIs. Here, it is assumed that the neuron's dynamics do not change over the duration of the experiment. As the applied current goes to zero before each cycle, it is also assumed that the effect of the neuron's accommodation current is reset before each cycle. Hence, the observed variability in the k'th ISI across repetitions is mainly caused by the application of the noisy current. Then, the average ISI is calculated for each one of the $n-1$ ISIs. The STA is calculated by subtracting the average ISI $< ISI_k >$ for each interval. This generates $n-1$ STA vectors with a length of m each.

Because the ISI varies in length, the stimulus waveform also varies in length. To normalize the stimulus length, the noisy current resampled into b samples by cubic spline interpolation, as indicated in panel (a) of figures 7.9 and 7.10. The stimuli can then be concatenated in a matrix R that is $m \times b$. Then, the relationship between the STA of the k'th spike following the current step onset, the PRC and the applied noise is:

$$\overrightarrow{S}_k = \mathbf{R_k} \overrightarrow{P}_k, \tag{7.10}$$



Panels **(a)** to **(c)** as indicated in 7.7. Panel **(d)** shows the average ISI of each one of the first 8 spikes on the 51 cycles, gray area indicates the ± 2 standard deviation band and the wide black line marks the last ISIs that are stationary.

Figure 7.8: Current-step response of a neuron.

where $\overrightarrow{S}_k$ is a vector of length m that contains all the STA's. $\mathbf{R}_k$ is a matrix that is $m \times b$, where each row corresponds to the resampled stimulus waveform. $\overrightarrow{P}_k$ is a vector of length b corresponding to the PRC (to be determined). Notice that the number of bins (b) used in the downsampling of the stimulus waveform determines the resolution of the PRC. For readability, we will omit the subscript k on these equations for the remainder of the chapter, except where needed for clarity.

$\overrightarrow{P}$ can be solved from equation 7.10 by calculating the pseudoinverse of $\mathbf{R}$. However, because the stimuli used are random the matrix $\mathbf{R}$ is ill-conditioned. Hence, the obtained solution is highly sensitive to noise. In the first section, we will show the results obtained by using the pseudoinverse, in the next section we will show how estimates of the PRC can be improved by using regularization techniques.

Pseudoinverse

First, $\overrightarrow{P}$ was solved from equation 7.10 by calculating $\mathbf{R}$'s pseudoinverse

$$\overrightarrow{P} = \left(\mathbf{R}^T\mathbf{R}\right)^{-1}\mathbf{R}^T\overrightarrow{S} \tag{7.11}$$

As this approach allows us to calculate independent PRCs for each one of the $n-1$ ISIs in the spike train, the $n-1$ PRCs for each experiment were calculated, as indicated in the left graph of panel (b) on figures 7.9 and 7.10 for the computational model and the cortical neuron, respectively. This figure also shows the average PRC calculated from the last ISIs when the neuron has approached a stationary firing rate by using all the intervals indicated by the dark line in panel (d) of Figures 7.7 and 7.8. The average PRC was calculated by concatenating the $\overrightarrow{S}$'s and $\mathbf{R}$'s included in the calculation. The resulting PRC ($\overrightarrow{\hat{P}}$) can then be used to estimate the STA given the stimulus waveform: $\overrightarrow{\hat{S}} = \mathbf{R}\overrightarrow{\hat{P}}$.

The PRC was validated by comparing the measured spike advance $\overrightarrow{S}$ to the predicted spike advance $\overrightarrow{\hat{S}}$, calculated from the input and the estimated PRC, $\overrightarrow{\hat{P}}$. The correlation coefficient between the predicted ($\overrightarrow{\hat{S}}$) and the measured STA ($\overrightarrow{S}$) was used to quantify the accuracy of the prediction and is shown in panel (c) of figures 7.9 and 7.10. To test the robustness of the method to predict $\overrightarrow{S}$, a new $\overrightarrow{P_{75}}$ was calculated using only the 75% of the m STA's for each one the $n-1$ ISIs (minimizing the within sample error). Then, the obtained $\overrightarrow{P}$ was used to predict $\overrightarrow{S}$ for the remaining 25% of the samples (out of sample data). The average R value was estimated through a 100 fold bootstrap. The average correlation coefficient between the predicted and measured STA is also shown in panel (c) of figures 7.9 and 7.10; notice how dramatically the correlation coefficient drops when $\overrightarrow{P_{75}}$ is used to predict the out of sample data, especially for the data from a real neuron. This indicates that the pseudoinverse 1) overfits the data and, 2) it is highly susceptible to noise.

The STA is also affected by perturbations applied to the neuron in the previous interval. Hence, a PRC considering the stimulus applied over the ISIs, ISI_k and ISI_{k-1} can be used. We will indicate this PRC as $\overrightarrow{P}^2$. Considering the past two ISIs it is only possible to calculate $n-2$ $\overrightarrow{S}^2$'s (the first ISI has no preceding spikes, so it was not included). In this case, $\mathbf{R}$ has $2b$ columns, because it includes the sampled current from both the previous and the actual period. $\overrightarrow{P^2}$ was solved using equation 7.11 and the same robustness analysis mentioned above was performed for this 2-periods $\overrightarrow{P}$. The calculated and average $\overrightarrow{P^2}$ is shown in the left figure of panel (d) on figures 7.9 and 7.10 for the computational model and the cortical neuron, respectively. Panel e) shows the correlation coefficients obtained from:

1. Comparing the measured and predicted $\overrightarrow{S}^2$ when all the data is used to calculate $\overrightarrow{P}^2$

2. Averaging the 100 correlations obtained when comparing the measured $\overrightarrow{S}^2$ with their predictions on the out of sample data when $\overrightarrow{P}^2$ was estimated in the within data.

The correlations obtained when the measured STA is compared with its estimation based on 1) using the entire data to make the prediction and 2) using the average for the in-sample and out-of-sample results, shown in panel (e) of figures 7.9 and 7.10.

Regularized solution: truncated singular value decomposition (TSVD)

While the PRCs calculated by the pseudoinverse method (figures 7.9 and 7.10) can explain most of the variance in the STA (R>0.95) for in-sampled data, the resulting PRC looks very noisy. In fact, each value of the PRC has a different sign than the preceding phase and the absolute values are rather large, indicating that the calculation 7.10 is an ill-posed system. In ill-posed systems, *i. e.* systems highly susceptible to noise, equation 7.10 can be solved more robustly by adding a constraint. When no *a priori* information is known about the solution *i. e.*, $\overrightarrow{P}$, a common constraint is to assume that the absolute values of each point in the solution should be minimum (minimum norm constraint). This goal can be achieved using regularization techniques [96, 97]. There are different regularization techniques. Here, the truncated singular value decomposition (TSVD) method is used (see appendix D for a graphical explanation of the method).

Roughly speaking, the solution to the problem

$$\min\|\overrightarrow{P}\| \text{ subject to } \min\|\mathbf{R}\overrightarrow{P} - \overrightarrow{S}\| \tag{7.12}$$

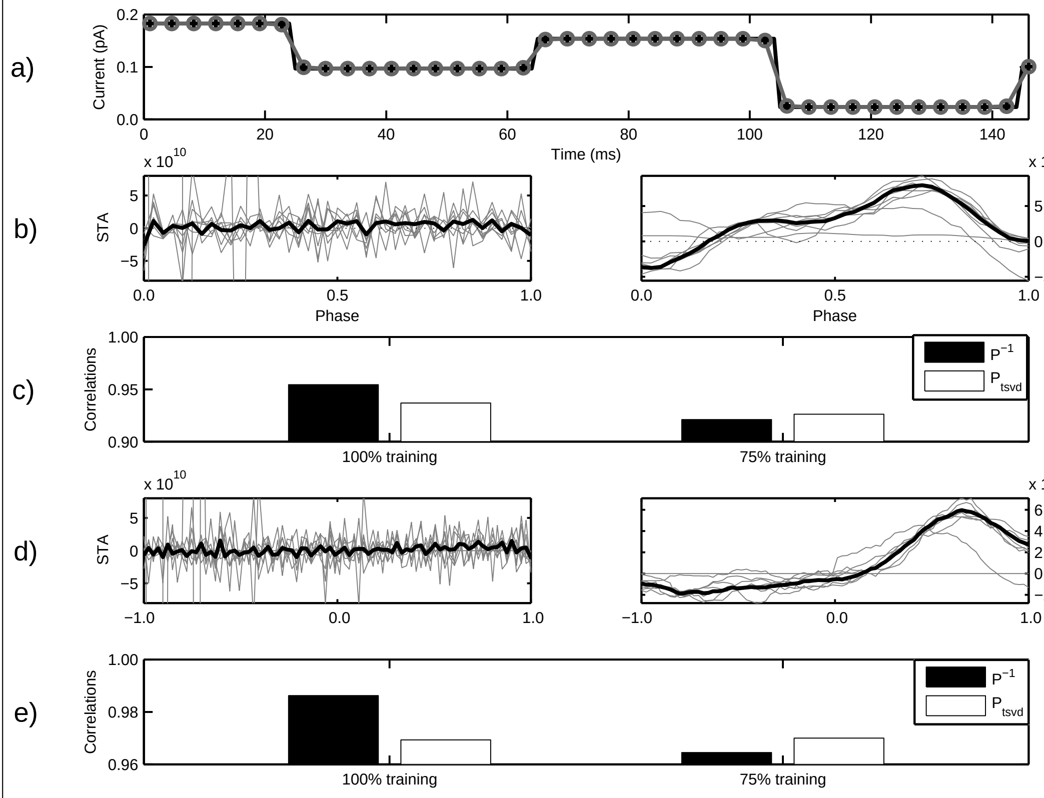
can be obtained by the factorizing $\mathbf{R}$ into its singular value decomposition

$$\mathbf{R} = \mathbf{U\Sigma V}^T = \sum_{i=1}^{b} \overrightarrow{u_i}\sigma_i \overrightarrow{v_i}^T, \tag{7.13}$$

where the matrices $\mathbf{U} = \overrightarrow{u}_1, \ldots, \overrightarrow{u}_b$ and $\mathbf{V} = \overrightarrow{v}_1, \ldots, \overrightarrow{v}_b$ have orthonormal columns, known as left and right singular vectors, respectively, and $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_b)$ contains the singular values of $\mathbf{R}$ on the diagonal.

The singular values of $\mathbf{R}$ are ordered such that

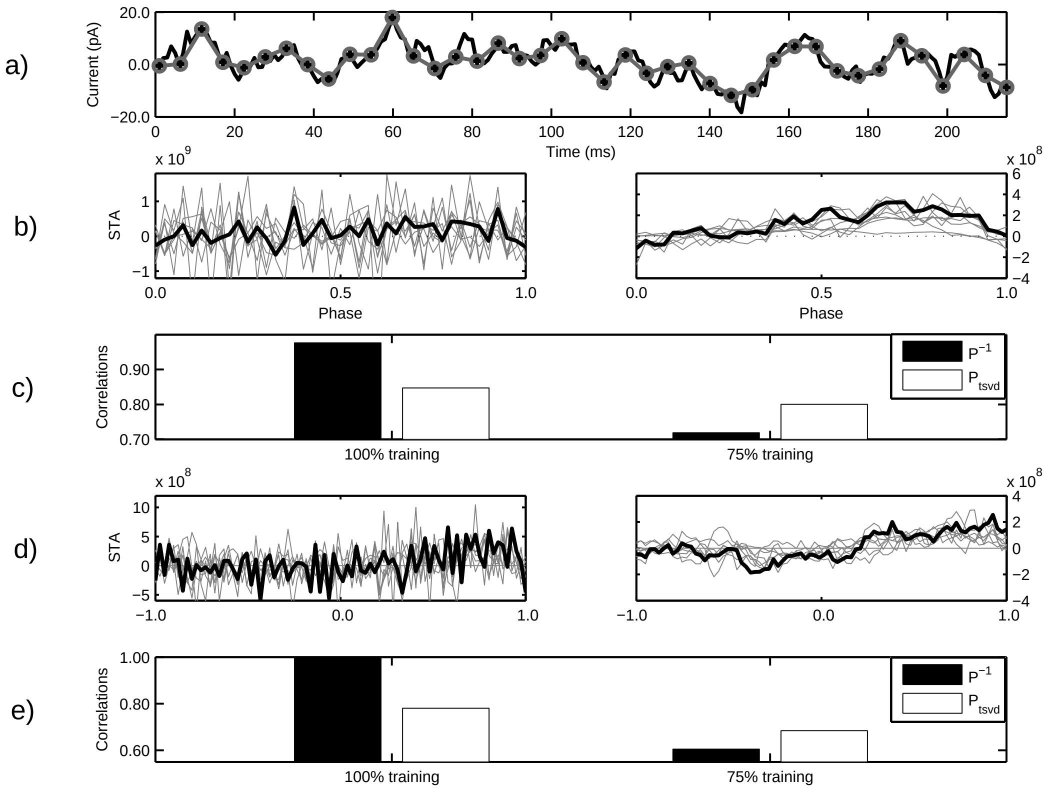
$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_\nu \geq 0. \tag{7.14}$$



Comparison between the PRCs calculated by the pseudo-inverse method and by using regularization techniques.

Panel **(a)** shows the applied current to the computational model (black line) and its discretized version (gray line). Left plot in panel **(b)** shows in gray the individual PRCs calculated for each spike by the pseudoinverse method. The black line shows the PRC considering the lasts spikes with the same ISI, also using the pseudoinverse method. The plot on the right shows the individuals (gray line) and global PRC (black line) using the truncated singular value decomposition method. Panel **(c)** shows the correlation coefficient obtained when the predictions of the spike time advance are compared to the experimental data. Here, the predictions are calculated on the in-sample predictions (100% training) *vs* the out-of-sample predictions (75% training). Panels **(d)** and **(e)** shows respectively the PRCs and their correlations when the PRC was calculated considering the actual and the previous ISI.

Figure 7.9: Infinitesimal transient PRC estimation in a computational neuron model.



Data obtained from a pyramidal neuron located in the hippocampal formation. Results are from the same neuron presented in Figure 7.8. Panels as labeled in figure 7.9.

Figure 7.10: Infinitesimal transient PRC estimation in a cortical neuron.

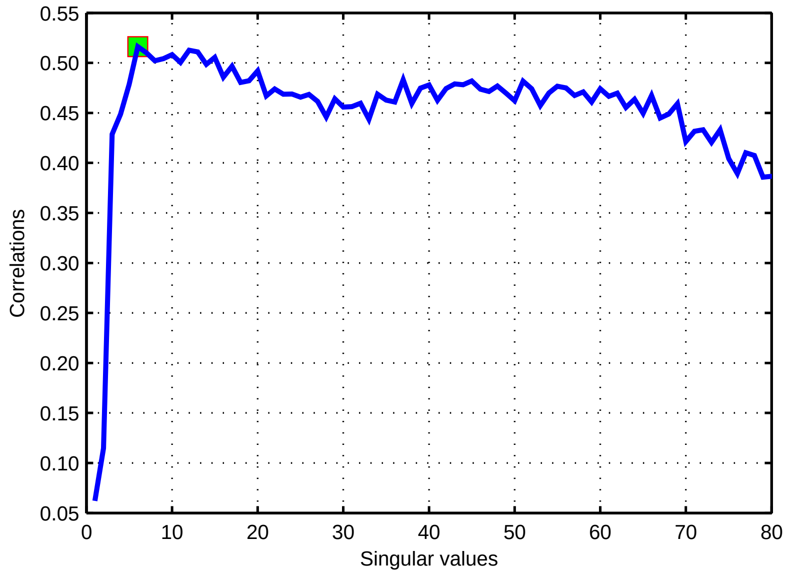
The solution to equation 7.12, given the singular value decomposition of $\mathbf{R}$ (equation 7.13), is:

$$\overrightarrow{P}_{k,l} = \sum_{i=1}^{l} \frac{\overrightarrow{u_i}^T \overrightarrow{S}_k}{\sigma_i} \overrightarrow{v}_i. \tag{7.15}$$

When $l = b$, this solution is the same as the pseudoinverse method resulting in $\overrightarrow{P}_k$ However, in ill-posed systems, the left and right singular vectors $\overrightarrow{u_i}$ and $\overrightarrow{v_i}$ tend to have more sign changes in their elements as the index l increases. As the singular values of $\mathbf{R}$ are ordered from major to minor, and as the index increases, the numerical value of σ_i decreases. Hence, we can see from equation 7.15 that the vectors with higher indices i used in the reconstruction of P amplifies the change of sign that are already present both in the left and right singular vectors $\overrightarrow{u_i}$ and $\overrightarrow{v_i}$. This is due to the fact that σ_i appears in the denominator of equation 7.15. Hence, truncating equation 7.15 to an appropriate value of l eliminates the noisier components of $\overrightarrow{u_i}$ and $\overrightarrow{v_i}$. This truncation makes the solution more robust to noise. The trade-off is between in-sample predictive power and model robustness to noise, which may increase out-of-sample predictive power.

To determine the number of singular values to include, P was calculated using equation 7.15 for $l = 1, 2, \ldots b$. P was estimated using 75% of the data and used to predict the remaining 25% of the data (out-of-sample testing). To estimate the accuracy of the prediction, the correlation coefficient between the predicted and measured STA was calculated. This procedure was repeated 100 times and the correlation coefficients were averaged for each value of l. Figure 7.11 shows the correlation coefficient between the predicted and measured STA as a function of the number of singular values used to calculate P. Notice that first six singular values are critical in making accurate predictions, but including higher singular values degrade the prediction.

The optimal value of the number of bins, b, to use can be empirically estimated by: 1) calculating the PRC using the in-sample data and predicting the corresponding STA in the out-of-sample data; 2) calculating the correlation coefficient between the predicted and the measured STA; 3) repeating this procedure (here, we repeated the estimations 100 times) using a different in-sample and out-of sample data and averaging the correlation coefficient; 4) repeating the previous steps for each one of the different values of b being tested; and 5) then, the optimal value of b is chosen from the highest correlation coefficient between the predicted and measured STA.

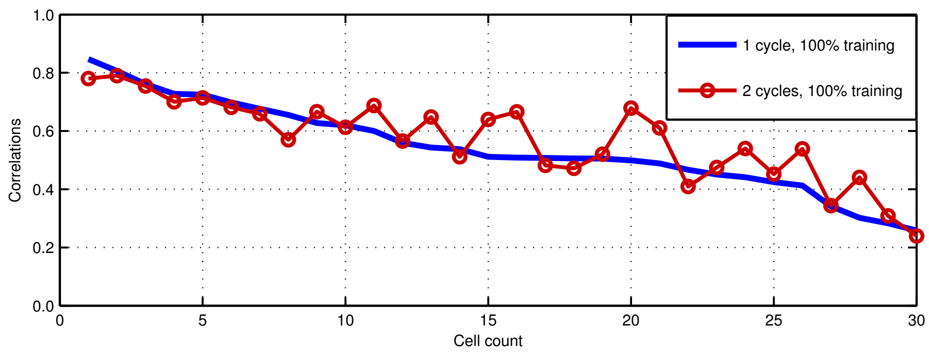


Correlation coefficient between the predicted and measured STA calculated as a function of the singular values used to calculate the PRC from equation 7.15. Green square shows that the maximum correlation is obtained when 6 singular values are used to calculate the PRC for this particular set of data.

Figure 7.11: Out-of-sample comparison between predicted and measured STA.

Figures 7.9 and 7.10, show the corresponding PRCs calculated for a computational neuron model and for a cortical neuron, respectively. Notice that the pseudo-inverse method has a higher predictive value than the TSVD for in-sample prediction errors. This is observed in all the four scenarios: computational neuron model using one (0.9550 *vs* 0.9370) and 2 cycles (0.9860 *vs* 0.9690), and cortical neuron, using one (0.9760 *vs* 0.8470) and 2 cycles (0.9999 *vs* 0.7800). However, for out-of-sample prediction errors, TSVD method outperforms the pseudo-inverse in the all 4 scenarios: computational neuron model using one (0.9260 *vs* 0.9210) and 2 cycles (0.9700 *vs* 0.9640), and cortical neuron, using one (0.8000 *vs* 0.7190) and 2 cycles (0.6840 *vs* 0.6050).

Finally, Figure 7.12 shows the summary results from 30 neurons where the TSVD method was used to calculate the PRC. The figure shows the correlation coefficient between the predicted and measured STA when the PRC was calculated considering the perturbations in current (one cycle) $\overrightarrow{P}$ and the previous two cycles, $\overrightarrow{P}^2$. As it can be seen, in some of the neurons where the Pearson's R is around 0.5, inclusion of the stimulus waveform from the previous cycle improved the prediction above 0.6, indicating that there are long term effects of the stimulus. While these correlation coefficients are significantly above chance (p-value $\ll 0.05$), the neurons selected for these experiments were non-regular spiking cells (coefficient of variation > 0.2) for which the traditional PRC methods could not be applied.



Correlation coefficient between the predicted and the experimental spike time advance in 30 cortical neurons. The spike time advance was calculated using the PRC obtained by the truncated singular value decomposition method.

Figure 7.12: Infinitesimal transient PRC in 30 cortical neurons: summary results.

7.2.3 Discussion

This method was designed to measured the PRC under transient conditions in non-regular spiking neurons. We take advantage of the fact that even non-regular spiking cells fire reliably for a few spikes at the onset of a step current. The reliability of the spike times is randomized by adding noise. Then, the PRC is estimated by relating the change in spike times to noisy stimulus waveform.

Neuron's response to noise vary. Each neuron has a preferred frequency band for which it is sensitive to inputs [98]; this is determined by the passive and active properties of the neuron's membrane [99]. So, the spectral characteristics of the noise used to characterize the neuron's response can affect the outcome of the estimated PRC. High frequency noise may have negligible effect on the change in the spike times because of the neuron's temporal filtering properties. Stimulus waveforms matched to the neuron's preferred frequencies could be optimal in terms of perturbing the spike times with minimal energy [100]. Hence, we hypothesize that the spectrum of the applied noise can be optimized for each neuron to recursively improve our estimates of the PRC. Furthermore, once the optimal frequency characteristics for each neuron has been determined, each noise realization preceding each spike can be optimized to generated a matrix $\mathbf{R}$ with a better condition number (*i. e.*, so that the estimate of P is not an ill-posed problem). In the results presented here, the noise frequency was not optimized, but this is a potential approach to improving the results.

The next problem to overcome is how to relate the noise to the change in the spike times. To this end, the noise applied to the neuron over the phase was resampled into b bins. Hence, b determines the resolution of the PRC. Increasing the size of b increases the number of estimated points in the PRC from the same set of data. For each one of the 30 neurons reported in this method, b was discretized using 10 different values. We found that in most of the cells, setting $b > 50$ did not improve the prediction power in the out-of sample data. The measurements of the voltage trace and applied current were obtained at 5 kHz, but the optimal value of b may change with the sampling frequency, firing rate of the neuron and many other variables.

The relationship between the applied noise and the change in the spike time is given by the equation 7.10. The PRC calculated using the pseudo-inverse of $\mathbf{R}$ had high predictive power when tested on in-sample data, but had limited power for out-of-sample data. Using the regularization technique, Truncated Singular Value Decomposition, reduced the predictive power on the in-sample data, but significantly improved the power for out-of-sample data. It can also bee seen that the shape of the estimated PRC using the TSVD method looks more like the PRCs calculated using more traditional methods.

Higher order PRCs that include the stimulus from this and previous ISIs also improved the prediction power. For regular-spiking neurons (coefficient of variation < 0.18) we found high predictive power using one cycle and the addition of the the previous cycle did not significantly add to the predictive power. However, for non-regular spiking neurons, using the information from 2 cycles to calculate the PRC significantly improved the predictive value of the model.

7.3 Conclusion

Most existing methods used to measure PRCs experimentally rely on periodically firing neurons. However, even regular spiking neurons present variations in their period. Furthermore, some neurons have strong accommodation currents, causing changes in their firing rate. This variability in spike time can be seen as an experimental nuisance, or it can be seen as an interesting dynamic of the neuron that should be modeled. A model that can describe the neuron's PRC as firing rate changes can be important in understanding network dynamics during transient conditions.

Here, we have introduced two new methods that allow to extract PRC under varying conditions. The first method presented in this chapter was designed to calculate the PRC as a function of different firing rates. It takes advantage of the fact that the interspike interval can be predicted using an adaptive linear model using the history of the ISI's and stimulus inputs. The PRC can then be measured as the advance of the neuron's spike time from the expected ISI given the stimulus phase. The main contributions of this method is that it allows us to determine the neuron's PRC as a function of the firing rate. This method can easily be extended to estimate the PRC as a function stimulus phase and another variable, like stimulus amplitude.

For neurons that are not regular spikers, we have developed another method of estimating PRCs. This method utilizes the fact that many irregular spiking neurons fire a few reliable spikes at the onset of a step current. These highly reliable spike times can be randomized by adding a noise to the step current. The variability in ISIs caused by the applied noise stimulus across trials can be used to estimate the infinitesimal PRC. However, determining the PRC this way is an ill-posed problem (*i. e.*, it is highly susceptible to noise). By using regularization techniques, PRCs can be estimated that make accurate predictions about the next spike times given the input waveforms. Because short current steps are used to stimulate the neuron, this method can be used to estimate PRCs under transient conditions, such as found during a burst of synaptic inputs as seen in epileptiform activity. We propose to use this method in future studies to understand the effects of anti-epileptic drugs on population synchrony in transient epileptiform activity.

Both methods presented in this chapter could be combined to measure PRCs from regular spiking neurons under transient conditions at different firing rates.

Our final observation from applying these two methods to real neurons is that the higher order PRCs can improve the predictive power of the model. This confirms observations by other groups [101, 102, 103, 104]. The increased modeling accuracy may also improve the sensitivity of this method for detecting changes in neuronal dynamics caused by drugs or other perturbations to the neuron.

References

[1] J. Nolte. *The human brain: an introduction to its functional anatomy.* Mosby Inc, 6th edition edition, 2009.

[2] M.J. Gutnick and I. Mody. *The cortical neuron.* Oxford University Press, USA, 1995.

[3] S.R. Cajal. *Textura del sistema nervioso del hombre y de los vertebrados*, volume 2. Madrid: Imprenta de Nicolás Moya, 1904.

[4] Anti-GAD67 Millipore catalog. http://www.millipore.com/catalogue/item/mab5406.

[5] G. Feng, R.H. Mellor, M. Bernstein, C. Keller-Peck, Q.T. Nguyen, M. Wallace, J.M. Nerbonne, J.W. Lichtman, and J.R. Sanes. Imaging neuronal subsets in transgenic mice expressing multiple spectral variants of GFP. *Neuron*, 28(1):41–51, 2000.

[6] R. Stufflebeam. Neurons, Synapses, Action Potentials, and Neurotransmission. *Consortium on Cognitive Science Instruction*, 2008.

[7] E.M. Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

[8] R. Brette and W. Gerstner. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *Journal of Neurophysiology*, 94(5):3637, 2005.

[9] B. Beverlin II, J. Kakalios, D. Nykamp, and T.I. Netoff. Dynamical changes in neurons during seizures determine tonic to clonic shift. *Journal of Computational Neuroscience*, pages 1–11, 2011.

[10] Charles J Wilson, Bryce Beverlin, and Theoden I Netoff. Chaotic desynchronization as the therapeutic mechanism of deep brain stimulation. *Frontiers in Systems Neuroscience*, 5(0), 2011.

[11] A L Hodgkin and A F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J Physiol*, 117(4):500–44, Aug 1952.

[12] E.M. Izhikevich. Which model to use for cortical spiking neurons? *IEEE transactions on neural networks*, 15(5):1063–1070, 2004.

[13] A.A. Boulton, G.B. Baker, and W. Walz. *Patch-clamp applications and protocols.* Humana Pr Inc, 1995.

[14] DA Brown, BH Gähwiler, WH Griffith, and JV Halliwell. Membrane currents in hippocampal neurons. *Progress in brain research*, 83:141, 1990.

[15] JF Storm. Potassium currents in hippocampal pyramidal cells. *Progress in Brain Research*, 83:161, 1990.

[16] Migliore M Carnevale NT Shepherd GM. Hines ML, Morse T. Modeldb: A database to support computational neuroscience. *J Comput Neurosci.*, Jul-Aug;17(1):7–11, 2004.

[17] D. Golomb and Y. Amitai. Propagating neuronal discharges in neocortical slices: computational and experimental study. *Journal of Neurophysiology*, 78(3):1199, 1997.

[18] D. Hansel and H. Sompolinsky. Chaos and synchrony in a model of a hypercolumn in visual cortex. *Journal of Computational Neuroscience*, 3(1):7–34, 1996.

[19] Astrid A. Prinz. Neuronal parameter optimization. *Scholarpedia*, page 2(1):1903, 2006.

[20] F. Nadim, Ø.H. Olsen, E. Schutter, and R.L. Calabrese. Modeling the leech heartbeat elemental oscillator i. interactions of intrinsic and synaptic currents. *Journal of Computational Neuroscience*, 2(3):215–235, 1995.

[21] A.A. Prinz, C.P. Billimoria, and E. Marder. Alternative to hand-tuning conductance-based models: construction and analysis of databases of model neurons. *Journal of neurophysiology*, 90(6):3998, 2003.

[22] US Bhalla and JM Bower. Exploring parameter space in detailed single neuron models: simulations of the mitral and granule cells of the olfactory bulb. *Journal of neurophysiology*, 69(6):1948, 1993.

[23] M.S. Reid, E.A. Brown, and S.P. DeWeerth. A parameter-space search algorithm tested on a hodgkin–huxley model. *Biological cybernetics*, 96(6):625–634, 2007.

[24] M. Pospischil, M. Toledo-Rodriguez, C. Monier, Z. Piwkowska, T. Bal, Y. Frégnac, H. Markram, and A. Destexhe. Minimal Hodgkin–Huxley type models for different classes of cortical and thalamic neurons. *Biological cybernetics*, 99(4):427–441, 2008.

[25] Rahul Konnur. Synchronization-based approach for estimating all model parameters of chaotic systems. *Phys. Rev. E*, 67(2):027204, Feb 2003.

[26] D. Haufler, F. Morin, JC Lacaille, and FK Skinner. Parameter estimation in single-compartment neuron models using a synchronization-based method. *Neurocomputing*, 70(10-12):1605–1610, 2007.

[27] Henry D. I. Abarbanel, Paul Bryant, Philip E. Gill, Mark Kostuk, Justin Rofeh, Zakary Singer, Bryan Toth, and Elizabeth Wong. Dynamical parameter and state estimation in neuron models. In Mingzhou Ding and Dennis Glanzman, editors, *An Exploration of Neuronal Variability and its Functional Significance*, pages 139–180. Oxford University Press, New York and Oxford, 2011.

[28] Erik P Cook, Jennifer A Guest, Yong Liang, Nicolas Y Masse, and Costa M Colbert. Dendrite-to-soma input/output function of continuous time-varying signals in hippocampal CA1 pyramidal neurons. *J Neurophysiol*, 98(5):2943–55, Nov 2007.

[29] E. Marder and J.M. Goaillard. Variability, compensation and homeostasis in neuron and network function. *Nature Reviews Neuroscience*, 7(7):563–574, 2006.

[30] Adam L Taylor, Jean-Marc Goaillard, and Eve Marder. How multiple conductances determine electrophysiological properties in a multicompartment model. *J Neurosci*, 29(17):5573–86, Apr 2009.

[31] M.S. Goldman, J. Golowasch, E. Marder, and LF Abbott. Global structure, robustness, and modulation of neuronal models. *The Journal of Neuroscience*, 21(14):5229, 2001.

[32] E. Hong, G. Kazanci, et al. Different roles of related currents in fast and slow spiking of model neurons from two phyla. *Journal of neurophysiology*, 100(4):2048, 2008.

[33] P. Sajda, K.R. Muller, and K.V. Shenoy. Brain-computer interfaces [from the guest editors]. *Signal Processing Magazine, IEEE*, 25(1):16–17, 2008.

[34] J.R. Wolpaw, N. Birbaumer, D.J. McFarland, G. Pfurtscheller, and T.M. Vaughan. Brain-computer interfaces for communication and control. *Clinical neurophysiology*, 113(6):767–791, 2002.

[35] L.R. Hochberg and J.P. Donoghue. Sensors for brain-computer interfaces. *Engineering in Medicine and Biology Magazine, IEEE*, 25(5):32 –38, sept.-oct. 2006.

[36] R.B. Northrop. *Introduction to instrumentation and measurements.* CRC Pr I Llc, 1997.

[37] E.O. Doebelin. *Measurement systems: application and design.* McGraw-Hill New York;, 4th edition, 1990.

[38] C. Torrence and G.P. Compo. A practical guide to wavelet analysis. *Bulletin of the American Meteorological Society*, 79(1):61–78, 1998.

[39] P. Mitra and H. Bokil. *Observed brain dynamics.* Oxford University Press, USA, 2008.

[40] G. Alarcon, CD Binnie, RDC Elwes, and CE Polkey. Power spectrum and intracranial eeg patterns at seizure onset in partial epilepsy. *Electroencephalography and clinical neurophysiology*, 94(5):326–337, 1995.

[41] A. Bragin, J. Engel Jr, C.L. Wilson, I. Fried, and G.W. Mathern. Hippocampal and entorhinal cortex high-frequency oscillations (100–500 hz) in human epileptic brain and in kainic acid-treated rats with chronic seizures. *Epilepsia*, 40(2):127–137, 1999.

[42] A.B. Gardner, G.A. Worrell, E. Marsh, D. Dlugos, and B. Litt. Human and automated detection of high-frequency oscillations in clinical intracranial eeg recordings. *Clinical neurophysiology*, 118(5):1134–1143, 2007.

[43] G. Worrell and J. Gotman. High-frequency oscillations and other electrophysiological biomarkers of epilepsy: clinical studies. *Biomarkers*, 5(5):557–566, 2011.

[44] E.C. Leuthardt, G. Schalk, J. Roland, A. Rouse, and D.W. Moran. Evolution of brain-computer interfaces: going beyond classic motor physiology. *Neurosurgical focus*, 27(1):E4, 2009.

[45] Willsky A.S. Oppenheim, A.V. *Signals and systems*. Prentice-Hall signal processing series. Prentice Hall, 1997.

[46] J.G. Proakis and D.G. Manolakis. *Digital signal processing: principles, algorithms, and applications*, volume 3. Prentice Hall Upper Saddle River, NJ:, 1996.

[47] E.O. Doebelin. *System dynamics: modeling, analysis, simulation, design*. Marcel Dekker, 1998.

[48] A. Budak. *Passive and active network analysis and synthesis*. Waveland Press, 1991.

[49] A.D. Dorval, D.J. Christini, and J.A. White. Real-time linux dynamic clamp: a fast and flexible way to construct virtual ion channels in living cells. *Annals of biomedical engineering*, 29(10):897–907, 2001.

[50] http://www.rtxi.org/topics/modules/.

[51] J. Glover Jr. Adaptive noise canceling applied to sinusoidal interferences. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 25(6):484–491, 1977.

[52] B. H. Kuo, S. M. Lee and Wenshun Tian. *Real-Time Digital Signal Processing.* Wiley, 2006.

[53] Simon Haykin. *Adaptive Filter Theory.* Prentice Hall, 4th edition edition, 2002.

[54] D.O. Olguín, FB Lara, and SO Martínez-Chapa. Adaptive notch filter for EEG signals based on the LMS algorithm with variable step-size parameter. In *Proceedings of the 39th International Conference on Information Sciences and Systems.* Citeseer, 2005.

[55] A. Capasso, A. Cavallini, AE Emanuel, M. Halpin, A. Imece, A. Ludbrook, G. Montanari, KJ Olejniczak, P. Ribeiro, S. Rios-Marcuello, et al. Time-varying harmonics: Part I-characterizing measured data. *IEEE Transactions on Power Delivery*, 13(3), 1998.

[56] Katsuhiko Ogata. *Modern control engineering.* Prentice Hall, Upper Saddle River, N.J., 3rd ed edition, 1997.

[57] Ernest O Doebelin. *Measurement systems: application and design.* McGraw-Hill, Boston, 5th ed edition, 2004.

[58] K. Gopal Gopalan. *An introduction to signal and system analysis.* Cengage Learning, Toronto, ON, 2009.

[59] John G Proakis, Dimitris G Manolakis, and John G Proakis. *Digital signal processing: principles, algorithms, and applications.* Macmillan, New York, 2nd ed edition, 1992.

[60] G Bard Ermentrout, Roberto F Galán, and Nathaniel N Urban. Relating neural dynamics to neural coding. *Phys Rev Lett*, 99(24):248103, Dec 2007.

[61] T.I. Netoff, M.I. Banks, A.D. Dorval, C.D. Acker, J.S. Haas, N. Kopell, and J.A. White. Synchronization in hybrid neuronal networks of the hippocampal formation. *Journal of neurophysiology*, 93(3):1197, 2005.

[62] Steven J. Schiff. *Neural Control Engineering: The Emerging Intersection Between Control Theory and Neuroscience.* The MIT Press, 2011.

[63] S.J. Julier, J.K. Uhlmann, and H.F. Durrant-Whyte. A new approach for filtering nonlinear systems. In *American Control Conference, 1995. Proceedings of the*, volume 3, pages 1628–1632. IEEE, 1995.

[64] S.J. Julier and J.K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, volume 3, page 26. Citeseer, 1997.

[65] D. Simon. *Optimal state estimation: Kalman, H_∞ and nonlinear approaches.* Wiley-Interscience, 2006.

[66] Henning U. Voss and Jens Timmer. Nonlinear dynamical system identification from uncertain and indirect measurements. *International Journal of Bifurcation and Chaos*, 14(6):1905–1933, 2004.

[67] G. Ullah and S.J. Schiff. Tracking and control of neuronal Hodgkin-Huxley dynamics. *Physical Review E*, 79(4):40901, 2009.

[68] E.M. Izhikevich. *Dynamical systems in neuroscience: The geometry of excitability and bursting.* The MIT press, 2007.

[69] C.K. Wikle and L.M. Berliner. A Bayesian tutorial for data assimilation. *Physica D: Nonlinear Phenomena*, 230(1-2):1–16, 2007.

[70] R. Pozo. Template numerical toolkit for linear algebra: High performance programming with c++ and the standard template library. *International Journal of High Performance Computing Applications*, 11(3):251–263, 1997.

[71] R. Kandepu, L. Imsland, and B.A. Foss. Constrained state estimation using the unscented kalman filter. In *Control and Automation, 2008 16th Mediterranean Conference on*, pages 1453–1458. IEEE, 2008.

[72] J.M. Bower, D. Beeman, and A.M. Wylde. *The book of GENESIS: exploring realistic neural models with the GEneral NEural SImulation System.* Springer New York:, 1998.

[73] B.W. Connors and M.J. Gutnick. Intrinsic firing patterns of diverse neocortical neurons. *Trends in neurosciences*, 13(3):99–104, 1990.

[74] H.L. Bryant and J.P. Segundo. Spike initiation by transmembrane current: a white-noise analysis. *The Journal of physiology*, 260(2):279, 1976.

[75] Z.F. Mainen and T.J. Sejnowski. Reliability of spike timing in neocortical neurons. *Science*, 268(5216):1503, 1995.

[76] J.S. Haas and J.A. White. Frequency selectivity of layer ii stellate cells in the medial entorhinal cortex. *Journal of neurophysiology*, 88(5):2422–2429, 2002.

[77] O.C. Zalay, D. Serletis, P.L. Carlen, and B.L. Bardakjian. System characterization of neuronal excitability in the hippocampus and its relevance to observed dynamics of spontaneous seizure-like transitions. *Journal of neural engineering*, 7:036002, 2010.

[78] U. Lu, D. Song, and T. Berger. Nonlinear dynamic modeling of synaptically driven single hippocampal neuron intracellular activity. *Biomedical Engineering, IEEE Transactions on*, (99):1–1, 2011.

[79] S.R. Williams and S.J. Mitchell. Direct measurement of somatic voltage clamp errors in central neurons. *Nature neuroscience*, 11(7):790–798, 2008.

[80] P.J. Uhlhaas and W. Singer. Neural synchrony in brain disorders: relevance for cognitive dysfunctions and pathophysiology. *Neuron*, 52(1):155–168, 2006.

[81] A.D. Reyes and E.E. Fetz. Effects of transient depolarizing potentials on the firing rate of cat neocortical neurons. *Journal of neurophysiology*, 69(5):1673–1683, 1993.

[82] B.S. Gutkin, G.B. Ermentrout, and A.D. Reyes. Phase-response curves give the responses of neurons to transient inputs. *Journal of neurophysiology*, 94(2):1623–1635, 2005.

[83] T.I. Netoff and S.J. Schiff. Decreased neuronal synchronization during experimental seizures. *The Journal of neuroscience*, 22(16):7297–7307, 2002.

[84] B. Ermentrout. *Simulating, analyzing, and animating dynamical systems: a guide to XPPAUT for researchers and students*, volume 14. Society for Industrial Mathematics, 2002.

[85] B. Torben-Nielsen, M. Uusisaari, and K.M. Stiefel. A comparison of methods to determine neuronal phase-response curves. *Frontiers in neuroinformatics*, 4, 2010.

[86] Ó Miranda-Domínguez, J Gonia, and T I Netoff. Firing rate control of a neuron using a linear proportional-integral controller. *J Neural Eng*, 7(6):066004, Dec 2010.

[87] R.F. Galán, G.B. Ermentrout, and N.N. Urban. Efficient estimation of phase-resetting curves in real neurons and its significance for neural-network modeling. *Physical review letters*, 94(15):158101, 2005.

[88] Keisuke Ota, Toshiaki Omori, and Toru Aonishi. MAP estimation algorithm for phase response curves based on analysis of the observation process. *Journal of Computational Neuroscience*, 26:185–202, 2009.

[89] Kaiichiro Ota, Masaki Nomura, and Toshio Aoyagi. A Weighted Spike-Triggered Average of a Fluctuating Stimulus Yielding the Phase Response Curve. *Physical Review Letters*, 103(2):5, 2008.

[90] B. Torben-Nielsen, M. Uusisaari, and K.M. Stiefel. A novel method for determining the phase-response curves of neurons based on minimizing spike-time prediction error. *Arxiv preprint arXiv:1001.0446*, 2010.

[91] R. Rosipal and N. Krämer. Overview and recent advances in partial least squares. *Subspace, Latent Structure and Feature Selection*, pages 34–51, 2006.

[92] S. de Jong. Simpls: an alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 18(3):251–263, 1993.

[93] C.G. Fink, V. Booth, and M. Zochowski. Effects of the frequency dependence of phase response curves on network synchronization. *Phase Response Curves in Neuroscience*, pages 475–487, 2012.

[94] T. Netoff, M.A. Schwemmer, and T.J. Lewis. Experimentally estimating phase response curves of neurons: Theoretical and practical issues. *Phase Response Curves in Neuroscience*, pages 95–129, 2012.

[95] G.B. Ermentrout, R.F. Galán, and N.N. Urban. Reliability, synchrony and noise. *Trends in neurosciences*, 31(8):428–434, 2008.

[96] P.C. Hansen. The truncated svd as a method for regularization. *BIT Numerical Mathematics*, 27(4):534–553, 1987.

[97] P.C. Hansen. Regularization tools: A matlab package for analysis and solution of discrete ill-posed problems. *Numerical algorithms*, 6(1):1–35, 1994.

[98] Bruce Hutcheon and Yosef Yarom. Resonance, oscillation and the intrinsic frequency preferences of neurons. *Trends in Neurosciences*, 23(5):216 – 222, 2000.

[99] Nicolas Brunel, Frances S. Chance, Nicolas Fourcaud, and L. F. Abbott. Effects of synaptic noise and filtering on the frequency response of spiking neurons. *Phys. Rev. Lett.*, 86:2186–2189, Mar 2001.

[100] A. Nabi and J. Moehlis. Charge-balanced optimal inputs for phase models of spiking neurons. In *Proceedings of 2009 ASME Dynamic Systems and Control Conference*, 2009.

[101] SA Oprisan and CC Canavier. Stability analysis of rings of pulse-coupled oscillators: the effect of phase resetting in the second cycle after the pulse is important at synchrony and for long pulses. *Differential Equations and Dynamical Systems*, 9(3-4):243–258, 2001.

[102] SA Oprisan, AA Prinz, and CC Canavier. Phase resetting and phase locking in hybrid circuits of one model and one biological neuron. *Biophysical journal*, 87(4):2283–2298, 2004.

[103] J. Cui, C.C. Canavier, and R.J. Butera. Functional phase response curves: a method for understanding synchronization of adapting neurons. *Journal of neurophysiology*, 102(1):387–398, 2009.

[104] S. Achuthan and C.C. Canavier. Phase-resetting curves determine synchronization, phase locking, and clustering in networks of neural oscillators. *The Journal of Neuroscience*, 29(16):5218–5233, 2009.

Appendix A

Experimental preparation

All experiments were performed following approved guidelines from Research Animal Resources of the University of Minnesota. Long Evans rats 14-21 days old of either sex were deeply anesthetized with isofluorane and decapitated. The brains were removed and placed into chilled artificial cerebral spinal fluid (ACSF) containing (in mM): 126 $NaCl$, 25 $NaHCO_3$, 25 glucose, 1.25 $NaH_2PO_4 \cdot H_2O$, 2.5 KCl, 1 $MgCl$, and 2 $CaCl_2$. Slices of 400 μm thickness were cut using a vibratome and perfused with ACSF. The slices were aerated with O_2 and kept at 37° C until use.

Patch clamp electrodes (6-10 $M\Omega$) were pulled from borosilicate glass (Micro-pipette Puller: Sutter Instrument P-97) and filled with (in mM) 120 $K-gluconate$, 20 KCl, 10 $Hepes$, 0.2 $EGTA$, 2 $MgCl_2$, 4 Na_2-ATP, 0.3 $Tris-GTP$ and 7 $Na_2-phoscreatine$.

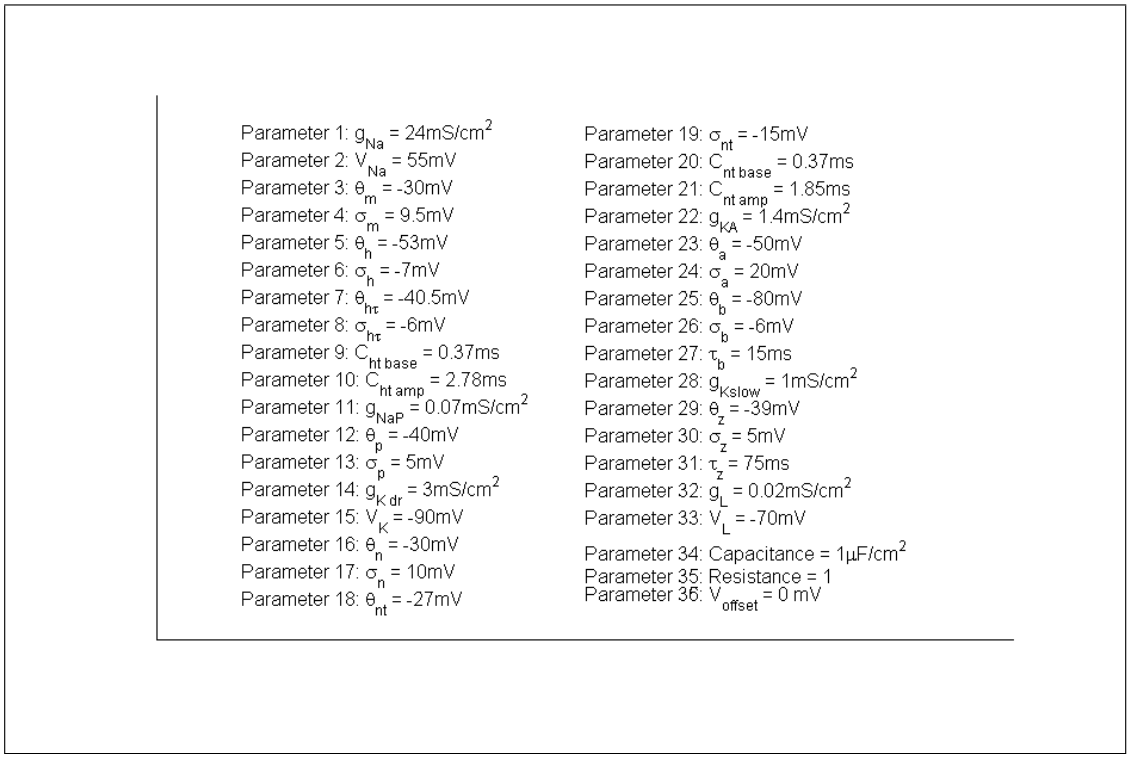
Pyramidal cells in the hippocampal formation were whole-cell patched and recorded using a current clamp amplifier (Axon Instruments, MultiClamp 700B). Data acquisition, was implemented using a real-time Linux-based system [49] called RTXI (Real-Time eXperiment Interface), sampling at 8 kHz. RTXI allows implementation of complex closed-loop protocols running in hard real-time and is publicly available (`http://www.rtxi.org`).

Appendix B

GACell

Here is described the computational model called “gacell”. The reference for this section is [17].

Next table shows the parameters and their default values:



Parameter 1: $g_{Na} = 24mS/cm^2$	Parameter 19: $\sigma_{nt} = -15mV$
Parameter 2: $V_{Na} = 55mV$	Parameter 20: $C_{nt\ base} = 0.37ms$
Parameter 3: $\theta_m = -30mV$	Parameter 21: $C_{nt\ amp} = 1.85ms$
Parameter 4: $\sigma_m = 9.5mV$	Parameter 22: $g_{KA} = 1.4mS/cm^2$
Parameter 5: $\theta_h = -53mV$	Parameter 23: $\theta_a = -50mV$
Parameter 6: $\sigma_h = -7mV$	Parameter 24: $\sigma_a = 20mV$
Parameter 7: $\theta_{h\tau} = -40.5mV$	Parameter 25: $\theta_b = -80mV$
Parameter 8: $\sigma_{h\tau} = -6mV$	Parameter 26: $\sigma_b = -6mV$
Parameter 9: $C_{ht\ base} = 0.37ms$	Parameter 27: $\tau_b = 15ms$
Parameter 10: $C_{ht\ amp} = 2.78ms$	Parameter 28: $g_{Kslow} = 1mS/cm^2$
Parameter 11: $g_{NaP} = 0.07mS/cm^2$	Parameter 29: $\theta_z = -39mV$
Parameter 12: $\theta_p = -40mV$	Parameter 30: $\sigma_z = 5mV$
Parameter 13: $\sigma_p = 5mV$	Parameter 31: $\tau_z = 75ms$
Parameter 14: $g_{K\ dr} = 3mS/cm^2$	Parameter 32: $g_L = 0.02mS/cm^2$
Parameter 15: $V_K = -90mV$	Parameter 33: $V_L = -70mV$
Parameter 16: $\theta_n = -30mV$	Parameter 34: Capacitance = $1\mu F/cm^2$
Parameter 17: $\sigma_n = 10mV$	Parameter 35: Resistance = 1
Parameter 18: $\theta_{nt} = -27mV$	Parameter 36: $V_{offset} = 0\ mV$

The single-cell dynamics is described by a single-compartment Hodgkin-Huxley type model with the use of a set of coupled differential equations.

B.1 Current balance equation

$$C\frac{d}{dt}V(t) = -I_{Na}(V,h) - I_{NaP}(V) - I_{Kdr}(V,n) - I_{KA}(V,b) - I_{K,slow}(V,z) - I_L(V) + I_{app} \tag{B.1}$$

$C = 1\frac{\mu F}{cm^2}$ where C is the membrane capacitance and $V(t)$ is the membrane potential of a neuron at time t. The right side incorporates an applied current I_{app}, and the following intrinsic currents: the fast sodium current I_{Na}, the persistent sodium current I_{NaP}, the delayed-rectifier potassium current I_{Kdr}, the A-type potassium current I_{KA}, the slow potassium current $I_{K,slow}$, and the leak current I_L.

The slow potassium current in the model represents potassium currents with kinetics slower than the action potential time scale, *i. e.* with activation time in the order of several tens to hundred of milliseconds. These currents are either calcium dependent, such as I_C and I_{AHP}, or voltage dependent, such as I_M. They are responsible for the adaptation in regular spiking cortical cells

B.2 Intrinsic currents

B.2.1 Sodium current I_{Na}

$$I_{Na}(V,h) = g_{Na}m_\infty^3 h (V - V_{Na}) \tag{B.2a}$$

$$\frac{d}{dt}h = \frac{h_\infty - h}{\tau_h} \tag{B.2b}$$

$$m_\infty = \frac{1}{1 + e^{-\frac{V-\theta_m}{\sigma_m}}} \tag{B.2c}$$

$$h_\infty = \frac{1}{1 + e^{-\frac{V-\theta_h}{\sigma_h}}} \tag{B.2d}$$

$$\tau_h = 0.37 + 2.78\frac{1}{1 + e^{-\frac{V-\theta_{ht}}{\sigma_{ht}}}} \tag{B.2e}$$

$g_{Na} = 24\ \frac{mS}{cm^2}$, $V_{Na} = 55\ mV$, $\theta_m = -30\ mV$, $\sigma_m = 9.5\ mV$, $\theta_h = -53\ mV$, $\sigma_h = -7\ mV$, $\theta_{ht} = -40.5\ mV$, $\sigma_{ht} = -6\ mV$. The function h_∞ is shifted to the right on the voltage axis to enable the model to replicate the current-clamp current injection experiments. Gating kinetics has been adjusted to $36°C$.

B.2.2 Sodium current I_{NaP}

$$I_{NaP}(V,h) = g_{NaP}P_{\infty}(V - V_{Na}) \tag{B.3a}$$

$$p_{\infty} = \frac{1}{1 + e^{-\frac{V-\theta_p}{\sigma_p}}} \tag{B.3b}$$

$g_{NaP} = 0.07\ \frac{mS}{cm^2}$, $\theta_p = -40\ mV$, $\sigma_p = 5.0\ mV$.

B.2.3 Delayed rectifier potassium current I_{Kdr}

$$I_{Kdr}(V,n) = g_{Kdr}n^4(V - V_K) \tag{B.4a}$$

$$\frac{d}{dt}n = \frac{n_{\infty} - n}{\tau_n} \tag{B.4b}$$

$$n_{\infty} = \frac{1}{1 + e^{-\frac{V-\theta_n}{\sigma_n}}} \tag{B.4c}$$

$$\tau_n = 0.37 + 1.85\frac{1}{1 + e^{-\frac{V-\theta_{nt}}{\sigma_{nt}}}} \tag{B.4d}$$

$g_{Kdr} = 3\ \frac{mS}{cm^2}$, $V_K = -90\ mV$, $\theta_n = -30\ mV$, $\sigma_n = 10\ mV$, $\theta_{nt} = -27\ mV$, $\sigma_{nt} = -15\ mV$.

B.2.4 A-type potassium current I_{KA}

$$I_{KA}(V,b) = g_{KA}a_{\infty}^3 b(V - V_K) \tag{B.5a}$$

$$\frac{d}{dt}b = \frac{b_{\infty} - b}{\tau_b} \tag{B.5b}$$

$$a_{\infty} = \frac{1}{1 + e^{-\frac{V-\theta_a}{\sigma_a}}} \tag{B.5c}$$

$$b_{\infty} = \frac{1}{1 + e^{-\frac{V-\theta_b}{\sigma_b}}} \tag{B.5d}$$

$g_{KA} = 1.4\ \frac{mS}{cm^2}$, $\theta_a = -50\ mV$, $\sigma_a = 20\ mV$, $\theta_b = -80\ mV$, $\sigma_b = -6\ mV$, $\tau_b = 15\ ms$.

B.2.5 Slow potassium current I_{Kslow}

$$\begin{aligned} I_{KSlow}(V,z) &= g_{KSlow} z (V - V_K) && \text{(B.6a)} \\ \frac{d}{dt} z &= \frac{z_\infty - z}{\tau_z} && \text{(B.6b)} \\ z_\infty &= \frac{1}{1 + e^{-\frac{V-\theta_z}{\sigma_z}}} && \text{(B.6c)} \end{aligned}$$

$g_{KSlow} = 1 \ \frac{mS}{cm^2}$, $\theta_z = -39 \ mV$, $\sigma_z = 5 \ mV$, $\tau_z = 75 \ ms$.

B.2.6 Leak current I_L

$$I_L(V) = g_L (V - V_L) \tag{B.7}$$

$g_L = 0.02 \ \frac{mS}{cm^2}$, $V_L = -70 \ mV$

Appendix C

Linear and Unscented Kalman Filter formulation

C.1 The discrete-time Kalman filter

The original formulation can be found in [65].

The next table shows the definitions

	Kalman filtering
$\hat{\overrightarrow{x}}_k^-$	*a priori* estimate
$\hat{\overrightarrow{x}}_k^+$	*a posteriori* estimate
P_k^-	*a priori* covariance
P_k^+	*a posteriori* covariance

Here we summarize the discrete-time Kalman filter by combining the above equations into a single algorithm.

1. The dynamic system is given by the following equations:

$$\overrightarrow{x}_k = F_{k-1}\overrightarrow{x}_{k-1} + G_{k-1}\overrightarrow{u}_{k-1} + \overrightarrow{w}_{k-1} \tag{C.1a}$$

$$\overrightarrow{y}_k = H_k\overrightarrow{x}_k + \overrightarrow{v}_k \tag{C.1b}$$

$$E\left(\overrightarrow{w}_k\overrightarrow{w}_j^T\right) = Q_k\delta_{k-j} \tag{C.1c}$$

$$E\left(\overrightarrow{v}_k\overrightarrow{v}_j^T\right) = R_k\delta_{k-j} \tag{C.1d}$$

$$E\left(\overrightarrow{w}_k\overrightarrow{v}_j^T\right) = 0 \tag{C.1e}$$

2. The kalman filter is initialized as follows

$$\begin{aligned} \overrightarrow{\hat{x}}_0^+ &= E\left(\overrightarrow{x}_0\right) && \text{(C.2a)} \\ P_0^+ &= E\left[\left(\overrightarrow{x}_0 - \overrightarrow{\hat{x}}_0^+\right)\left(\overrightarrow{x}_0 - \overrightarrow{\hat{x}}_0^+\right)^T\right] && \text{(C.2b)} \end{aligned}$$

   If we know the initial state perfectly, then $P_0^+ = 0$. If we have no idea of the value of $\overrightarrow{x}_0$, then $P_0^+ = \infty I$. In general, P_0^+ represents the uncertainty in our initial estimate of $\overrightarrow{x}_0$.

3. The Kalman filter is given by the following equations, which are computed for each time step $k = 1, 2, \ldots$

$$\begin{aligned} P_k^- &= F_{k-1}P_{k-1}^+F_{k-1}^T + Q_{k-1} && \text{(C.3a)} \\ K_k &= P_k^- H_k^T \left(H_k P_k^- H_k^T + R_k\right)^{-1} && \text{(C.3b)} \\ &= P_k^+ H_k^T R_k^{-1} && \text{(C.3c)} \\ \overrightarrow{\hat{x}}_k^- &= F_{k-1}^- \overrightarrow{\hat{x}}_{k-1} + G_{k-1}\overrightarrow{u}_{k-1} \text{ \textit{a priori} state estimate} && \text{(C.3d)} \\ \overrightarrow{\hat{x}}_k^+ &= \widehat{\overrightarrow{x}}_k^- + K_k\left(\overrightarrow{y}_k - H_k\overrightarrow{\hat{x}}_k^-\right) \text{ \textit{a posteriori} state estimate} && \text{(C.3e)} \\ P_k^+ &= \left(I - K_kH_k\right)P_k^-\left(I - K_kH_k\right)^T + K_kR_kK_k^T && \text{(C.3f)} \\ &= \left[\left(P_k^-\right)^{-1} + H_k^T R_k^{-1} H_k\right]^{-1} && \text{(C.3g)} \\ &= \left(I - K_kH_kP_k^-\right) && \text{(C.3h)} \end{aligned}$$

The first expression for P_k^+ above is called the Joseph stabilized version of the covariance measurement update equation. It was formulated by Peter Joseph in 1960s and can be shown to be more stable and robust than the third expression for P_k^+. The first expression for P_k^+ guarantees that P_k^+ will always be symmetric positive definite, as lon as P_k^- is symmetric positive definite. The third expression for P_k^+ is computationally simpler than the first expression, but its form does not guarantee symmetry or positive definiteness for P_k^+. The second form for P_k^+ is rarely implemented as written above.

If we know the initial state perfectly, then $P_0^+ = 0$. If we have no idea of the value of $\overrightarrow{x}_0$, then $P_0^+ = \infty I$. In general, P_0^+ represents the uncertainty in our initial estimate of $\overrightarrow{x}_0$.

The calculation of P_k^-, K_k, and P_k^+ does not depend on the measurements $\overrightarrow{y}_k$, but depends only on the system parameters F_x, H_k, Q_k, and R_k. That means that the Kalman K_k gain can be calculated offline before the systems operates and saved in memory. In contrast, the filter gain and covariance for nonlinear systems cannot (in general) be computed offline because they depend on the measurements.

When the Kalman filter is implemented on a real system it may not work, even though the theory is correct. Two of the primary causes for the failure of Kalman filtering are the finite precision arithmetic and modeling errors. The theory presented assumes that the system model is precisely known. It is assumed that the F, Q, H, and R matrices are exactly known, and it is assumed that the noise sequences $\{w_k\}$ and $\{v_k\}$ are pure white, zero mean, and completely uncorrelated. If any of these assumptions are violated, as they always are in real implementations, then the Kalman filter assumptions are violated and the theory may not work.

In order to improve the filter performance in the face of these realities, the designer can use several strategies:

1. Increase arithmetic precision.
2. Use some form of square root filtering.
3. Symmetrize P at each time step $P = \left(P + P^T\right)/2$
4. Initialize P appropriately to avoid large changes in P
5. Use a fading memory filter
6. Use fictious process noise (especially for estimating "constants").

C.2 The Unscented Kalman Filter

The original formulation can be found in [65].

1. We have an n-state discrete-time nonlinear system given by

$$\overrightarrow{x}_{k+1} = f\left(\overrightarrow{x}_k, \overrightarrow{u}_k, t_k\right) + \overrightarrow{w}_k \tag{C.4a}$$

$$\overrightarrow{y}_k = h\left(\overrightarrow{x}_k, t_k\right) + \overrightarrow{v}_k \tag{C.4b}$$

$$w_k \sim (0, Q_k) \tag{C.4c}$$

$$v_k \sim (0, R_k) \tag{C.4d}$$

2. The UKF is initialized as follows

$$\begin{aligned} \overrightarrow{\hat{x}}_0^+ &= E\left(\overrightarrow{x}_0\right) && \text{(C.5a)} \\ P_0^+ &= E\left[\left(\overrightarrow{x}_0 - \overrightarrow{\hat{x}}_0^+\right)\left(\overrightarrow{x}_0 - \overrightarrow{\hat{x}}_0^+\right)^T\right] && \text{(C.5b)} \end{aligned}$$

3. The following time update equations are used to propagate the state estimate and covariance from one measurement time to the next

   (a) To propagate from time step $(k-1)$ to k, first choose sigma points $\overrightarrow{x}_{k-1}^{(i)}$

$$\begin{aligned} \overrightarrow{\hat{x}}_{k-1}^{(i)} &= \overrightarrow{\hat{x}}_{k-1}^+ + \overrightarrow{\tilde{x}}^{(i)},\ i = 1, \ldots, 2n && \text{(C.6a)} \\ \overrightarrow{\tilde{x}}^{(i)} &= \left(\sqrt{nP_{k-1}^+}\right)_i^T,\ i = 1, \ldots, n && \text{(C.6b)} \\ \overrightarrow{\tilde{x}}^{(n+i)} &= -\left(\sqrt{nP_{k-1}^+}\right)_i^T,\ i = 1, \ldots, n && \text{(C.6c)} \end{aligned}$$

   (b) Use the known nonlinear system equation f to transform the sigma points into $\overrightarrow{\hat{x}}_k^{(i)}$

$$\overrightarrow{\hat{x}}_k^{(i)} = f\left(\overrightarrow{\hat{x}}_{k-1}^{(i)}, \overrightarrow{u}_k, t_k\right) \tag{C.7}$$

   (c) Combine the $\overrightarrow{\hat{x}}_k^{(i)}$ vectors to obtain the *a priori* state estimation at time k

$$\overrightarrow{\hat{x}}_k^- = \frac{1}{2n}\sum_{i=1}^{2n} \overrightarrow{\hat{x}}_k^{(i)} \tag{C.8}$$

   (d) Estimate the *a priori* error covariance

$$P_k^- = \frac{1}{2n}\sum_{i=1}^{2n}\left[\left(\overrightarrow{\hat{x}}_k^{(i)} - \overrightarrow{\hat{x}}_k^-\right)\left(\overrightarrow{\hat{x}}_k^{(i)} - \overrightarrow{\hat{x}}_k^-\right)^T\right] + Q_{k-1} \tag{C.9}$$

4. Now that the time update equations are done, we implement the measurement update equations

   (a) Choose sigma points $\overrightarrow{\hat{x}}_k^{(i)}$

$$\begin{aligned} \overrightarrow{\hat{x}}_k^{(i)} &= \overrightarrow{\hat{x}}_k^- + \overrightarrow{\tilde{x}}^{(i)},\ i = 1, \ldots, 2n && \text{(C.10a)} \\ \overrightarrow{\tilde{x}}^{(i)} &= \left(\sqrt{nP_k^-}\right)_i^T,\ i = 1, \ldots, n && \text{(C.10b)} \\ \overrightarrow{\tilde{x}}^{(n+i)} &= -\left(\sqrt{nP_k^+}\right)_i^T,\ i = 1, \ldots, n && \text{(C.10c)} \end{aligned}$$

This step can be omitted if desired. That is, instead of generating new sigma points we can reuse the sigma points that were obtained from the time update. This will save computational effort.

(b) Use the known nonlinear measurement equation h to transform the sigma points into $\vec{y}_k^{(i)}$ vectors (predicted measurements)

$$\vec{y}_k^{(i)} = h\left(\vec{x}_k^{(i)}, t_k\right) \tag{C.11}$$

(c) Combine the $\vec{y}_k^{(i)}$ vectors to obtain the predicted measurement at time k

$$\vec{y}_k = \frac{1}{2n}\sum_{i=1}^{2n} \vec{y}_k^{(i)} \tag{C.12}$$

(d) Estimate the covariance of the predicted measurement and add the noise R

$$P_y = \frac{1}{2n}\sum_{i=1}^{2n}\left[\left(\vec{y}_k^{(i)} - \vec{y}_k\right)\left(\vec{y}_k^{(i)} - \vec{y}_k\right)^T\right] + R_k \tag{C.13}$$

(e) Estimate the cross covariance between the $\vec{x}_k^-$ and $\vec{y}_k$

$$P_{xy} = \frac{1}{2n}\sum_{i=1}^{2n}\left[\left(\vec{x}_k^{(i)} - \vec{x}_k^-\right)\left(\vec{y}_k^{(i)} - \vec{y}_k\right)^T\right] + R_k \tag{C.14}$$

(f) The measurement update of the state estimate can be performed using the normal Kalman filter equations

$$\begin{aligned} K_k &= P_{xy}P_y^{-1} && \text{(C.15a)}\\ \vec{x}_k^+ &= \vec{x}_k^- + K_k\left(\vec{y}_k - \vec{y}_k\right) && \text{(C.15b)}\\ P_k^+ &= P_k^- - K_kP_yK_k^T && \text{(C.15c)} \end{aligned}$$

The algorithm above assumes that the process and measurement equations are linear with respect to the noise, as shown in EQ(14.65). In general, the process and measurement equations may have noise that enters the process and measurement equation nonlinearly. That is

$$\begin{aligned} \vec{x}_{k+1} &= f\left(\vec{x}_k, \vec{u}_k, \vec{w}_k, t_k\right) && \text{(C.16a)}\\ \vec{y}_k &= h\left(\vec{x}_k, \vec{v}_k, t_k\right) && \text{(C.16b)} \end{aligned}$$

In this case, the UKF presented above is not rigorous because it treats the noise as additive. To handle this situation, we can augment the noise into the state vector

$$\vec{x}_k^a = \begin{bmatrix} \vec{x}_k \\ \vec{w}_k \\ \vec{v}_k \end{bmatrix} \tag{C.17}$$

Then, we can use the UKF to estimate the augmented state $\vec{x}_k^a$. The UKF is initialized as

$$\vec{x}_k^{a+} = \begin{bmatrix} E\left(\vec{x}_0\right) \\ \vec{0} \\ \vec{0} \end{bmatrix} \tag{C.18a}$$

$$P_0^{a+} = \begin{bmatrix} E\left[\left(\vec{x}_0 - \hat{\vec{x}}_0^+\right)\left(\vec{x}_0 - \hat{\vec{x}}_0^+\right)^T\right] & 0 & 0 \\ 0 & Q_0 & 0 \\ 0 & 0 & R_0 \end{bmatrix} \tag{C.18b}$$

Then we use the UKF algorithm presented above, except that we are estimating the augmented mean and covariance, so we remove Q_{k-1} and R_k from equations C.9 (page 139) and C.13 (page 140).

Appendix D

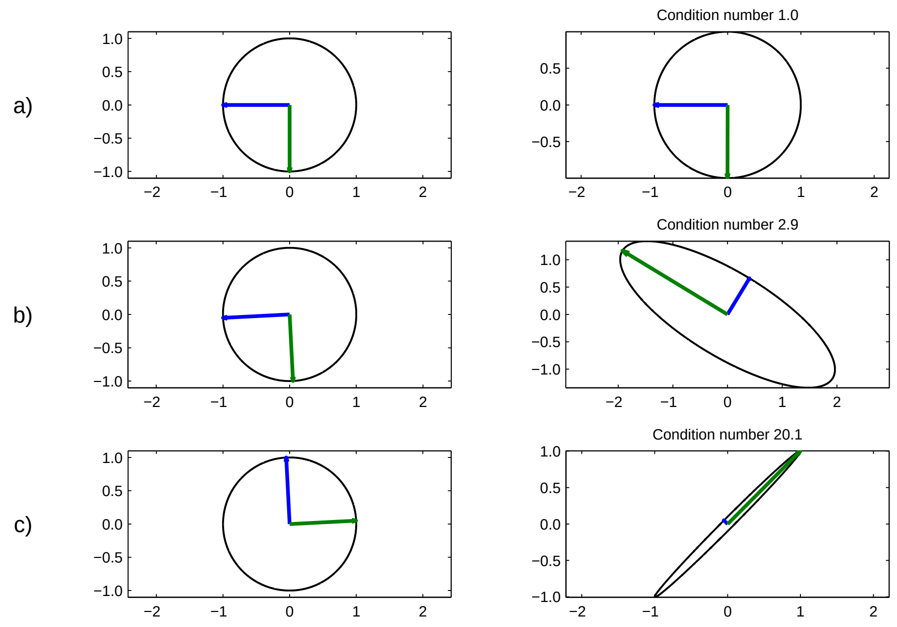
Truncated singular value decomposition method

This appendix describes the truncated singular value decomposition method. The content is based on [96, 97] A linear system

$$\overrightarrow{y} = \mathbf{A}\overrightarrow{x} \tag{D.1}$$

is ill-posed if small perturbations in $\overrightarrow{x}$ generates a significant change on $\overrightarrow{y}$ [97]. This is easy to visualize in a 2 dimensional system. Lets consider the family of vectors of length 1 (an unitary circle) and transform them using equation D.1. The resulting figure can be another circle, an ellipse or a singularity (a straight line), as shown in figure D.1. A miss-estimation of $\overrightarrow{x}$ can be dramatically enhanced by a ill-conditioned matrix, *i.* e. vectors that are close together in any region in the unitary circle could be far away when they are transformed using equation D.1. The condition number of **A** is obtained as the ratio of the longer to the shorter semi-axis of the resulting ellipsis. The length of the semi-axis are the singular values of the matrix. A well conditioned matrix has a condition number close to one, while a large condition number implies an ill-conditioned matrix.

Here, **R** generates an ill-posed problem, so the PRC estimation is highly susceptible to noise. There are different regularization techniques to deal with ill-posed problems. Here, it is used the truncated singular value decomposition (tsvd) method. Roughly speaking, in the tsvd method the lowest condition values are set to zero, so the ratio of



Each panel indicates the transformation of all the vectors of magnitude 1 using matrices with different condition number (Adapted from `http://www.cse.illinois.edu/iem/linear_equations/condition_number/`.

Figure D.1: Linear transformation.

the largest to the smallest remaining condition values is not so big. Hence, the system is more robust to noise

tsvd method requires to decompose the matrix $\mathbf{A}_{\mu\times\nu}$ (where $\mu \geq \nu$) into 3 matrices: $\mathbf{U}$, $\mathbf{\Sigma}$ and $\mathbf{V}$

$$\mathbf{A} = \mathbf{U\Sigma V}^T = \sum_{i=1}^{n} \overrightarrow{u_i}\sigma_i \overrightarrow{v_i}^T \tag{D.2}$$

where $\mathbf{U} = \overrightarrow{u}_1, \ldots, \overrightarrow{u}_\nu$ and $\mathbf{V} = \overrightarrow{v}_1, \ldots, \overrightarrow{v}_\nu$ are matrices with orthonormal columns and $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_\nu)$ has the singular values in the diagonal. The singular values are ordered such that

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_\nu \geq 0 \tag{D.3}$$