

2019

# Controllability analysis and design for underactuated stochastic neurocontrol

---

<https://hdl.handle.net/2144/34932>

*"Downloaded from OpenBU. Boston University's institutional repository."*

BOSTON UNIVERSITY  
COLLEGE OF ENGINEERING

Dissertation

**CONTROLLABILITY ANALYSIS AND DESIGN FOR  
UNDERACTUATED STOCHASTIC NEUROCONTROL**

by

**SHUO HUANG**

B.S., Beijing University of Post and Telecommunications, 2012  
M.S., Northeastern University, 2014

Submitted in partial fulfillment of the  
requirements for the degree of  
Doctor of Philosophy

2019

© 2019 by  
SHUO HUANG  
All rights reserved

Approved by

First Reader

---

Jason Ritt, Ph.D.  
Adjunct Research Assistant Professor of Biomedical Engineering

Second Reader

---

John A. White, Ph.D.  
Professor and Chair of Biomedical Engineering

Third Reader

---

Uri Eden, Ph.D.  
Professor of Mathematics and Statistics

Fourth Reader

---

Kamal Sen, Ph.D.  
Associate Professor of Biomedical Engineering

Fifth Reader

---

ShiNung Ching, Ph.D.  
Das Family Career Development Distinguished Assistant Professor of  
Electrical and Systems Engineering  
Washington University in St. Louis

## **DEDICATION**

I would like to dedicate this work to my loving wife, Yan Ke, for her kindly encouragement and support, and to my newborn daughter, Mu Ke Huang, welcome to this world.

## **ACKNOWLEDGMENTS**

### **Committee Members**

Jason Ritt, PhD

John White, PhD

Kamal Sen, PhD

Uri Eden, PhD

ShiNung Ching, PhD

**Ritt lab members:** Samuel Brown, Su Jin Kim, Michael Palmiere

**CONTROLLABILITY ANALYSIS AND DESIGN FOR  
UNDERACTUATED STOCHASTIC NEUROCONTROL**

**SHUO HUANG**

Boston University College of Engineering, 2019

Major Professor: Jason Ritt, Ph.D., Adjunct Research Assistant Professor of Biomedical Engineering

**ABSTRACT**

Neuroengineering has advanced tremendously over the past decade, but for sensory prosthetics and similar applications, it remains an extraordinary challenge to access neurons at the single cell resolution of most sensory encoding theories. In particular, if each neuron is “tuned” to particular stimulus features, then eliciting a target percept requires activating only neurons tuned to that percept and not others. However, most available technology is underactuated, with orders of magnitude fewer independent control inputs than neural degrees of freedom, possibly limiting its effectiveness given the inherent trade-off of resolution with network size. Here I analyze controllability for pairs of neurons receiving a common input. In particular, I extend previous work on the deterministic control problem to include stochastic membrane dynamics, treating both cases as a bifurcation problem in the noise parameter. I determine controllable regions in parameter space using a combination of mathematical analysis and numerical solution of stochastic differential and Fokker-Planck equations. I explain how boundaries between these regions change with noise level, and connect to the dynamical mechanisms by which controllability is lost. I show that in stochastic systems, in contrast to deterministic systems, expanding the allowable input space to include exponential ramps expands the

parameter range over which neuron pairs are controllable. I also describe an alternative controllability definition using only mean spike times, as compared to the probability distribution of spiking within prespecified time intervals. These results could guide future control strategies in the development of sensory neuroprosthetics and other neurocontrol application.

## TABLE OF CONTENTS

DEDICATION .....	iv
ACKNOWLEDGMENTS .....	v
ABSTRACT .....	vi
TABLE OF CONTENTS .....	viii
LIST OF FIGURES .....	xi
CHAPTER ONE-Background of neuronal underactuated control .....	1
1.1 Importance of underactuation and spike timing in neurocontrol design.....	1
1.2 The Leaky Integrate-and-Fire neuronal model.....	6
1.3 Spike sequence control results in the deterministic case .....	7
1.4 Overview of the dissertation .....	11
CHAPTER TWO-Delineate underactuated pairwise stochastic controllability using single-pulse inputs.....	13
2.1 Definition of pairwise stochastic controllability .....	13
2.2 Brute force calculation of pair-wise controllable parameters .....	17
2.3 Interpretation of deterministic and stochastic controllability boundaries.....	27
2.4 Analytic approximation of controllability boundaries.....	32
2.5 Noise adjusted symmetry of boundaries about the nominal neuron.....	40
2.6 Comparison of analytically approximated and numerical derived boundaries .....	42
2.7 Pathwise simulation of membrane potentials.....	45

CHAPTER THREE-Positive ramp inputs restore controllability for a subset of single pulse uncontrollable pairs .....	52
3.1 Motivation for the expansion of input class .....	52
3.2 Derivation of ramp inputs and numerical assessment .....	53
CHAPTER FOUR- Mean spike time is a coarse definition of stochastic controllability .....	60
4.1 Alternative definitions of controllability.....	60
4.2 Defining and solving mean spike time control.....	61
CHAPTER FIVE- Discussion.....	68
5.1 Major contributions.....	68
5.2 Limitations and extensions.....	71
APPENDIX: Matlab code used in the dissertation .....	74
BIBLIOGRAPHY .....	122
CURRICULUM VITAE.....	129

## LIST OF TABLES

Table 2.1: Averaged statistics of different spike outcome groups.....	51
---	----

## LIST OF FIGURES

Figure 2.1. Pairwise stochastic underactuated controllability map for noise level $\hat{\sigma} =$ 0.2.....	20
Figure 2.2: Stochastic pairwise controllability map with extended SD search pool.....	23
Figure 2.3. Stochastic pairwise controllability maps at different $P_{th}$ .....	25
Figure 2.4: Pairwise stochastic controllability boundaries.....	29
Figure 2.5. Survival probability approximated by one minus the upper tail of the Gaussian density of membrane potentials.....	33
Figure 2.6: Numerical and approximated analytical boundary curves at several noise levels.....	43
Figure 2.7: Sample traces for 4 neuron pairs.....	48
Figure 2.8: Percentages of successful spike outputs of fully controllable neuron pairs in $\hat{\sigma} = 0.2$ map.....	49
Figure 3.1: geometric interpretation of deterministic pairwise control using single pulse inputs.....	54
Figure 3.2. A ramp input and the deterministic membrane potentials.....	57
Figure 3.3: parameter extents that ramp input helps single-pulse uncontrollable neuron pairs.....	59
Figure 4.1: Mean spike time function varying with $\alpha$ when no input .....	63
Figure 4.2: Mean spike times of two IAF neurons when no input .....	64
Figure 4.3: Examples of sequence controllability using mean spike time alone .....	66

## **CHAPTER ONE-Background of neuronal underactuated control**

### **1.1 Importance of underactuation and spike timing in neurocontrol design**

The neuronal action potential is generally seen as the primary unit of information processing [1]. Neurons interact with each other in complicated ways, and generate spike patterns with often complicated connections to behavior. Artificially modifying neural activity, or neurostimulation, plays an important role in elucidating mechanisms in normal and pathological neural circuits, it especially has enormous applicability in Brain-Machine Interface (BMI) and smart prosthetics [2, 3, 4, 5]. In [3], through experiments with nonhuman primates, they have developed approaches to intuitively convey sensory information that is critical for object manipulation, information about contact locations, pressure, and timing through intra-cortical microstimulation of primary somatosensory cortex. Similar approaches also show promising results of human test [2].

In this dissertation, I focus on invasive stimulation, including electrical and optical methods. The majority of neurostimulation applications employ electrical stimulation [3, 2], including currently available clinical devices. Although optogenetic stimulation and recording [6, 7, 8, 9, 10, 11] has been growing quickly over the last decade, the largest obstacle to implement clinical application is that gene therapy are required to introduce the optogenetic methods, which are currently unproven for human use [12]. However, the ability to restrict stimulation to genetically targeted cell populations has propelled optogenetic stimulation to a dominant position in animal models [6], and improvements

in control design for such studies could be important in both preclinical and basic science applications.

In most electrical approaches, a distant electrode supplies the current return path and ground potential. Intracellular stimulation [13] uses an electrode penetrating the cell membrane and contacting the cytoplasm directly. Although intracellular stimulation provides the ability to change the membrane potential arbitrarily, and record the subthreshold potential, it is not suitable for most clinical applications, or in animal studies with physical restraints, since sensitive mechanical contact between the electrode and cell is required during the whole process, and this contact can be easily disrupted by very small tissue movement. Therefore, almost all clinical applications use extracellular electrodes [14], where the electrodes are placed near neurons of interest. General discussion of extracellular stimulation and recording can be found in [15].

One major issue that extracellular stimulation techniques face is a high neuron-to-actuator ratio. By actuator we mean an independent channel of stimulation. For example, MEMS fabricated electrode arrays typically span 1-2 mm, and while there could be hundreds to thousands of independent channels of stimulation [6], within the volume they cover, there may be millions of neurons, yielding a neuron-to-actuator ratio of at least 1000 to 1. Other probe designs may produce smaller ratios, but then the total circuit size is limited by physical (e.g. wiring) constraints to small numbers of neurons. The neuron-to-actuator ratio and neural ensemble size together capture the important design concept

of *stimulator density*, not in terms of tissue volume, but rather “fraction” of a neural circuit. During stimulation, the electrode imposes a non-uniform electric field that influences all neurons in the neighborhood of its tip, making selective stimulation of subset of neurons difficult to impossible, although current steering techniques with multiple contacts provide some improvement [16], but do not fundamentally alter the electrode-neuron coupling mechanism. In this dissertation, I focus on cases where the desired neural control resolution is higher than the actuator resolution, which as described above is likely a common concern in many clinical and animal settings.

A major issue extracellular recordings has is that they generally are unable to resolve subthreshold changes in membrane potential, and are limited to the detection of the timing of action potentials [15] [17]. Also, extracellular electrode can easily result in a persistent ambiguity as to which neuron is being recorded when multiple neurons are nearby. Although it is common to introduce closed-loop/feedback design for noisy system [18], given such extracellular constraints, open loop designs [19] are more feasible for this application.

Most current neuron control techniques are open loop, and build upon a binary stimulation state (On or Off). Examples include deep brain stimulation (DBS) as a treatment for Parkinson’s disease [20, 21, 22], and cochlear implants to restore hearing [23, 24, 25]. While use of these systems will include calibration stages that change waveform parameters, typically with a clinician in the loop, once “good” parameters are

selected, the control design itself is highly limited. However, the complexity of neuronal networks implies that timing of spikes plays an essential role in function and behavior [26, 27, 28]. So a central question is, can we use neurostimulation techniques to control the dynamics of spike in more complicated patterns that are similar to naturally observed neuronal activity? This problem can be framed in the context of systems and control theory. Given a model for neuronal activity, we can study the extent to which spike dynamics can be manipulated overall (“controllability”), and which inputs achieve a desired spike pattern (“design”). Taking the high neuron-to-actuator ratio issue into consideration, I formulate the control problem by limiting the stimulation to underactuated inputs.

A parallel and distinct research area has studied how external input changes the ensemble dynamics of large groups of neurons [29, 30, 31, 32]. In [32], control and network theories are used to offer a mechanistic explanation for how the brain moves between cognitive states drawn from the network organization of white matter microstructure. There are also control designs proposed to drive neuron oscillators to certain synchronized spiking patterns [33, 34, 35, 36, 37]. A pathological synchronization of neurons in the basal ganglia cortical loop is thought to be a factor contributing to Parkinson’s Disease [38, 39], and a number of desynchronizing control of neuronal oscillator network studies have been proposed [40, 41, 42, 43, 44, 45]. An important distinction between these results and the work in this dissertation is that here I focus on specific multi-neuron spike patterns, at a higher level of specificity than the “bulk”

synchronization or desynchronization typical of studies of oscillatory ensembles.

Classical system and control theory has been previously applied to the control of spiking in single neurons [46, 47, 48, 49, 50, 51]. In [50], the single neuron spike timing control problem is framed using a statistical model, and an optimal stimulus is calculated through maximizing the likelihood function dependent on the target spike train, under implementation constrains (e.g. stimulation power). In [51], a stochastic optimal control problem is solved for a single leaky integrate and fire neuron with a focus of precise spike time control. The work in this dissertation concentrates on the challenge of spike time control with underactuated systems, which these papers did not address. Notably, [46] introduced controllability definitions and control strategies for neuronal populations with underactuated stimulation constraints, though their analysis was limited to deterministic systems. A high level review of relevant neurocontrol problems is presented in [17].

## 1.2 The Leaky Integrate-and-Fire neuronal model

Hodgkin and Huxley model [52] opens up the field of mathematical modeling of neuron. However, its explanatory power comes at the cost of analytical and numerical tractability, especially when formulating larger networks of neurons. After the Hodgkin and Huxley model, several models have been formulated that preserve many of its dynamical characteristics but with lower dimension and overall equation complexity [53, 54].

Most published results (see summary above), and also the approach here, rely heavily on a highly simplified but well known and useful neural model, the Leaky Integrate-and-Fire neuron (LIF) [1]. For two noisy, uncoupled neurons receiving a common (underactuated) input, the LIF model is

$$\begin{pmatrix} \frac{dV_1}{dt} \\ \frac{dV_2}{dt} \end{pmatrix} = \begin{bmatrix} -\alpha_1 & 0 \\ 0 & -\alpha_2 \end{bmatrix} \begin{pmatrix} V_1 \\ V_2 \end{pmatrix} + \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} u(t) + \sigma \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \end{pmatrix} \quad (1.1)$$

Here,  $V_i$ ,  $i = 1,2$  is the membrane potential;  $\alpha_i$ ,  $\beta_i$ , relate to the (normalized) membrane resistance and membrane capacitance respectively; and  $\varepsilon_i$  are standard Gaussian white noises scaled by an intensity  $\sigma$ . For simplicity, we assume the two neurons have the same noise intensity. Whenever either voltage hits a predetermined threshold,  $V_i = V_{th}$ , a ‘spike’ is recorded, and the membrane potential is reset to  $V_{rest}$ . We choose normalized units so that  $V_{th} = 1$  and  $V_{rest} = 0$  throughout the dissertation. This choice of input form corresponds to current, as opposed to conductance, input, though for conductances with a suitably high reversal potential (e.g. Channelrhodopsin2 stimulation) relative to threshold, the differences in controls should be small, as the “driving force”  $E_{rev} - V$  will

remain roughly constant.

Equation 1.1 describes subthreshold membrane potential dynamics, with the external reset mechanism providing a simplified effect of action potential non-linearity. The control consists of the selection of the input function  $u(t)$  to produce a desired set of spike sequences or times, defined next.

### 1.3 Spike sequence control results in the deterministic case

I now summarize several definitions related to spike control used in previous work, as reviewed in [55]. These definitions were formed in the zero-noise case ( $\sigma = 0$ ), and will be modified in the next chapter for application to stochastic neurons. However, the fundamental ideas can be fixed clearly in the simpler deterministic setting.

*Definition 1.1 (Spike Sequence):* In a population of  $N$  neurons, an  $M$ -spike *sequence* is a vector

$$S = [s_1, s_2, \dots, s_M] \quad (1.2)$$

where  $s_i \in \{1, 2, \dots, N\}$  indicates the neuron which produces the  $i^{th}$  spike in the sequence.

*Definition 1.2 (Spike Pattern):* In a population of  $N$  neurons, an  $M$ -spike *pattern* is a sequence with associated timing, i.e.,

$$SP = [(s_1, t_1), (s_2, t_2), \dots, (s_M, t_M)] \quad (1.3)$$

where  $s_i \in \{1, 2, \dots, N\}$  indicates the neuron which produces the  $i^{th}$  spike at time  $t_i > 0$ ,

where  $t_1 < t_2 < \dots < t_M$ .

As discussed in [46], when considering spike sequences and patterns, the traditional pathwise definition of controllability, that all multineuronal voltage trajectories are achievable through some input choice, is likely to be unnecessarily strong. Instead, I allow for weaker notions of controllability dependent only on (extracellularly observable) spike trains:

*Definition 1.3:* We say that a population of neuron is *sequence controllable* if all spike sequences can be achieved via an appropriate choice of  $u(t)$ . A population is *pattern controllable* if all spike patterns can be achieved via some input choice.

The definitions of spike sequence or pattern controllability differ from full controllability in that the complete state trajectory (of  $V_i(t)$ ) need not be specified. In neuronal populations, when spikes are the key entities, full state specification may be too strict a condition. Moreover, in the applications I consider, the subthreshold potential is unobserved, further lowering its interest as a control target.

Following [46], I consider pairwise control of spike sequences first. Assume a pair of neurons has heterogeneous parameters,  $\alpha_1 \neq \alpha_2, \beta_1 \neq \beta_2$ . If we insert a long silent duration between each spike, where the input is off, to allow both membrane potentials to decay to  $V_{rest}$ , then we convert the spike sequence control problem to spike order control,

that is, choosing two different inputs  $u(t)$  that respectively make either neuron spike first, starting from rest. Note that it is still possible to achieve pattern controllability with this strategy, if all desired interspike intervals are at least as long as the minimally necessary silent period. Thus underactuated pattern control may be achievable with this simple strategy, at the expense of some constraints on feasible timing. However, this approach is not optimal (see [56] for extensions to time optimal control). The definition is in the following:

*Definition 1.4 (Pairwise Spike Order Control):* Consider a pair of neurons, labeled {Neuron 1, Neuron 2}. If there exists a non-negative input  $u(t) \geq 0$ ,  $t \in [0, T]$  such that  $V_1, V_2$  satisfy the following conditions,

$$\begin{cases} \exists 0 \leq \tau \leq T, & \text{s. t.} & V_1(\tau) = V_{th} \\ \forall 0 \leq \tau \leq T, & & V_2(\tau) < V_{th} \end{cases} \quad (1.4)$$

then we say  $u(t)$  makes Neuron 1 spike first. Similarly, we define a  $u(t)$  that makes Neuron 2 spike first when it exists. If inputs exist for both orderings, we say the pair is *spike order controllable*.

In the following, I summarize pairwise spike order controllability in the deterministic situation, summarizing [46, 55, 56]. By reducing the control problem to single spike intervals, I assume the initial condition

$$\begin{cases} V_1(t = 0) = V_{rest} \\ V_2(t = 0) = V_{rest} \end{cases}, \quad (1.5)$$

throughout the dissertation when discussing deterministic systems (in the stochastic case,

the initial condition is a 2D Gaussian distribution centered at rest, with variance  $\frac{\sigma^2}{2\alpha}$ . This choice is the steady state distribution for zero input, which is the natural analogue of “rest”). I everywhere constrain the input to be non-negative.

A necessary condition for pairwise sequence controllability is shown to be

$$\frac{\alpha_1 - \alpha_2}{\beta_1 - \beta_2} > 0 \quad (1.6)$$

When this condition is violated, there is no choice of (positive) input such that the leakier (higher  $\alpha$ ) neuron’s membrane potential will ‘cross’ the other neuron’s potential from below, thus blocking that neuron from ever reaching threshold without also producing a spike in the other neuron. A sufficient condition for pairwise sequence control (without loss of generality, we can choose indices so that  $\alpha_1 < \alpha_2$ ) is shown to be

$$\frac{\alpha_1}{\beta_1} < \frac{\alpha_2}{\beta_2}, \quad \alpha_1 < \alpha_2 \quad (1.7)$$

The proofs are found in [46, 55]. For a neuron pair satisfying both Equation 1.6 and 1.7, we can always choose two single square pulse inputs (positive constant  $u(t)$  over a specified duration) to achieve the two spike orders.

### 1.4 Overview of the dissertation

In this dissertation, I expand the above analysis of underactuated controllability of spike timing (using neurostimulation methods such as current injection or optogenetic activation, as common in real-world, systems level applications) to the case of neurons with stochastic dynamics. The primary contribution of this work is to elucidate pairwise underactuated neuron controllability in stochastic settings, by framing the problem as a bifurcation from the deterministic setting, with noise as the bifurcation parameter.

Chapter 2 discusses stochastic controllability using a simple input strategy: single pulse inputs. I introduce a definition of stochastic controllability using probabilistic criteria, and delineate a pairwise controllability map (over parameter space) using a brute force numerical approach based on Fokker-Planck equations. I discuss the differences between stochastic and deterministic controllability results. I then analytically derive approximate stochastic controllability boundary equations, based on simplifications of the threshold crossing process, and compare these analytic boundaries to the results found through brute force numerical search. In particular, the analytic approach clarifies how boundaries shift with changing noise levels, and provides an organizing framework of the different ways controllability can be lost. As a second numerical approach, I justify our density analysis through large scale solution of stochastic differential equations to find individual realizations of the membrane potentials. These results allow me to consider timing more directly, as compared to the probability of spiking within the stimulus duration as in the earlier numerical results.

Chapter 3 expands the input space to include ramping inputs, that can restore controllability for a subset of single-pulse-uncontrollable neuron pairs. As mean exit time distributions are another prominent form of analyzing stochastic systems, in Chapter 4, I redefine stochastic controllability based solely on mean exit times, and provide an analysis across parameters. I conclude that mean exit time is a coarse measurement of spike dynamics, compared to the above analysis of probability of spiking. That is, I find it is insufficient to know only which neuron “spikes first” on average in order to achieve pairwise ordered control.

## **CHAPTER TWO-Delineate underactuated pairwise stochastic controllability using single-pulse inputs**

### **2.1 Definition of pairwise stochastic controllability**

In this section, I define pairwise stochastic controllability for uncoupled neurons, and highlight differences with deterministic definitions. In the stochastic case, a probability threshold is applied for at least one spike, but the target neuron is allowed to spike multiple times. I use the Fokker-Planck (FP) equation to find the “survival probability” of a noisy LIF neuron (that is, the probability that the neuron remains entirely subthreshold), and use this survival probability to define control criteria.

Throughout this chapter, I limit the space of inputs  $u$  to single-pulse inputs, with a positive strength  $S$  and duration  $D$  (I will relax this assumption in Chapter 3). Combined with the assumption that both neurons start from their resting distribution, that is, treating spike sequence control as the spike order control problem, it is therefore sufficient to consider the probability of spiking during positive constant inputs. It is necessary that the input duration be chosen long enough to produce target spiking, but also short enough to limit non-target spiking. In connection with Definition 1.4, the deterministic pairwise spike order control problem, we first define its stochastic analogue by the following,

*Definition 2.1(Pairwise Stochastic Controllability):* Consider an uncoupled pair of neurons, labeled {Neuron 1, Neuron 2}, receiving a common input  $u(t)$ . Given a threshold,  $0 < P_{th} < 1$ , and (deterministic or stochastic) initial conditions for both

neurons,  $v_0^1, v_0^2 < V_{th}$ , if there are two nonnegative inputs,  $u_1(t), t \in [0, D_1]; u_2(\tau), \tau \in [0, D_2]$ , such that,

$$Prob(\text{Neuron 1 spikes}|u_1) (1 - Prob(\text{Neuron 2 spikes}|u_1)) \geq P_{th} \quad (2.1)$$

$$Prob(\text{Neuron 2 spikes}|u_2) (1 - Prob(\text{Neuron 1 spikes}|u_2)) \geq P_{th} \quad (2.2)$$

where  $Prob\{\dots\}$  means the probability of at least one spike occurring before the end of the input duration ( $D_1$  or  $D_2$ ), then we say Neuron 1, Neuron 2 are *pairwise stochastically controllable*.

Similar to Definition 1.4, the actual spike time in Definition 2.1 is unimportant, however, there are a couple of distinctions. First, a probabilistic threshold  $P_{th}$  is used, instead of a strict criterion on exactly one neuron hitting  $V_{th}$ . It is possible, for example, to compensate for an increase in non-target spiking by an even larger increase in target spiking while maintaining the same value of the criterion function. Conversely, one input may perform better than another by either reducing non-target or increasing target spiking. Note, however, that since probabilities are limited within zero and one, the inequality on the product also enforces that each individual term must meet the probabilistic threshold. That is, both the probability of target spiking and the probability against non-target spiking individually must be greater than  $P_{th}$ . Second, in Definition 1.4, the target neuron is meant to spike once, while in Definition 2.1,  $Prob\{\dots\}$  is the probability that the neuron spikes at least once before the end of input. This definition facilitates the use of the survival densities found through direct solution, rather than

having to find pathwise probabilities of threshold crossing. However, I expect for most of the inputs under consideration (short pulses, constrained to prevent excessive non-target spiking) that the probability of multiple target spikes remains low, and in practice the difference in outcomes should be small.

For a pair of noisy leaky integrate-and-fire (LIF) neurons stated in Equation 1.1, one can find the individual probability density functions (PDFs)  $f(v_i, t)$  of their membrane potentials  $V_i$  by solving the Fokker-Planck equation [57]

$$\frac{\partial f}{\partial t} = \frac{\sigma^2}{2} \frac{\partial^2 f}{\partial v_i^2} - \frac{\partial}{\partial v_i} [(-\alpha_i v_i + \beta_i u) f] \quad (2.3)$$

with an absorbing boundary condition that takes neurons out of the pool at their first spike:

$$f(v_i, t) = 0 \text{ when } v_i = V_{th} \quad (2.4)$$

Given this boundary condition,  $f$  will tend to zero over time, and represents the density of those realizations that have never spiked. As such, one can define the *survival probability*  $G$  of Neuron  $i$  at time  $t$  (given the initial condition  $v_0$  and input  $u$ ), as

$$G(t) = \int_{-\infty}^{V_{th}} f(v_i, t) dv_i \quad (2.5)$$

If at  $t = 0$ , a spike just occurred and membrane potential is reset to  $v_{rest}$ ,  $G(t)$  is the probability that there is no spike during  $(0, t]$ , and  $1 - G(t)$  is the probability that there has been at least one spike during  $(0, t]$ . Incorporating the survival probability  $G(t)$  into Definition 2.1, we can refine the definition of pairwise stochastic controllability:

*Definition 2.2 (Pairwise Stochastic Controllability using Survival Functions):* Consider an uncoupled pair of neurons, labeled {Neuron 1, Neuron 2}, receiving a common input  $u(t)$ . Given a threshold,  $0 < P_{th} < 1$ , and (deterministic or stochastic) initial conditions for both neurons,  $v_0^1, v_0^2 < V_{th}$ , if there are two nonnegative inputs,  $u_1(t), t \in [0, D_1]; u_2(\tau), \tau \in [0, D_2]$ , such that,

$$(1 - G_1(D_1)) G_2(D_1) \geq P_{th} \quad (2.6)$$

$$(1 - G_2(D_2)) G_1(D_2) \geq P_{th} \quad (2.7)$$

where  $G_1$  and  $G_2$  are respective survival probabilities for Neuron 1 and Neuron 2, then we say independent noisy LIF Neuron 1, Neuron 2 are underactuated pairwise stochastic controllable.

A major difference between deterministic and stochastic pairwise controllability is that an additional parameter is introduced in the probability threshold  $P_{th}$ . The deterministic controllability definition can be thought of as the special case  $P_{th} = 1$ , but I now accept “imperfect” control. Since the survival probability of a stochastic LIF neuron will never strictly equal 0 or 1 under any nonnegative input, stochastic controllability is *a priori* impossible if we apply  $P_{th} = 1$  under noise. Conversely, taking  $P_{th} \approx 0$  suggests that nearly all neuron pairs are stochastically controllable, which is also meaningless. One should therefore choose a “high” value such that the control problem remains feasible for a large set of neuron pairs. I set  $P_{th} = 0.9$  throughout most of the dissertation, but discuss how varying  $P_{th}$  around 0.9 would change the stochastic controllability in later sections.

## 2.2 Brute force calculation of pair-wise controllable parameters

In this section, I describe a simple numerical approach to find neuron parameters ( $\alpha$  and  $\beta$ , for fixed levels of  $\sigma$ ) of pairwise controllable (stochastic) neuron pairs using single-pulse inputs. Unsurprisingly, stochastic controllable regions are a subset of corresponding regions for the deterministic case. An issue with the numerical results is that only a finite number of single pulse inputs can be tested at a fixed point in parameter space, but the range of inputs necessary to achieve pairwise control can vary widely across those parameters. As such, these computationally determined controllable regions are expected to be a conservative (subset) measure of the true controllable regions. I assess the importance of this issue by extending the input space (to a larger range of strengths and durations), and show that pairwise controllable regions expand only slightly for a significant increase in simulation time. Thus with the caveat that the naïve algorithm using a modified grid search is computationally inefficient if precise boundaries are needed, this result suggests the approach is still useful, which I further demonstrate by looking at changes in controllable parameter extents at different  $P_{th}$  values near 0.9.

I first simplify the analysis by re-scaling the Equation 1.1 with the substitutions:

$$\hat{t} = \alpha_1 t$$

$$\hat{\alpha}_2 = \alpha_2 / \alpha_1$$

$$\hat{\beta}_2 = \beta_2 / \beta_1$$

$$\hat{u} = \frac{\beta_1}{\alpha_1} u$$

$$\hat{\sigma} = \sigma / \sqrt{\alpha_1}$$

to yield dynamical equations

$$\frac{dV_1}{d\hat{t}} = -V_1 + \hat{u} + \hat{\sigma}\varepsilon_1 \quad (2.8)$$

$$\frac{dV_2}{d\hat{t}} = -\hat{\alpha}_2 V_2 + \hat{\beta}_2 \hat{u} + \hat{\sigma}\varepsilon_2 \quad (2.9)$$

(recall  $\varepsilon_1, \varepsilon_2$  are independent standard Gaussian white noises). As in the previous section, we can consider  $\hat{u}$  to be nonnegative constant for solving the spike ordering problem. In essence, Neuron 1 is declared to be a “nominal neuron” with fixed unit parameters. The search is then limited to  $\hat{\alpha}_2, \hat{\beta}_2$  to find other neurons controllable with Neuron 1. Note that here no assumptions are made as to relative sizes of the parameters for the two neurons, and the selection of Neuron 1 is arbitrary.

Since it is one-dimensional in space, it is straightforward to apply the standard centered finite difference Crank-Nicholson scheme to step in time [58].

The search algorithm proceeds as follows: at a fixed noise level,  $\hat{\sigma}$ , I iterate over a fixed increment grid of  $\hat{\alpha}_2$  and  $\hat{\beta}_2$  within the respective ranges  $[0, \hat{\alpha}_2^{max}]$  and  $[0, \hat{\beta}_2^{max}]$ . I use two different grids in separate computations: one grid surrounds the nominal neuron and captures the full range of behaviors, and a second higher resolution grid covers the  $0 < \hat{\alpha}_2 < 1, 0 < \hat{\beta}_2 < 1$  region, for better visualization. At each parameter point, I

iterate over a Strength-Duration pool, generating single-pulse inputs with input strength  $\hat{u}$  from  $[S_{min}: dS: S_{max}]$  and input duration  $\hat{T}$  from  $[D_{min}: dD: D_{max}]$ .

I can skip some  $\hat{\alpha}_2$  and  $\hat{\beta}_2$  analysis, if Neuron 2 ( $\hat{\alpha}_2^i, \hat{\beta}_2^i$ ) always spike when Neuron 1 is the target (only Equation 2.7 is met), then I can skip analyzing  $\hat{\alpha}_2 \leq \hat{\alpha}_2^i, \hat{\beta}_2 \geq \hat{\beta}_2^i$ , because these Neuron 2 are even more likely to spike. Similarly, if Neuron 1 always spike if Neuron 2 ( $\hat{\alpha}_2^j, \hat{\beta}_2^j$ ) is the target (only Equation 2.6 is met), I can skip analyzing  $\hat{\alpha}_2 \geq \hat{\alpha}_2^j, \hat{\beta}_2 \leq \hat{\beta}_2^j$  since they are even less likely to spike.

In the deterministic case, it is not possible for the non-target neuron to spike once stimulation is turned off. However, in the stochastic case, we expect non-target neurons to be at elevated (subthreshold) potentials at stimulus termination, such that noise might still produce a non-target spike before the neuron has decayed fully to rest. In order to check for spikes from the non-target neuron after stimulation, I add a “silence period” (the input is off) to every input and continue the Fokker-Plank simulation throughout this period.

For each parameter point  $\hat{\alpha}_2 \hat{\beta}_2$ , I calculate the two probabilistic criteria  $G_1$  and  $G_2$  in Definition 2.2 (Equations 2.6 and 2.7) at every point in the Strength-Duration pool, at the end of the solution period (including silence). I place the parameter point into one of four categories:

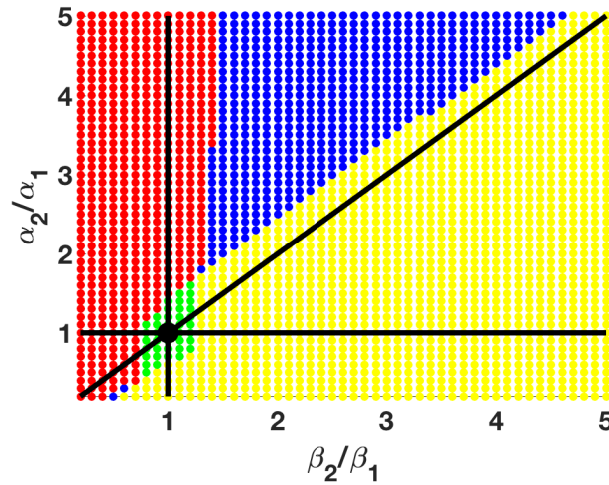
- (1) Only Equation 2.6 is met, so only Neuron 1 can spike alone.

(2) Only Equation 2.7 is met, so only Neuron 2 can spike alone.

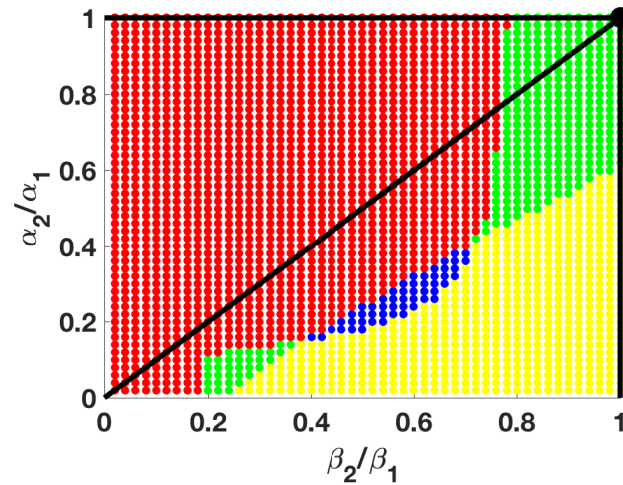
(3) Both Equation 2.6 and 2.7 are met, Neuron 1 and Neuron 2 are stochastically pairwise controllable.

(4) Neither Equation 2.6 nor 2.7 can be met, so neither neuron can spike alone.

A



B



**Figure 2.1. Pairwise stochastic underactuated controllability map for noise level  $\hat{\sigma} = 0.2$ .** (A). Large scale map for  $\hat{\alpha}_2 \in [0.2:0.1:5]$ ,  $\hat{\beta}_2 \in [0.2:0.1:5]$  (B). Higher resolution map in lower left corner, with  $\hat{\alpha}_2 \in [0.02:0.02:1]$ ,  $\hat{\beta}_2 \in [0.02:0.02:1]$ . The color code is the same for both panels: The nominal neuron is shown as a black dot at coordinates  $[1, 1]$ . *Blue dots* mark fully controllable pairs. *Yellow dots* mark pairs where only Neuron 2 can spike alone. *Red dots* mark pairs where only Neuron 1 can spike alone. *Green dots* mark totally uncontrollable pairs. *Black lines* mark controllability boundaries for the deterministic case.

Figure 2.1 shows the results for one choice of  $\sigma$ . The solid black lines surround the pairwise controllable regions in the deterministic case (see Equations 1.6 and 1.7), consisting of upward and downward wedges extending from the nominal neuron [46] [56]. The stochastic pairwise controllable regions (blue dots) cover only a subset of these deterministically controllable regions. Moreover, compared to the upper right ( $\hat{\alpha}_2 > 1, \hat{\beta}_2 > 1$ ) quadrant, the loss of controllability in the ( $\hat{\alpha}_2 \leq 1, \hat{\beta}_2 \leq 1$ ) quadrant appears more substantial.

To the left side (red dots) of the pairwise controllable region, only when Neuron 1 is the target can an input meet criterion (Equation 2.6). In this region, Neuron 2 is not sufficiently responsive to inputs, as  $\hat{\beta}_2$  is small, so it becomes impossible to induce Neuron 2 spiking without also decreasing the probability of Neuron 1 silence below threshold. To the right side (yellow dots) of the pairwise controllable region, the situation is reversed, with only Neuron 2 possible as a successful target. Both of these outcomes have analogues in the deterministic case. However, when Neuron 2 is near Neuron 1 (green dots), a novel situation arises, where neither neuron can spike alone. In the deterministic case, the only “totally uncontrollable” point (where it is impossible to make either neuron spike alone) is for two identical neurons, but with noise, this totally uncontrollable region expands to have positive diameter. I will discuss the stochastic controllability in more details in the following sections.

There is an important caveat to Figure 2.1, that it is not a fully accurate stochastic pairwise controllability map under noise  $\hat{\sigma} = 0.2$  and  $P_{th} = 0.9$ . The strength-duration pool consists of only a subset of single-pulse inputs, and is not adapted to each  $(\alpha, \beta)$  point, so may miss inputs that in fact meet the criteria in Definition 2.2. This bias is conservative, in that it will only make a mistake in the direction of falsely declaring a controllable pair uncontrollable. The pairwise controllable region shown in Figure 2.1 should therefore be only a subset of the real one, with improving accuracy of the numerical result as the strength-duration pool is enlarged.

Parameters of the Strength-Duration pool I used above are  $S_{min} = 0.1$ ,  $S_{max} = 12$ ,  $D_{min} = 0.1$ ,  $D_{max} = 15$ , and  $dS = dD = 0.1$ . The pool is defined for input parameters  $\hat{u}, \hat{T}$  normalized by Neuron 1 parameters, but we can convert back to non-normalized parameters to find  $u_{min} = 0.1 \frac{\alpha_1}{\beta_1}$ ,  $u_{max} = 12 \frac{\alpha_1}{\beta_1}$ ,  $T_{min} = 0.1/\alpha_1$ ,  $T_{max} = 15/\alpha_1$ ,  $du = 0.1 \frac{\alpha_1}{\beta_1}$  and  $dT = 0.1/\alpha_1$ . These ranges correspond to steady-state (average) Neuron 1 potentials from 1/10 to 12 times the threshold, and durations from 1/10 to 15 times the Neuron 1 time constant. I now expand the pool by doubling the range and using a 2.5 times more refined duration grid. The new pool parameters are thus  $S_{min} = 0.1$ ,  $S_{max} = 24$ ,  $D_{min} = 0.04$ ,  $D_{max} = 30$ ,  $dS = 0.1$ ,  $dD = 0.04$ .

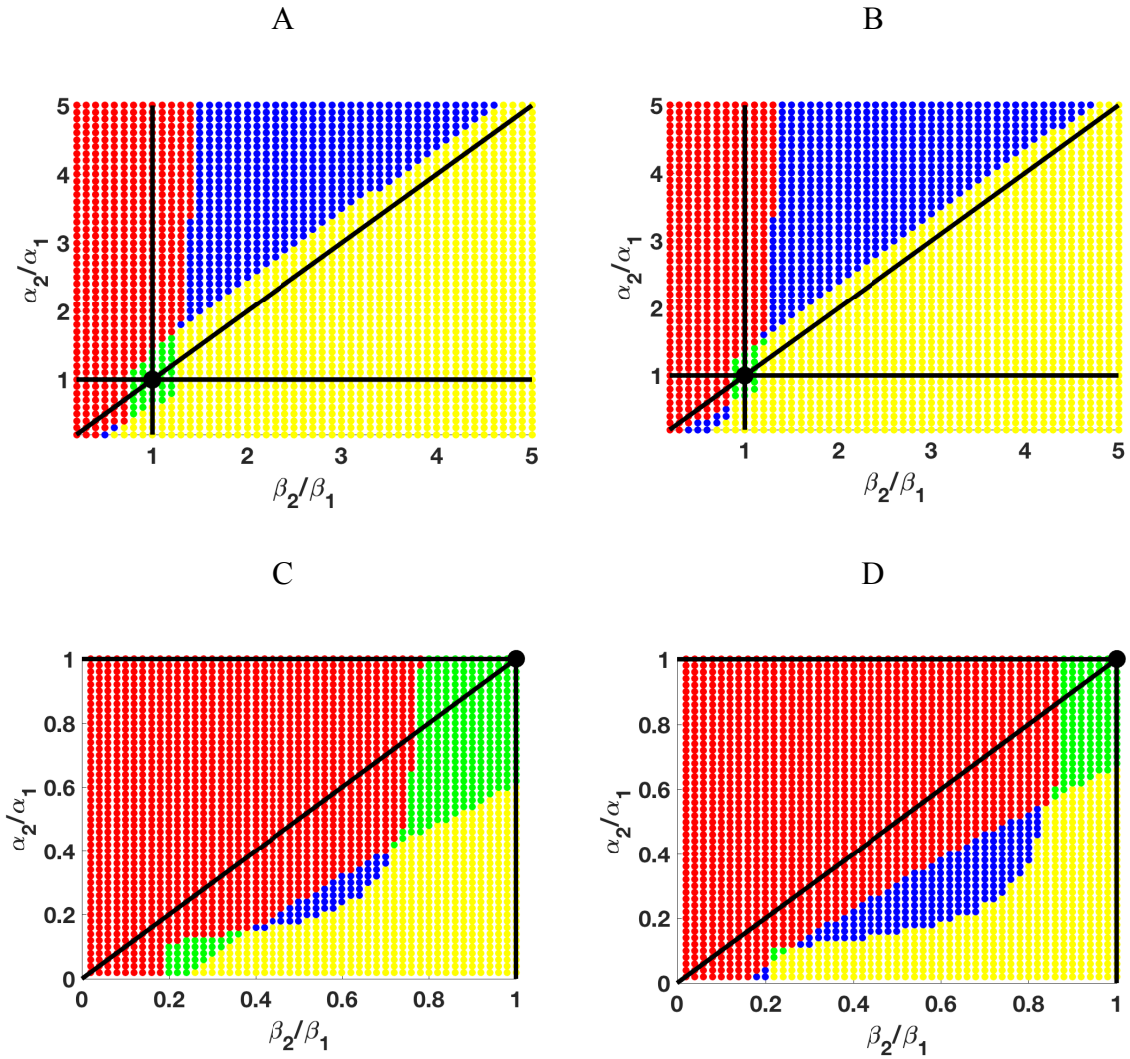


Figure 2.2: Stochastic pairwise controllability map with extended SD search pool. (A, C) are copied from Figure 2.1. (B) Updated controllability map with extended SD search pool,  $\hat{\alpha}_2, \hat{\beta}_2 \in [0.2:0.1:5]$ . (D) Updated controllability map with extended SD search pool,  $\hat{\alpha}_2, \hat{\beta}_2 \in [0.02:0.02:1]$ .  $\hat{\sigma} = 0.2$  in all panels.

The new controllability map (over the same  $(\alpha, \beta)$  range) is shown in Figure 2.2. For comparison, Figure 2.1 is copied in Figure 2.2. Consistent with the expected bias, the pairwise controllable region (blue dots) extends further outward, and the size of the totally uncontrollable region (green dots) around  $\hat{\alpha}_2 = 1, \hat{\beta}_2 = 1$  decreases. The computation time increased dramatically with the expanded pool. It took more than 100 hours to compute Figure 2.2 B and D, which is nearly 8 times the original computation time. However, the regions calculated by the two pools show similar shapes, with limited change in extent.

As such, for analyses where it is sufficient to find the general shape of regions, and some conservative bias is tolerable, using the smaller pool is an acceptable trade-off for computation speed. I will now use this pool to examine how the choice of stochastic controllability criterion  $P_{th}$  affects pairwise controllability.

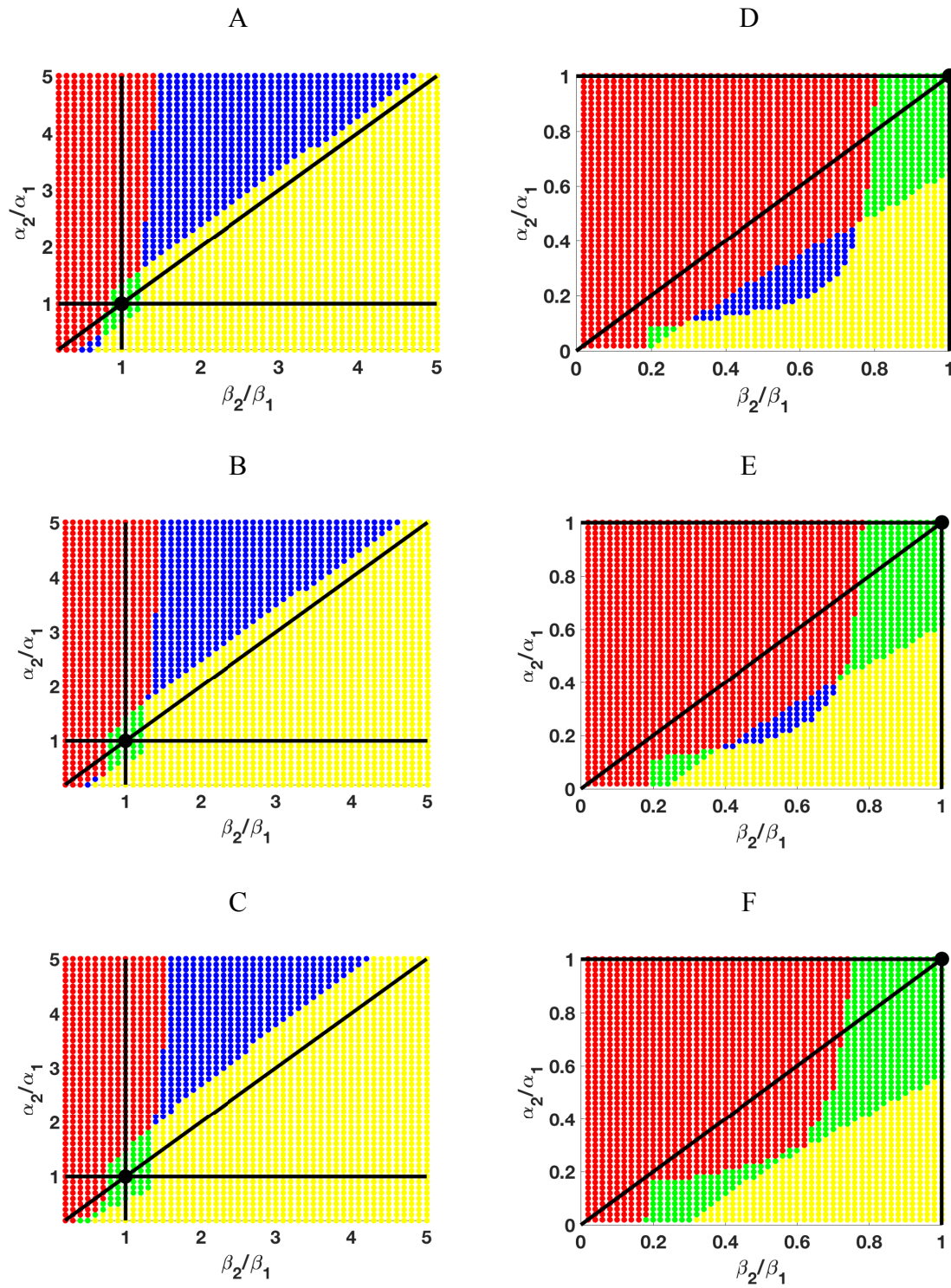


Figure 2.3. Stochastic pairwise controllability maps at (A,D)  $P_{th} = 0.85$ , (B,E)  $P_{th} = 0.9$ , copied from Figure 2.1. (C,F)  $P_{th} = 0.95$ .

Figure 2.3 shows controllability maps at  $P_{th} = 0.85, 0.9, \text{ and } 0.95$ , computed using identical methods to the above. As the criterion is made more strict, from  $P_{th} = 0.85$  (A and D) to  $P_{th} = 0.95$ , (C and F), the pairwise controllable region shrinks, and the totally uncontrollable region grows. The extent of change appears larger in the lower left quadrant  $\hat{\alpha}_2 < 1, \hat{\beta}_2 < 1$ , than in the upper right  $\hat{\alpha}_2 > 1, \hat{\beta}_2 > 1$ . A possible reason for this result is that the noisy LIF neuron membrane potential has larger standard deviation (for a given noise level) if it has smaller leak constant  $\alpha$ . So under the same noise intensity, Neuron 2 with smaller  $\hat{\alpha}_2$  is more easily perturbed by noise and harder to control. However, the limitation of the strength-duration pool must be kept in mind, so while the qualitative picture should hold, the exact values at which controllability is lost are likely not accurate.

### 2.3 Interpretation of deterministic and stochastic controllability boundaries

In this section, I more precisely define and examine boundaries of controllability. The boundaries “move inward” in the stochastic case, representing the loss of controllability of neuron pairs near the deterministic boundaries. I qualitatively explain the reasons for this loss of controllability.

I first define pairwise stochastic controllability boundaries based on the probabilistic criteria from Definition 2.2, Equations 2.6 and 2.7. The definition makes extensive use of monotonicity, in the sense that when a certain parameter point is known not to correspond to a controllable pair then, depending on the category as described in the previous section, all points with either lower or higher  $\hat{\beta}_2$  (respectively, to the “left” or the “right”) will have the same property.

*Definition 2.3 (Pairwise Stochastic Controllability Boundaries):* Assume the same hypotheses as Definition 2.2. For each  $\hat{\alpha}_2$ , if there is a  $\hat{\beta}_2^R$  such that for all  $\hat{\beta}_2 \in [0, \hat{\beta}_2^R]$  ( $\hat{\beta}_2^R$  is the largest nonnegative value among all qualified candidates), there is a nonnegative single-pulse input  $u_1$  for which the following holds,

$$(1 - G_1(T_1)) G_2(T_1) \geq P_{th}$$

then  $(\hat{\alpha}_2, \hat{\beta}_2^R)$  is called a point on the *Right-Hand Side (RHS) boundary* (that is, these points determine the limit of where it is possible to reach criterion with Neuron 1 as target). For each  $\hat{\alpha}_2$ , if there is a  $\hat{\beta}_2^L$  such that for all  $\hat{\beta}_2 \in [\hat{\beta}_2^L, \infty)$  ( $\hat{\beta}_2^L$  is the smallest

nonnegative value among all qualified candidates), there is a nonnegative single-pulse input  $u_2$  for which the following holds,

$$G_1(T_2) (1 - G_2(T_2)) \geq P_{th}$$

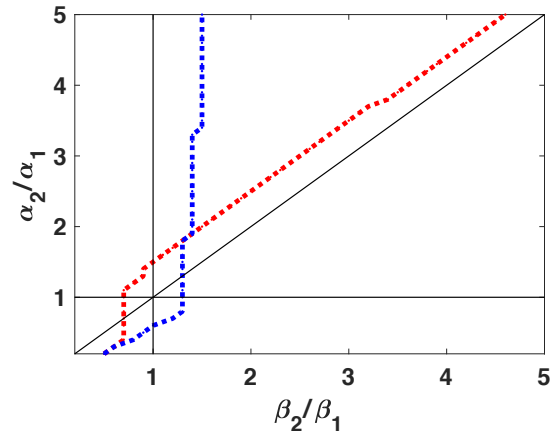
then  $(\hat{\alpha}_2, \hat{\beta}_2^L)$  is called a point on the *Left-Hand Side (LHS) boundary*. (that is, these points determine the limit of where it is possible to reach criterion with Neuron 2 as target).

Note Definition 2.3 uses normalized parameters. To connect the definition with the numerical controllability map (Figure 2.1), fix a sufficiently large  $\hat{\alpha}_2 > 1$ . that some points at that parameter are blue. At  $\hat{\beta}_2 = 0$  only the nominal neuron, Neuron 1, can spike alone, meaning the criterion  $(1 - G_1) G_2 \geq P_{th}$  is met for some input (this is the red region). Once  $\hat{\beta}_2$  increases to  $\hat{\beta}_2^L$ , Neuron 2 becomes sufficiently responsive to input that it also can spike alone, as  $(1 - G_2) G_1 \geq P_{th}$  is met by a proper input selection. As  $\hat{\beta}_2$  continues to increase (now through the blue region), Neuron 2 spikes more easily, until  $\hat{\beta}_2 > \hat{\beta}_2^R$ , at which point Neuron 2 is so much more sensitive to input than Neuron 1 that only  $(1 - G_2) G_1 \geq P_{th}$  can be met, starting the yellow region.

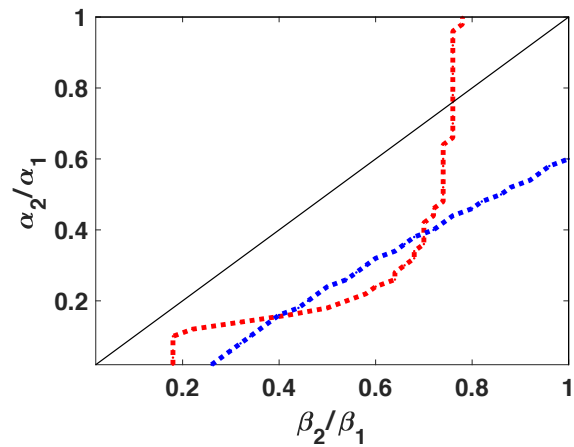
At an  $\hat{\alpha}_2$  for which  $\hat{\beta}_2^L > \hat{\beta}_2^R$ , any Neuron 2 with parameter  $\hat{\beta}_2 \in (\hat{\beta}_2^R, \hat{\beta}_2^L)$  will be totally uncontrollable with Neuron 1 (green dots). Basing the argument on the crossing of these two boundary curves makes explicit something apparent in the original maps: at each level of  $\hat{\alpha}_2$ , the existence of a controllable region is mutually exclusive with existence of a region of total uncontrollability. However, for lower  $\hat{\alpha}_2$ , the boundaries can again cross,

permitting a controllable region.

A



B



**Figure 2.4: Pairwise stochastic controllability boundaries. Dotted blue curve: LHS boundary. Dotted red curve: RHS boundary.**

In Figure 2.4, I extract the pairwise stochastic controllability boundaries from the numerical map (the colors of the boundary curves have no relation to the pixel colors in Figure 2.1), and include straight lines (black) that form the deterministic boundaries. The LHS controllability boundary in the deterministic case is defined by [46] and Chapter 1.

$$\begin{cases} \hat{\beta}_2 = \hat{\alpha}_2, \hat{\alpha}_2 \leq 1 \\ \hat{\beta}_2 = 1, \hat{\alpha}_2 > 1 \end{cases} \quad (2.10)$$

The corresponding RHS boundary is defined by

$$\begin{cases} \hat{\beta}_2 = 1, \hat{\alpha}_2 \leq 1 \\ \hat{\beta}_2 = \hat{\alpha}_2, \hat{\alpha}_2 > 1 \end{cases} \quad (2.11)$$

Note that while the boundary of the controllable region could be described more simply as the union of the diagonal  $\alpha = \beta$  and a vertical line at  $\beta = 1$ , the decomposition into two “kinked” LHS and RHS pieces preserves information about the manner by which controllability changes as the boundary is crossed. Note also that the deterministic boundaries are open (neuron pairs residing on the boundaries are not controllable).

As shown in Figure 2.4, the LHS and RHS boundaries move in opposite directions as noise is introduced. In particular, with zero noise the kinks in the two curves coincide at the nominal neuron,  $(\alpha, \beta) = (1, 1)$ . Once noise is introduced, the LHS curve moves down to the right, and the RHS curve moves up to the left, so that the nominal neuron becomes enveloped. This process is what leads to the formation of the totally uncontrollable region.

One reason the boundary shifts occur is that noise creates a “danger zone” near  $V_{th}$  for the non-target neuron, where its membrane potential can be “dragged” above  $V_{th}$  even if its mean potential remains subthreshold. The longer the duration of positive input stimulation the lower the non-target neuron’s survival probability becomes, and in the vicinity of the deterministic boundaries, the required inputs that achieve pairwise control

become “unrealistic” as either impulses or of infinitely long duration. Recall also that limitations on the input space produce some inaccuracy of the numerically computed boundaries, so the inward shifts at a given noise level may not be as far from the analytically computed deterministic boundaries as appears in Figure 2.4.

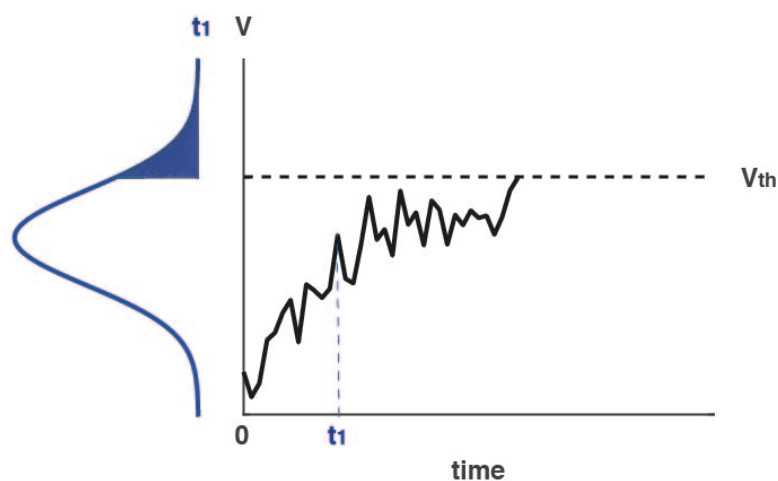
The kinks in the boundary curves indicate a change of input strategy. For example, in the deterministic case, if Neuron 2 is close to the LHS boundary and  $\hat{\alpha}_2 \leq 1$ , in order to make it spike before Neuron 1, the input should be a small strength, long duration pulse. The required input changes to an impulse for  $\hat{\alpha}_2 > 1$ . In the deterministic case, the change points on the LHS and RHS boundaries coincide at  $\hat{\alpha}_2 = 1, \hat{\beta}_2 = 1$ . However, they split apart under noise.

## 2.4 Analytic approximation of controllability boundaries

The previous sections relied heavily on computational assessment of controllability boundaries. While the methods were exact in principle, numerical limitations introduced errors that are difficult to quantify, and that could be simply resolved but at the expense of much greater computation time. In this section, I find an approximate but analytical expression for the survival probability using the Gaussian cumulative distribution function, and use this expression to derive the stochastic controllability boundaries. I compare the characteristics of the analytic boundaries to those from the numerical approach. Because these approximations are continuous at zero noise, I argue they should be close to the true boundaries at small noise levels.

In order to get analytic expressions for the stochastic boundaries in Definition 2.3, the key component is an analytic expression of the survival probability  $G$ . However, because of the absorbing boundary condition associated with the Fokker-Planck equation (Equations 2.3 and 2.4), the probability density  $f(v, t)$  is not Gaussian, and lacks a known, closed analytical form [57]. The absorbing boundary removes realizations that cross threshold. While the individual paths would be reset, I count only the first spike, and do not track whether subsequent spikes also occur, so that the reset dynamics can be ignored in the survival probability.

As an approximate alternative, one can instead determine what fraction of paths would be above threshold (in the absence of an absorbing boundary) at the end of the input pulse. All such paths correspond to a neuron that would spike at least once, but the fraction will underestimate total spiking, by not keeping track of paths that passed threshold before the end of the stimulus, and then fell below. For the IAF model and positive pulses considered here, the set of such paths should be small except at high noise or for long duration pulses.



**Figure 2.5. Survival probability approximated by one minus the upper tail of the Gaussian density of membrane potentials, which obey an Ornstein-Uhlenbeck process.**

Figure 2.5 illustrates the idea by showing a sample path (solid black curve) of a noisy LIF neuron. Its membrane potential at each time obeys a Gaussian density. At some time  $t_1$ , which one could imagine is the time of stimulus offset, the density is shown as a blue curve on the left. The survival probability is approximated by the probability mass below threshold, which equals one minus the shaded area of the upper tail. I set up this approximation explicitly in the following assumptions.

*Assumption 2.1:* The true solution  $f(v, t)$  to the IAF Fokker-Planck equation can be approximated by  $\tilde{f}(v, t)$ , the solution without the absorbing boundary condition, which is equivalent to low-pass filtered Brownian motion, also known as an Ornstein-Uhlenbeck process [57].

For input  $u$  a nonnegative constant, and Gaussian initial condition  $N(v_0, \sigma_{init})$ , standard methods show  $\tilde{f}(v, t)$  is a Gaussian density [57] with mean

$$\tilde{\mu} = \frac{\beta u}{\alpha} (1 - e^{-\alpha t}) \quad (2.12)$$

and variance

$$\tilde{\sigma}^2 = \frac{\sigma^2}{2\alpha} + (\sigma_{init}^2 - \frac{\sigma^2}{2\alpha}) e^{-2\alpha t} \quad (2.13)$$

If we assume inter-spike intervals are sufficiently long for both neurons to return to their resting state after stimulation (note  $u = 0$  during this time), then when the next input arrives, the initial condition should have the steady state parameters,  $v_0 = V_{rest} = 0$  and

$\sigma_{init} = \sqrt{\frac{\sigma^2}{2\alpha}}$ , so the variance during the input becomes

$$\tilde{\sigma}^2 = \sigma_{init}^2 = \frac{\sigma^2}{2\alpha} \quad (2.14)$$

In particular, the variance is a constant.

*Assumption 2.2:* Defining the cumulative distribution function for  $\tilde{f}(v, t)$  as  $\tilde{F}(v, t)$ , the true survival probability  $G(t)$  can be approximated by the cumulative distribution at threshold,

$$\tilde{G}(t) = \tilde{F}(V_{th}, t) \quad (2.15)$$

I use the approximated survival probability  $\tilde{G}(t)$  in place of the true  $G(t)$  in Definition 2.3 to find analytic expressions of stochastic controllability boundaries in the following. If we apply Assumptions 2.1 and 2.2 to Definition 2.3, the probabilistic criteria become

$$\left(1 - \tilde{F}_1(V_{th}, T_1)\right) \tilde{F}_2(V_{th}, T_1) \geq P_{th} \quad (2.16)$$

$$\left(1 - \tilde{F}_2(V_{th}, T_2)\right) \tilde{F}_1(V_{th}, T_2) \geq P_{th} \quad (2.17)$$

where  $\tilde{F}_1, \tilde{F}_2$  are cumulative distributions corresponding to  $\tilde{f}_1, \tilde{f}_2$  respectively. At the boundaries, I make a simplifying assumption that target Neuron  $i$  and non-target Neuron  $j$  contribute equally to the product, and look for when equality holds (since that is the boundary case), that is,

$$\left(1 - \tilde{F}_i(V_{th}, T_i)\right) = \sqrt{P_{th}} \quad (target) \quad (2.18)$$

$$\tilde{F}_j(V_{th}, T_i) = \sqrt{P_{th}} \quad (non - target) \quad (2.19)$$

Using that  $\tilde{F}_i, \tilde{F}_j$  are Gaussian distributions and, for my probability threshold choice of 0.9, that  $\sqrt{P_{th}} \approx 0.95$ , Equation 2.18 and 2.19 can be rewritten as

$$\tilde{\mu}_i - 1.64\tilde{\sigma}_i = V_{th} \quad (target) \quad (2.20)$$

$$\tilde{\mu}_j + 1.64\tilde{\sigma}_j = V_{th} \quad (non - target) \quad (2.21)$$

$\tilde{\mu}_i, \tilde{\mu}_j, \tilde{\sigma}_i, \tilde{\sigma}_j$  are the means and standard deviations defined after Assumption 2.1. The coefficient in front of the standard deviation comes from the 5% (one-sided) tail. For notational simplicity, I define

$$\Delta_i = 1.64\tilde{\sigma}_i \quad (2.22)$$

and use  $\Delta_i$  instead in the following.

For concreteness, I first consider the approximated analytic expression for the LHS boundary, on which Neuron 2 is the target, and Neuron 1 (the nominal neuron) is the non-target. To find the boundary in the  $\hat{\alpha}_2, \hat{\beta}_2$  plane, Equations 2.20 and 2.21 must hold. Recall Neuron 1 is the nominal neuron, with constant unitary parameters after rescaling. I therefore start by finding the strength and duration for an input  $\hat{u}$  such that Neuron 1 satisfies Equation 2.21; this solution depends only on the noise level, and not neural parameters  $\alpha$  and  $\beta$ . I then find those parameters of Neuron 2 that satisfy Equation 2.20 with the *same* input  $\hat{u}$ , to define the boundary curve.

In the deterministic case, the two inputs used for selective control can be chosen as nearly impulsive (large strength and small duration) and a “slow” pulse (small strength and long duration) [46]. Motivated by this case, and the similarity of pulse types found by the numerical search in the previous sections, I adopt a similar strategy to perform analysis in the stochastic case.

Consider first the limit of short pulses of duration  $\hat{T} \ll 1$ . Taking Taylor expansions in the path means, Equations 2.12, and rewriting the constant standard deviation, yields

$$\tilde{\mu}_1 \approx \hat{u}\hat{T} \quad \text{and} \quad \tilde{\sigma}_1 = \sqrt{\frac{\hat{\sigma}^2}{2}}$$

Note that these equations are in normalized units, so  $\hat{\alpha}_1 = \hat{\beta}_1 = 1$  by construction. If

Neuron 1 is the non-target, Equation 2.21 requires

$$V_{th} = \tilde{\mu}_1 + \Delta_1 = \hat{u}\hat{T} + 1.16\hat{\sigma}$$

The constant 1.16 comes from the adjusted coefficient  $\frac{1.64}{\sqrt{2}}$ . The input thus satisfies

$$\hat{u} = \frac{V_{th} - 1.16\hat{\sigma}}{\hat{T}}$$

Plugging this input into the expression for Neuron 2, Equation 2.20, yields

$$V_{th} = \tilde{\mu}_2 - \Delta_2 = \hat{\beta}_2\hat{u}\hat{T} - 1.16\frac{\hat{\sigma}}{\sqrt{\hat{\alpha}_2}}$$

which, solved for  $\hat{\alpha}_2, \hat{\beta}_2$  becomes

$$\hat{\beta}_2 = \frac{V_{th} + 1.16\frac{\hat{\sigma}}{\sqrt{\hat{\alpha}_2}}}{V_{th} - 1.16\hat{\sigma}} \quad (2.23)$$

Equation 2.23 defines a boundary equation when the input is the properly scaled impulse.

Applying this input to a Neuron 2 with any  $\hat{\alpha}_2 > 0$  and  $\hat{\beta}_2$  satisfying Equation 2.23

should producing selective spiking with Neuron 1 activity meeting the non-target

criterion. When  $\hat{\sigma} = 0$ , Equation 2.23 becomes  $\hat{\beta}_2 = 1$ , which coincides with the vertical

line in the deterministic case. However, in the region  $\hat{\alpha}_2 < 1, \hat{\beta}_2 < 1$ , this impulse is not

a good choice to make Neuron 2 spike alone; instead, a small strength-long duration input

is.

If the input duration is long enough, then from Equation 2.12, the mean approaches its

stationary value,  $\tilde{\mu}_{stat} = \frac{\beta}{\alpha}u$  (and the variance is constant), so applying a long duration

input to non-target Neuron 1, using Equation 2.21 requires

$$V_{th} = \tilde{\mu}_1 + \Delta_1 = \hat{u} + 1.16\hat{\sigma}$$

and the input is

$$\hat{u} = V_{th} - 1.16\hat{\sigma}$$

Plugging this input into the expression for Neuron 2, Equation 2.20, yields

$$V_{th} = \tilde{\mu}_2 - \Delta_2 = \frac{\hat{\beta}_2 \hat{u}}{\hat{\alpha}_2} - 1.16 \frac{\hat{\sigma}}{\sqrt{\hat{\alpha}_2}}$$

which, solved for  $\hat{\alpha}_2$ ,  $\hat{\beta}_2$  becomes

$$\hat{\beta}_2 = \hat{\alpha}_2 \frac{V_{th} + 1.16 \frac{\hat{\sigma}}{\sqrt{\hat{\alpha}_2}}}{V_{th} - 1.16\hat{\sigma}} \quad (2.24)$$

Equation 2.24 defines another boundary equation for the long duration input strategy.

When  $\hat{\sigma} = 0$ , Equation 2.24 becomes  $\hat{\beta}_2 = \hat{\alpha}_2$ , which again coincides with the deterministic case. Note also that this analysis suggests the noise level should be bounded by  $V_{th} - 1.16\hat{\sigma} > 0$  for the approximation by single-time Gaussian distributions to remain valid.

Comparing the approximated analytic LHS boundary equations, Equation 2.23 and 2.24,

to the deterministic ones, Equation 2.10, they have a similar product structure: in the

lower left corner ( $\hat{\beta}_2 < 1$ ,  $\hat{\alpha}_2 < 1$ ), we can write  $\hat{\beta}_2 = \hat{\alpha}_2 M$ , where  $M = 1$  for the

deterministic case, while  $M = \frac{V_{th} + 1.16 \frac{\hat{\sigma}}{\sqrt{\hat{\alpha}_2}}}{V_{th} - 1.16\hat{\sigma}} > 1$  for the stochastic case. Writing the

boundary as this product emphasizes that the approximated LHS boundary is on the right

of the deterministic boundary (that is, the controllable region is smaller). This structure

also holds for upper right region ( $\hat{\beta}_2 > 1$ ,  $\hat{\alpha}_2 > 1$ ), where  $\hat{\beta}_2 = M$ . We can find where

these two LHS segments intersect, which indicates the kink location, via

$$\hat{\alpha}_2 \frac{V_{th} + 1.16 \frac{\hat{\sigma}}{\sqrt{\hat{\alpha}_2}}}{V_{th} - 1.16 \hat{\sigma}} = \frac{V_{th} + 1.16 \frac{\hat{\sigma}}{\sqrt{\hat{\alpha}_2}}}{V_{th} - 1.16 \hat{\sigma}}$$

The solution is

$$\hat{\alpha}_2 = 1, \quad \hat{\beta}_2 = \frac{V_{th} + 1.16 \hat{\sigma}}{V_{th} - 1.16 \hat{\sigma}}$$

connecting the kink location to the change of input strategy.

Now I apply the same methodology to derive the approximate analytic RHS boundary, on which Neuron 1 is the target neuron and Neuron 2 is the non-target. The RHS boundary can be expressed as

$$N = \frac{V_{th} - 1.16 \frac{\hat{\sigma}}{\sqrt{\hat{\alpha}_2}}}{V_{th} + 1.16 \hat{\sigma}}, \quad V_{th} - 1.16 \frac{\hat{\sigma}}{\sqrt{\hat{\alpha}_2}} > 0 \quad (2.25)$$

$$\hat{\beta}_2 = N, \quad 0 < \hat{\alpha}_2 < 1 \quad (2.26)$$

$$\hat{\beta}_2 = \hat{\alpha}_2 N, \quad \hat{\alpha}_2 \geq 1 \quad (2.27)$$

The kink location is at  $\hat{\alpha}_2 = 1$ ,  $\hat{\beta}_2 = \frac{V_{th} - 1.16 \hat{\sigma}}{V_{th} + 1.16 \hat{\sigma}}$ . If Neuron 2 is on the left hand side of the RHS boundary, then Neuron 1 can spike alone using nonnegative constant input.

Comparing the approximated RHS stochastic boundaries, Equation 2.25 to 2.27, to the deterministic ones, Equation 2.11, the approximated boundaries again have similar product structures, and they are more restrictive ( $N < 1$ ) than the deterministic case ( $N = 1$ ). Moreover, the boundaries are continuous at the noiseless condition.

## 2.5 Noise adjusted symmetry of boundaries about the nominal neuron

Because parameter rescaling does not change controllability, knowing the boundary equations in the  $\hat{\alpha}_2 \geq 1$  region allows one to calculate reciprocal boundaries (LHS to RHS, and vice versa) in the  $\hat{\alpha}_2 < 1$  region, by changing which neuron is labeled as the nominal neuron without changing which neuron is the target. This arbitrary labeling enforces a symmetry which one can check against the approximated boundaries just derived. However, there is a technical detail in that the apparent noise level depends on the choice of nominal neuron. In this subsection, I demonstrate this symmetry, and use it to clarify the interpretation of the boundary equations from the previous subsection.

Assume without loss of generality that  $\alpha_1 < \alpha_2$ , and that the neuron pair is deterministically controllable (so  $\beta_1 < \beta_2$ ). Choosing Neuron 1 as the nominal neuron, yields rescaled parameters  $\hat{\alpha}_2 > 1, \hat{\beta}_2 > 1, \hat{\alpha}_1 = 1, \hat{\beta}_1 = 1$ , and  $\hat{\sigma} = \sigma/\sqrt{\alpha_1}$ .

Alternatively, choosing Neuron 2 as the nominal neuron yields normalized parameters (note the inverted hat)  $\check{\alpha}_1 = 1/\hat{\alpha}_2, \check{\beta}_1 = 1/\hat{\beta}_2, \check{\alpha}_2 = 1, \check{\beta}_2 = 1$ , and  $\check{\sigma} = \hat{\sigma}/\sqrt{\hat{\alpha}_2}$ . Thus to convert a boundary curve in the  $\hat{\alpha}_2 \geq 1$  region to one in the  $\hat{\alpha}_2 < 1$  region requires adjusting the normalized noise  $\check{\sigma}$  to  $\hat{\sigma}\sqrt{\hat{\alpha}_2}$  and then inverting  $\hat{\alpha}_2, \hat{\beta}_2$ .

The approximate boundaries, Equation 2.23 to 2.27, obey this symmetric relation. The LHS boundary equation in the  $\hat{\alpha}_2 \geq 1$  region is

$$\hat{\beta}_2 = \frac{V_{th} + 1.16 \frac{\hat{\sigma}}{\sqrt{\hat{\alpha}_2}}}{V_{th} - 1.16 \hat{\sigma}}.$$

Changing  $\hat{\sigma}$  to  $\hat{\sigma}\sqrt{\hat{\alpha}_2}$ , and then inverting both  $\hat{\alpha}_2, \hat{\beta}_2$  yields

$$\frac{1}{\hat{\beta}_2} = \frac{V_{th} + 1.16 \hat{\sigma}}{V_{th} - 1.16 \hat{\sigma} \sqrt{\frac{1}{\hat{\alpha}_2}}}$$

or

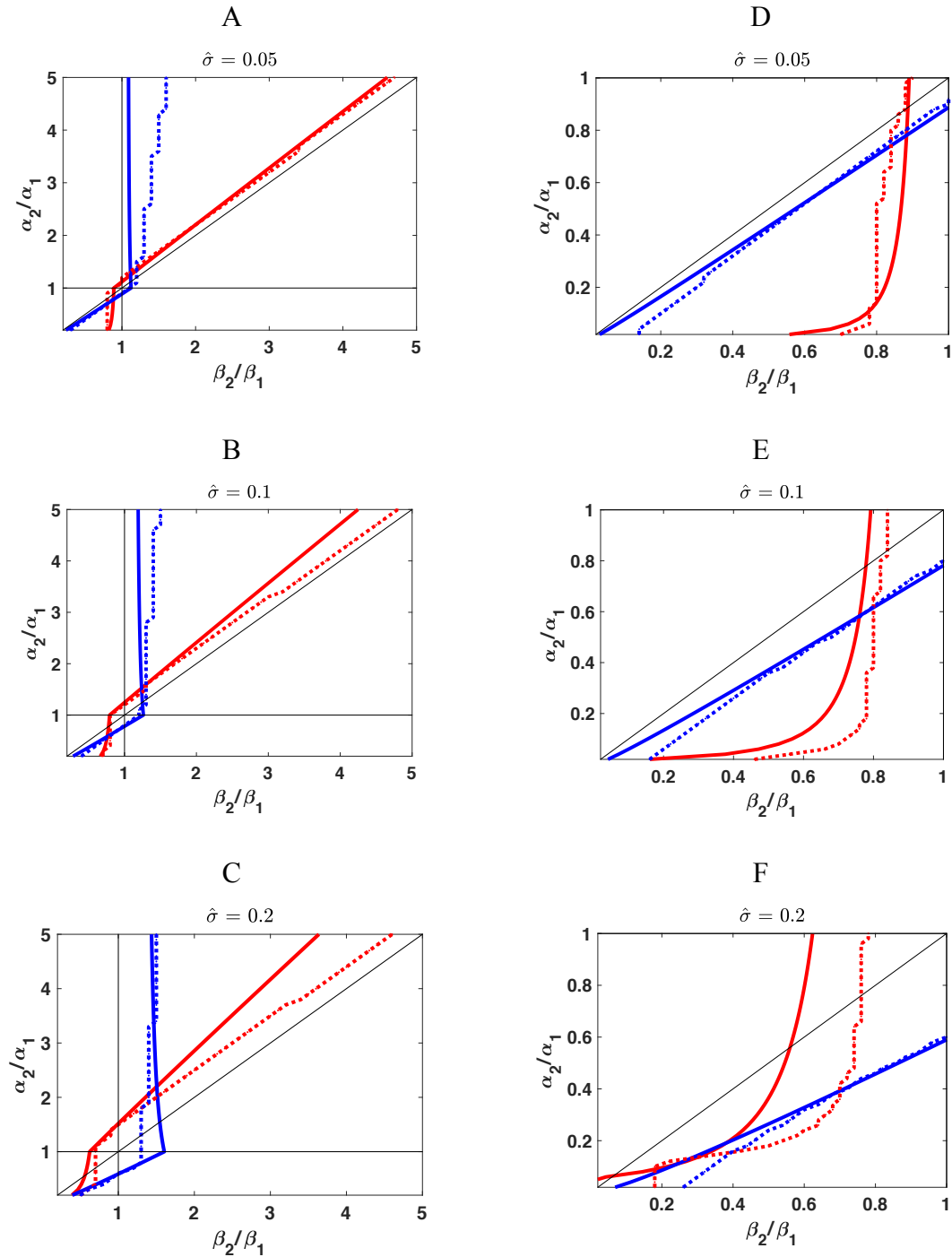
$$\hat{\beta}_2 = \frac{V_{th} - 1.16 \frac{\hat{\sigma}}{\sqrt{\hat{\alpha}_2}}}{V_{th} + 1.16 \hat{\sigma}}$$

This equation is identical to that for the RHS boundary in the lower left corner, Equation 2.26.

The other boundary pair can be shown to be symmetric following essentially identical steps. That the approximations obey the symmetry bolsters their validity, despite their non-rigorous derivation. Moreover, the form of the boundary equations is illuminated by the symmetry. In particular, the critical  $\hat{\beta}_2$  is expressed as a ratio of tail probabilities (the upper tail for the non-target neuron, and the lower tail for the target), just one of which depends on  $\hat{\alpha}_2$  through its inverse square root. This term appears precisely to account for the apparent noise level that emerges after rescaling. Whether the term appears in the numerator or denominator, and with or without a negative sign, follows whether it is the target or non-target neuron that is used for the normalization.

## **2.6 Comparison of analytically approximated and numerical derived boundaries**

In this subsection, I compare the analytic and numerical boundary curves across noise levels. Not surprisingly, the fit becomes poorer as noise increases, but the description of the numerical regions in terms of the analytic conditions that must hold at their boundaries helps explain how noise impairs pairwise controllability, even when the boundary agreement is merely qualitative.



**Figure 2.6: Numerical and analytical boundary curves at several noise levels. Solid curves: analytic boundary curves based on Equation 2.23 to 2.27. Dotted curves: numerical derived boundary curves from sections 2.2 and 2.3. Blue curves: LHS boundaries. Red curves: RHS boundaries. (A,D):  $\hat{\sigma} = 0.05$ . (B,E):  $\hat{\sigma} = 0.1$ . (C,F):  $\hat{\sigma} = 0.2$ . Right panels expand the lower left regions (below (1,1)) in the left panels.**

Using Equations 2.23 to 2.27 to calculate analytic boundaries, Figure 2.6 shows the comparison of boundaries at several noise levels. Note that there are no free parameters for the curves in this figure; the analytic curves are not fits to the numerical boundaries, but a separate method to find the boundaries. The agreement is generally good at the lowest noise level,  $\hat{\sigma} = 0.05$ . While the agreement degrades for increasing noise levels, the qualitative features are similar across all levels.

The analytical curves help explain the change of pairwise controllability in several ways. First, in the deterministic case the “kink point” of both boundaries coincide at the location of nominal neuron. In the stochastic case, this special point bifurcates into two “kink points” that move away from each other as noise increases. This boundary motion creates the totally uncontrollable region around the nominal neuron. Second, the analytic boundaries move “inward” at all points as noise level increases. That is, the defining equations show strictly decreasing controllability for all non-zero noise levels. Third, the different shapes of controllable regions in the upper-right ( $\hat{\alpha}_2 > 1, \hat{\beta}_2 > 1$ ) and lower-left ( $\hat{\alpha}_2 < 1, \hat{\beta}_2 < 1$ ) quadrants are well captured, illustrating the partial symmetry that follows from noise-adjusted relabeling of the two neurons (as in section 2.5).

## 2.7 Pathwise simulation of membrane potentials

Up to this point, I have focused on the general picture of controllability across neuron parameters, using a simple definition that considers only the probability of at least one spike occurring. I have not considered the expected number of spikes from either target or non-target neurons, or how close to threshold neurons that do not spike are at the end of an input pulse. It was then straightforward to find the required probabilities through numerical solution of the joint distribution function from the Fokker-Planck equations (sections 2.1, 2.2) or an even simpler approximation using a single-time Gaussian distributions at pulse offset (sections 2.3, 2.4). In this subsection, I use “pathwise” numerical solution of the stochastic differential equations (SDEs) to provide complementary information on the behavior of neuron pairs.

I show example membrane potential traces for fully controllable, partially isolatable, and totally uncontrollable neuron pairs. I apply inputs found during the numerical analysis in section 2.2 to every controllable neuron pair in the controllability map. I find that although neuron pairs near the boundaries show discrepancies with the prior joint density analysis, most pairs found to be controllable under the previous analysis do meet the 90% criterion. Moreover, by repeatedly solving the SDEs for different noise realizations, I am able to consider also how close the pairs are to the criterion, in the sense of estimating the number of spikes emitted by both target and non-target neurons.

I applied the Euler-Maruyama method [58] for numerical solution of stochastic differential equations (SDEs). Briefly, the method works by a simple generalization of the Euler method for ordinary differential equations to SDEs. See the code provided in the Appendix, and [58] for further details.

In Figure 2.7, I show sample traces for four neuron pairs (four different parameter choices for Neuron 2 with a fixed choice for Neuron 1) that exhibit the dynamic changes underlying loss of controllability. The parameters for Neuron 1 are  $\alpha_1 = \frac{1}{25} \text{ ms}^{-1}$ ,  $\beta_1 = \frac{11}{270}$ , and for both neurons  $\sigma = 0.04 \text{ (ms)}^{-\frac{1}{2}}$ . For Neuron 2, the remaining parameters varying with panels are:

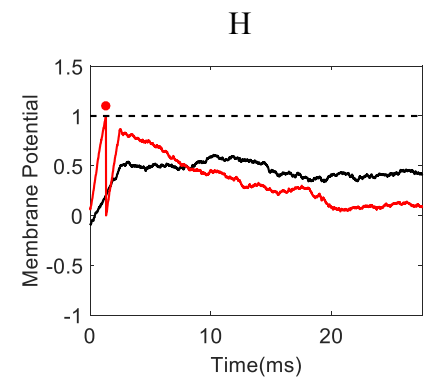
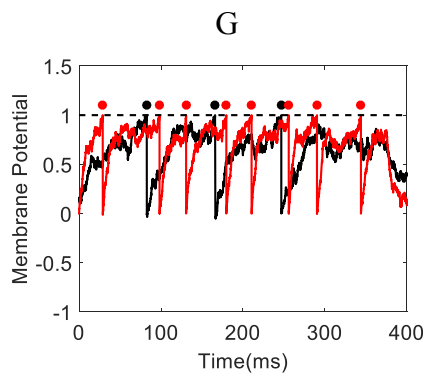
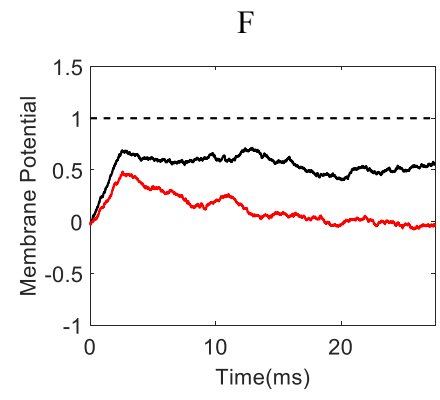
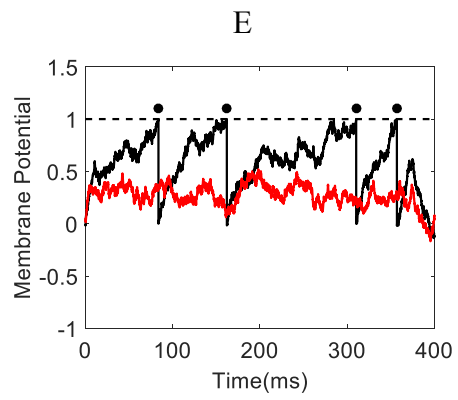
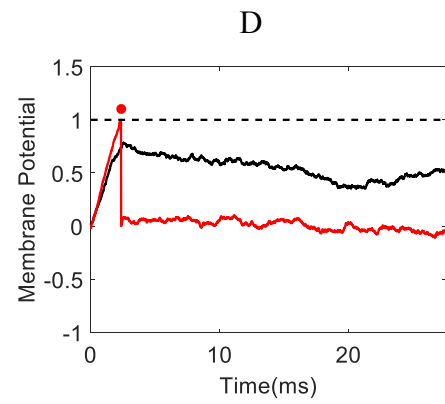
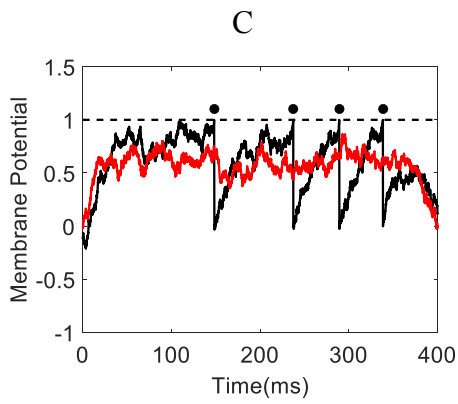
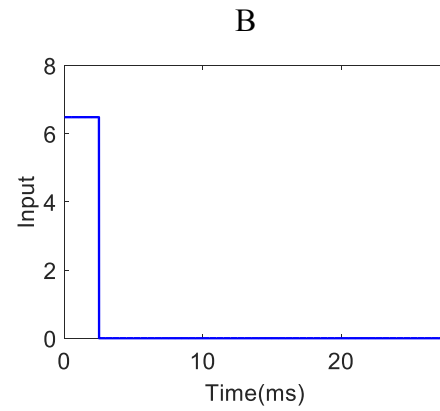
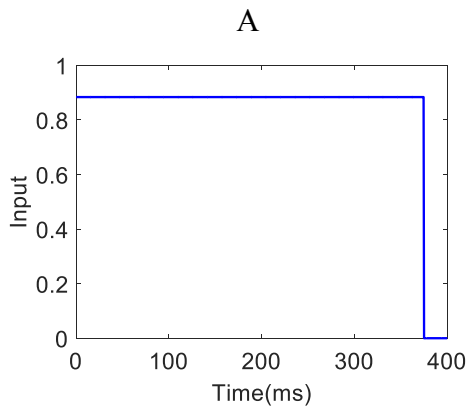
(C,D)  $\alpha_2 = \frac{3}{25} \text{ ms}^{-1}$ ,  $\beta_2 = \frac{22}{270}$ . This case is fully controllable.

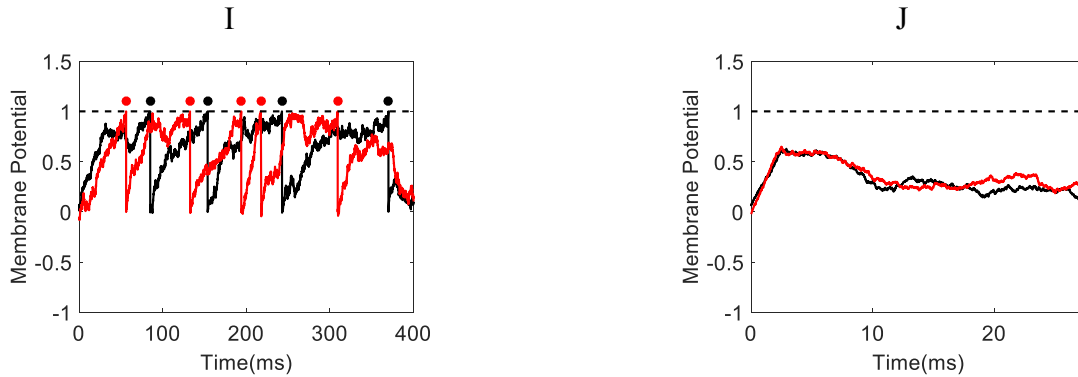
(E,F)  $\alpha_2 = \frac{3}{25} \text{ ms}^{-1}$ ,  $\beta_2 = \frac{11}{270}$ . Only Neuron 1 is isolatable.

(G,H)  $\alpha_2 = \frac{3}{25} \text{ ms}^{-1}$ ,  $\beta_2 = \frac{33}{270}$ . Only Neuron 2 is isolatable.

(I,J)  $\alpha_2 = \frac{11}{250} \text{ ms}^{-1}$ ,  $\beta_2 = \frac{121}{2700}$ . This case is totally uncontrollable; for every pulse, either both neurons spike with superthreshold probability, or neither neuron spikes.

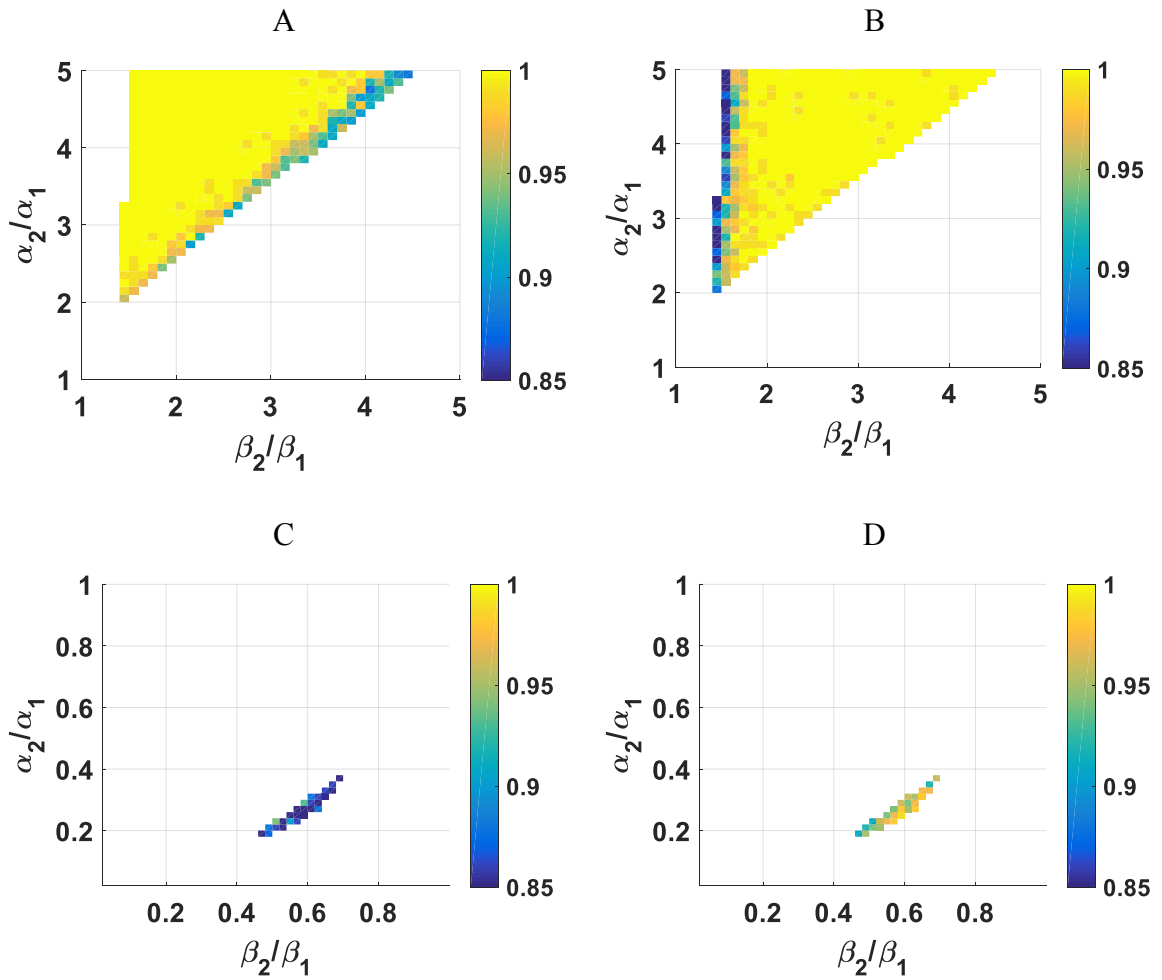
After parameter rescaling, the noise level corresponds to  $\hat{\sigma} = 0.2$ , as used in the controllability map shown in Figure 2.1. The two inputs in Figure 2.7 A,B were chosen by numerical search over the Strength-Duration pool as producing the highest criterion (Equation 2.6 and 2.7) for the controllable neuron pair (Figure 2.7 C,D).





**Figure 2.7: Example traces for four neuron pairs illustrating qualitatively different spiking outcomes. Panels on the left received an input (A) intended to make Neuron 1 spike, while panels on the right received an input (B) intended to make Neuron 2 spike. (C,D) A fully controllable neuron pair,  $\alpha_2 = \frac{3}{25} \text{ ms}^{-1}, \beta_2 = \frac{22}{270}$ . (E,F) Only Neuron 1 is isolatable,  $\alpha_2 = \frac{3}{25} \text{ ms}^{-1}, \beta_2 = \frac{11}{270}$ . (G,H) Only Neuron 2 is isolatable,  $\alpha_2 = \frac{3}{25} \text{ ms}^{-1}, \beta_2 = \frac{33}{270}$ . (I,J) A totally uncontrollable neuron pair,  $\alpha_2 = \frac{11}{250} \text{ ms}^{-1}, \beta_2 = \frac{121}{2700}$ .**

Figure 2.8 shows the percentage of “successful” (that is, superthreshold criterion, Equations 2.6, 2.7) trials out of 100 SDE simulations, for each controllable neuron pair in the previous  $\hat{\sigma} = 0.2$  map, Figure 2.1. I kept the stochastic initial conditions intact. For each pair, I picked the two inputs in the original, less sophisticated Strength-Duration pool that maximized the criteria Equations 2.6 and 2.7. The distinction between left hand and right hand boundaries is evident in the asymmetric drop in success rates at the edges of the controllable region (e.g. only the right boundary in Figure 2.8A, when Neuron 1 is the target).



**Figure 2.8: Percentages, shown as color map, of successful spike outcomes in repeated simulations of the SDE, for every controllable neuron pair in the  $\hat{\sigma} = 0.2$  map found from the joint density, Figure 2.1. (A,C) Neuron 1 is the target. (B,D) Neuron 2 is the target.**

For each input applied to a neuron pair, I placed the spike response into one of five categories:

- 1). Non-target neuron spikes at least once.
- 2). Both non-target and target neuron are silent.
- 3). Target neuron spikes once, and non-target neuron is silent.
- 4). Target neuron spikes twice, and non-target neuron is silent.

5). Target neuron spikes more than twice, and non-target neuron is silent.

Groups 3 to 5 are considered successful outcomes.

I ran simulations at parameter values used in the  $\hat{\sigma} = 0.2$  map from section 2.2, for which the neuron pair was found to be controllable. Because I set  $P_{th} = 0.9$ , if 90% of the trials result in spiking just of the target neuron, I consider the input a successful control. Figure 2.8 shows the percentage of realizations (in color; white background indicates unsimulated parameters) for which numerical solution of the SDE produced successful spiking. Not surprisingly, I found neuron pairs near the boundaries are less likely to meet this 90% threshold within a finite sample. In Figure 2.8A, (nominal) Neuron 1 is the target, and the success rate drops near the RHS boundary. In Figure 2.8B, Neuron 2 is target, the success rate drops near the LHS boundary, with many pairs not meeting the 90% criteria. Similar descriptions apply to Figure 2.8 panels C and D. Because target neurons in Figure 2.8 B and C have larger  $\alpha$ ,  $\beta$ , and  $\frac{\alpha}{\beta}$  than non-targets, the inputs are large-strength short-duration pulses, and most of the simulation time is within the silent period (for example, compare to the right panels in Figure 2.7). I speculate this large duration of silence allows incidental non-target neuron spiking.

To further elucidate the behavior of neurons within each control region, I averaged the spiking statistics within each group of “controllable” pixels, separately for each panel in Figure 2.8.

	Target: Neuron 1 $\hat{\alpha}_2, \hat{\beta}_2 \in [1,5]$ (Fig. 2.8A)	Target: Neuron 2 $\hat{\alpha}_2, \hat{\beta}_2 \in [1,5]$ (Fig. 2.8B)	Target: Neuron 1 $\hat{\alpha}_2, \hat{\beta}_2 \in [0.02,1]$ (Fig. 2.8C)	Target: Neuron 2 $\hat{\alpha}_2, \hat{\beta}_2 \in [0.02,1]$ (Fig. 2.8D)
(1) Non-target neuron spikes	0.36%	1.31%	12.04%	2.06%
(2) Both neurons silent	1.04%	0.17%	3.06%	3.10%
(3) Target neuron spikes once	5.36%	98.53%	84.90%	22.88%
(4) Target neuron spikes twice	11.35%	0.00%	0.00%	45.71%
(5) Target neuron spikes more than twice	81.89%	0.00%	0.00%	26.25%

Table 2.1: Averaged statistics of different spike outcome groups

Looking at the first column, for nearly 82% of trials on average, the target neuron spikes more than twice, and the target neuron spikes at least once in 98.6% of trials. This value is well above the  $P_{th} = 0.9$  criterion, and also the  $\sqrt{P_{th}} \sim 0.95$  used in the derivation of approximate boundaries. The loss of controllability along the LHS boundary is instead due mainly to unwanted spikes from the non-target neuron. For neuron pairs not close to boundaries, control is generally successful. A similar but more severe issue underlies poor controllability when neuron 1 is the target but neuron 2 is to the bottom left, as shown in the third column. Although the SDE simulations show quantitative disagreement with the prior density analysis for neuron pairs near the boundaries, the majority of controllable neuron pairs produce desired spike outputs above the 0.9 probability criterion.

## **CHAPTER THREE-Positive ramp inputs restore controllability for a subset of single pulse uncontrollable pairs**

### **3.1 Motivation for the expansion of input class**

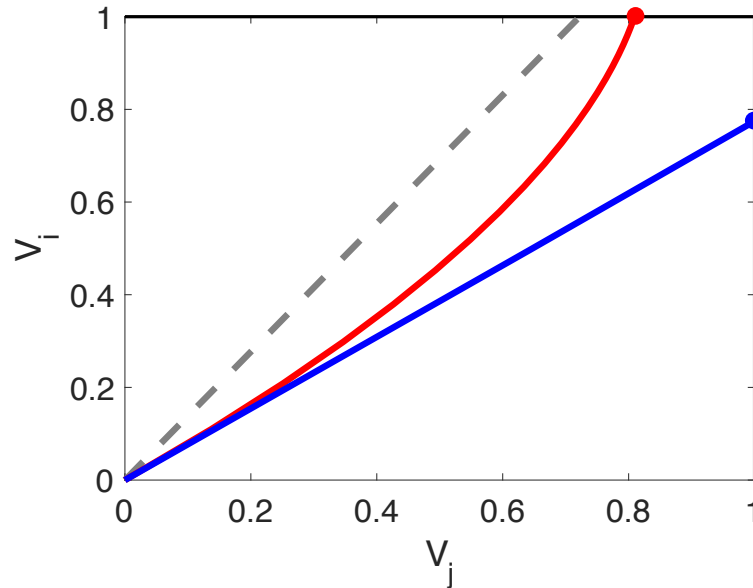
Up to this point, this dissertation has defined controllability solely in terms of an input space of single pulses. This restriction on inputs is strong but not without motivation. For example, single pulses (possibly with an initial “off” period) are time optimal for deterministic pairwise control of integrate and fire neurons [56], a special case of the more general result that for endpoint control of linear systems (e.g. lacking regularization or “energy” constraints), bang-bang inputs are typically time optimal [17]. Moreover, integrate and fire neurons are strictly monotonic in their inputs, in the sense that increasing the input always increases depolarization rates, so strategies available in some more complicated systems, such as pre-pulse inactivation [59], do not apply. Thus, a larger input space does not improve controllability in the deterministic case, and it is not obvious that stochastic spike control can be improved by expanding the input space. However, in this chapter, I develop a positive ramp input strategy that can control a subset of neuron pairs that are not controllable with pulsatile inputs. I first motivate the ramps by comparison to the deterministic case, and describe the applicability and limitation of this strategy. Then I apply this input strategy to single-pulse uncontrollable neuron pairs, and find parameters for which controllability is restored.

### 3.2 Derivation of ramp inputs and numerical assessment

Before applying the ramp strategy, I review the geometric interpretation of deterministic pairwise control, that motivates this ramp strategy. First assume labels are chosen so that Neuron  $i$  and Neuron  $j$  satisfy  $\alpha_i < \alpha_j$ ,  $\beta_i < \beta_j$ , and  $\alpha_i/\beta_i < \alpha_j/\beta_j$ ; these conditions correspond to a deterministically controllable neuron pair [46]. For any constant input  $u$ , there is a unique equilibrium vector  $V_{eq}$  for  $(V_i, V_j)$ :

$$V_{eq} = \begin{pmatrix} \beta_i/\alpha_i \\ \beta_j/\alpha_j \end{pmatrix} u \quad (3.1)$$

From previous results [46], we can choose the single-pulse input for targeting neuron  $i$  to have strength  $\alpha_i/\beta_i < u < \alpha_j/\beta_j$ , and choose strength  $u = u_{max} \gg \alpha_j/\beta_j$  to target neuron  $j$ . For these choices,  $V_{eq}$  will traverse a line above the diagonal in the  $(V_i, V_j)$  phase plane, and solutions for the two inputs will curve upward, as shown in Figure 3.1. In particular, when Neuron  $j$  is the target the input is a short duration high amplitude pulse, and we cannot lower the deterministic solution below the blue curve, which gives the solution for  $u = u_{max}$ . Without this cap on the input, the linear control problem (for example, for time minimal control [56]) tends to select an ‘‘impulsive bang’’ input with  $u_{max} \rightarrow \infty$ , a biologically unrealistic solution. In experimental applications, some limit on input amplitude is imposed, and what matters here is that the limit exists, rather than its precise value.



**Figure 3.1: Geometric interpretation of deterministic pairwise control using single pulse inputs. Dashed gray line:  $V_{eq}$ . Solid red line:  $V_i, V_j$  trace under input with Neuron i as the target. Solid blue line:  $V_i, V_j$  trace under input with Neuron j as the target.**

In contrast, for the input pulse when  $V_i$  is the target the input is a long duration low amplitude pulse. The instantaneous equilibrium  $V_{eq}(u)$  during the pulse lies to the left of the threshold  $V_j = 1$  and above the threshold  $V_i = 1$ . The solution curve (red) will thus hit the  $V_i$  threshold somewhere between its intersection with the  $V_{eq}$  line and the corner  $(V_i, V_j) = (1, 1)$ . On the way, the solution bows away from the  $V_{eq}$  line, and this provides the motivation for the ramp strategy. Intuitively, if the red curve is too close to the threshold for the non-target neuron, a fluctuation in the corresponding stochastic case may kick the solution over that threshold and cause non-target spiking. A different choice of input that produces a solution closer to the  $V_{eq}$  line may sufficiently decrease the probability of such superthreshold fluctuations to meet the stochastic control criterion.

I thus analytically construct a ramp input that makes the deterministic trace  $(V_i, V_j)$  rise arbitrarily close to  $V_{eq}$ , and then numerically assess the performance of this input when applied to the stochastic system. For simplicity, I rescale the parameters as  $\hat{\alpha}_i = 1$ ,  $\hat{\beta}_i =$

1,  $\hat{\alpha}_j = \alpha_j/\alpha_i > 1$ , and  $\hat{\beta}_j = \beta_j/\beta_i > 1$ . Then  $V_{eq} = \begin{pmatrix} V_i^{eq} \\ V_j^{eq} \end{pmatrix}$  can be expressed as

$$V_i^{eq} = \frac{\hat{\alpha}_j}{\hat{\beta}_j} V_j^{eq} \quad (3.2)$$

where  $\frac{\hat{\alpha}_j}{\hat{\beta}_j} > 1$ . There is no nonnegative input that can make  $(V_i, V_j)$  progress exactly along  $V_{eq}$ . However, for every sufficiently small  $\epsilon > 0$ , there is a  $u(t) > 0$  that drives  $(V_i, V_j)$  along a line of the form

$$V_i = \left( \frac{\hat{\alpha}_j}{\hat{\beta}_j} - \epsilon \right) V_j \quad (3.3)$$

which is a line arbitrarily close to the equilibrium line. To find this input, suppose the initial conditions of  $(V_i, V_j)$  is some point already on the line,

$$\begin{cases} V_i = \left( \frac{\hat{\alpha}_j}{\hat{\beta}_j} - \epsilon \right) V^* \\ V_j = V^* \end{cases} \quad (3.4)$$

where  $V^* > 0$  (see also below). To keep the solution on this line, I seek an input such that

$V_i = \left( \frac{\hat{\alpha}_j}{\hat{\beta}_j} - \epsilon \right) V_j$  and such that both membrane potentials have positive derivative.

Plugging the constraint into the ODE and simplifying yields

$$u(t) = \frac{\left( \frac{\hat{\alpha}_j}{\hat{\beta}_j} - \epsilon \right) (\hat{\alpha}_j - 1)}{\hat{\alpha}_j - \hat{\beta}_j \epsilon - 1} V_j(t) \quad (3.5)$$

While this input appears to be closed loop, we can plug it back into the ODE for some

fixed choice of  $V^*$  and explicitly solve the linear equation to define an open loop input

$$u(t) = Ce^{nt} \quad (3.6)$$

where  $C = \frac{\left(\frac{\hat{\alpha}_j}{\hat{\beta}_j} - \epsilon\right)(\hat{\alpha}_j - 1)}{\hat{\alpha}_j - \hat{\beta}_j \epsilon - 1} V^*$  and  $n = \frac{\hat{\beta}_j \epsilon}{\hat{\alpha}_j - \hat{\beta}_j \epsilon - 1}$ .

If we set  $\epsilon = 0$ , then  $u = \frac{\hat{\alpha}_j}{\hat{\beta}_j} V_j$  and  $\frac{dV_i}{dt} = \frac{dV_j}{dt} = 0$ , which confirms  $(V_i, V_j)$  cannot rise

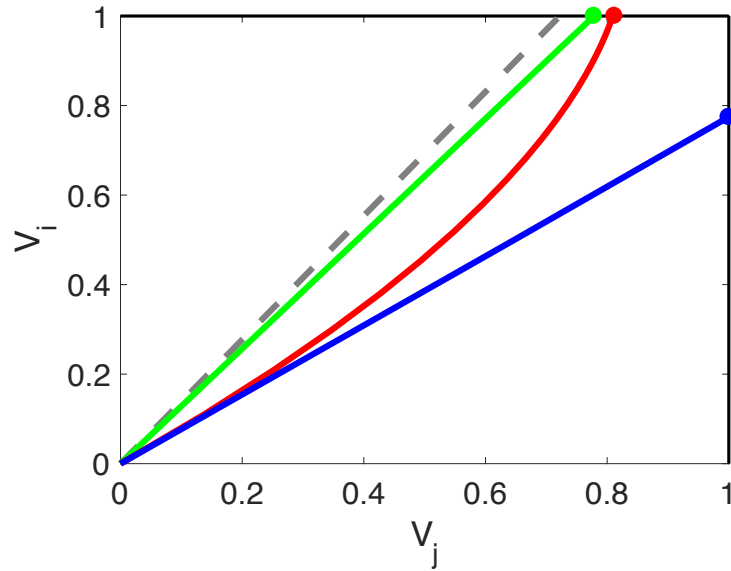
along  $V_{eq}$ . In order to simultaneously produce  $\frac{dV_i}{dt} > 0$  and  $\frac{dV_j}{dt} > 0$ ,  $\epsilon$  must satisfy

$$0 < \epsilon < \epsilon_{max} = \frac{\hat{\alpha}_j}{\hat{\beta}_j} - 1 \quad (3.7)$$

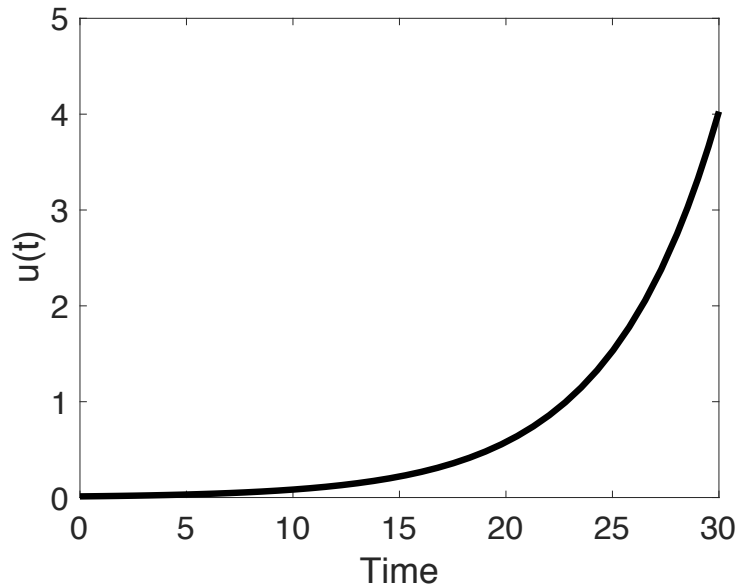
Note  $\epsilon_{max}$  is a function of  $\hat{\alpha}_j, \hat{\beta}_j$  that requires the deterministic controllability conditions to be met in order to be positive. The smaller  $\epsilon$  is, the closer the deterministic solution  $(V_i, V_j)$  progresses to the  $V_{eq}$  line, but with the tradeoff that the duration of  $u(t)$  will be longer, as seen by looking at the time constant in Equation (3.6). Regardless of responses, the ramp input turns off when the deterministic neuron would have reached threshold.

There is a technical issue that the initial condition in Equation 3.4 is already nonzero, leaving undefined an input that could drive neurons sitting at rest (the issue is similar to that noted in [46] when applying deterministic control for conductance inputs). However, under noise, there is zero probability of initial conditions being strictly at the origin, and we might hope that for any appropriate but arbitrary choice of  $V^* > 0$ , the ramp input will prove successful for a small ball of initial conditions.

A



B



**Figure 3.2. A ramp input and the deterministic membrane potentials. A:** Same as Figure 3.1, adding a green trace for  $(V_i, V_j)$  under ramp input. **B:** The ramp input  $u(t)$ .

Figure 3.2 shows an example deterministic solution and ramp input. Compared to the original red curve for a single pulse input, the ramp input makes the deterministic  $(V_i, V_j)$

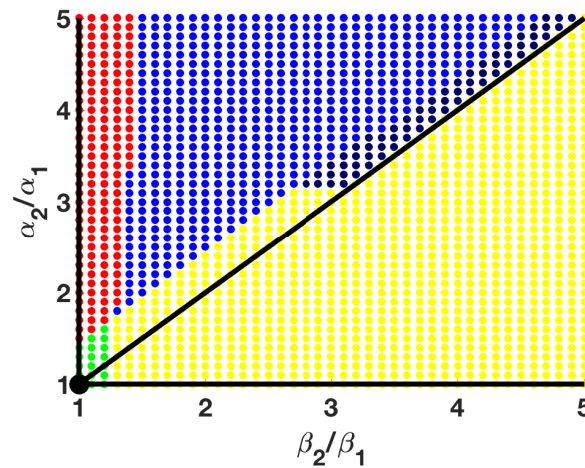
rise along a line closer to  $V_{eq}$ . With neuron  $i$  as the target, I conjecture this new route would be safer in the stochastic case, as it keeps the non-target neuron  $j$  further from its threshold throughout the trajectory. This argument is not rigorous, as, for example, although the trajectory is farther from the non-target threshold, it takes more time for the target to reach its threshold, so that the probability of aberrant fluctuations may actually increase.

In order to assess any possible improvement in control using ramp inputs, I applied this strategy to a subset of single-pulse uncontrollable pairs in  $\hat{\sigma} = 0.2$  map (as in Figure 2.1). Recall that the ramps are designed to replace long duration, small amplitude pulses, and are not expected to provide any benefit for short duration, high amplitude pulses. Thus ramps could provide improved control only when the neuron with smaller  $\alpha$ , and  $\beta$  is the target. In normalized parameters, this means neuron 1 is the target if  $\hat{\alpha}_2 > 1, \hat{\beta}_2 > 1$ , and we need consider only those pixels for which neuron 2 is already isolatable. Similarly, ramp inputs applied when neuron 2 is the target may recover controllability for pairs in  $\hat{\alpha}_2 < 1, \hat{\beta}_2 < 1$  for which neuron 1 is already isolatable. I use the same probabilistic criterion,  $(1 - G_{target}) \times G_{non-target} \geq P_{th}$  as in single pulse control.

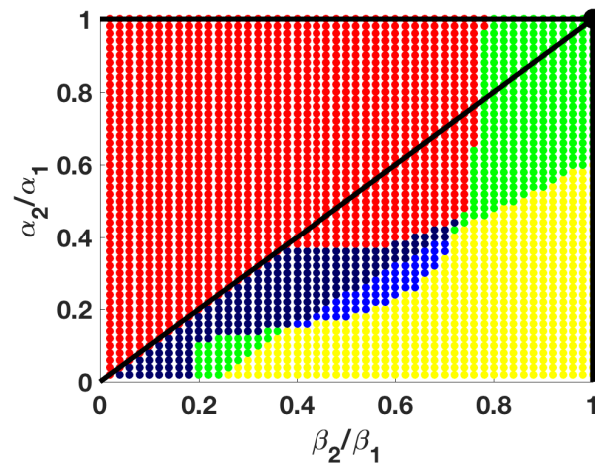
Figure 3.3 shows the parameter map of controllability for this expanded space of controls. Following Figure 2.1, I applied this ramp strategy to only Neuron 2 isolatable and deterministically controllable pairs in upper right region (that is, the yellow pixels in  $\hat{\alpha}_2 > 1, \hat{\beta}_2 > 1, \frac{\hat{\alpha}_2}{\hat{\beta}_2} > 1$  region), and only Neuron 1 isolatable and deterministically

controllable pairs in lower left region (that is, the red pixels in  $\hat{\alpha}_2 < 1, \hat{\beta}_2 < 1, \frac{\hat{\alpha}_2}{\hat{\beta}_2} < 1$  region). I set the closeness parameter  $\epsilon = 0.1\epsilon_{max}$ . The expansion of pixels for which the pair is controllable demonstrates that the ramp strategy can control a subset of single-pulse uncontrollable pairs in the stochastic case.

A



B



**Figure 3.3: Controllability map and parameters for which ramp inputs control single-pulse uncontrollable neuron pairs. A:  $\hat{\alpha}_2, \hat{\beta}_2 \in [1, 5]$ . The dark blue dots are pairwise controllable pairs using ramp input strategy. All other pixels are the same as Figure 2.1. B:  $\hat{\alpha}_2, \hat{\beta}_2 \in [0.02, 1]$ . The dark blue dots are pairwise controllable pairs using ramp input strategy. All other pixels are the same as Figure 2.1.**

## **CHAPTER FOUR- Mean spike time is a coarse definition of stochastic controllability**

### **4.1 Alternative definitions of controllability**

In the previous chapters, stochastic controllability was defined in terms of the probability of spiking within the duration of pulse (or ramp) inputs. An alternative approach is to define controllability in terms of mean spike timing. In the deterministic case, the two definitions overlap, since inputs can be chosen to be only as long as the target neuron's spike time. In the stochastic case, there is no single spike time associated to a given input, leaving open the choice of mixture of spike timing and probability of spiking in the criterion function.

Generalizing from the deterministic emphasis on spike order control (see Section 1.3), one can define a pair of neurons as controllable if their mean spike times can be arranged in either order through suitable choice of input. For the integrate and fire model, this choice reduces control to the well-studied and tractable problem of finding the mean exit time. More specifically, I redefine a neuron pair at a certain noise level to be stochastic controllable if their mean exit time curves, as a function of input strength, cross at some positive input value. However, I show that mean exit time alone does not adequately capture the temporal dynamics of spiking probability, and conclude that using mean exit time alone is too coarse measurement of the stochastic controllability.

## 4.2 Defining and solving mean spike time control

Survival analysis of stochastic systems concerns the expected duration until one or more events happen, such as death of biological organisms or failure of mechanical systems. Mean exit time (or first-hitting-time) analysis, is a subclass of survival analysis that concerns the amount of time required for a stochastic process, starting from some initial state, to hit a threshold for the first time.

One advantage of casting controllability in terms of ordering of mean spike times is that there is a long history of work on this problem, including for applications of diffusion processes in economics and physics [60, 61, 62, 63]. Mean exit time analysis has been applied to neuronal model dynamics, and specifically to the integrate and fire model. For example, in [64], moments of the firing time are explicitly obtained. [65] presents result on the mean exit time of a leaky integrate and fire neuron with deterministic subthreshold dynamics and a firing threshold that evolves as an Ornstein-Uhlenbeck process. In [66], the mean exit time is used to examine the effect of a random initial value on several stochastic integrate and fire neural models with constant threshold and constant input.

Another motivation for considering mean spike timing is that, similar to the case of spiking probability as in previous sections, the contributions of the two neurons can be calculated separately. Asking instead which neuron spikes first on a trial to trial basis would require consideration of the joint density of spike times. It is not obvious which criterion might have the greatest biological relevance. However, by considering a

sequence of interspike intervals, one could also interpret mean spike time control as being a rate control strategy. That is, by altering inputs to choose a neuron with the lower mean time, one is selecting a neuron to have a faster spike rate.

I redefine the controllability criterion, under the assumption that for every positive pulse amplitude  $u$ , a mean time to spike  $T(u; \sigma)$  is well defined for both neurons (note that the time depends also on neural parameters).

*Definition 4.1 (Stochastic Pairwise Controllability Using Mean Spike time):* Suppose  $T_1(u; \sigma)$  and  $T_2(u; \sigma)$  are mean spike times for Neuron 1 and Neuron 2 under a constant input with amplitude  $u$ . If there are two positive inputs,  $u_1 \neq u_2$ , such that  $T_1(u_1; \sigma) < T_2(u_1; \sigma)$  and  $T_1(u_2; \sigma) > T_2(u_2; \sigma)$ , we say Neuron 1 and Neuron 2 are stochastic pairwise controllable under noise level  $\sigma$ .

The criterion holds if the two curves  $T_1(u; \sigma)$  and  $T_2(u; \sigma)$  cross in at least one point  $u \geq 0$ . My approach is first to discuss the zero input case,  $u = 0$ , while varying other parameters, and use those results to find conditions for this crossing. Without loss of generality, I assume  $\alpha_1 > \alpha_2$ , so that Neuron 1 is leakier.

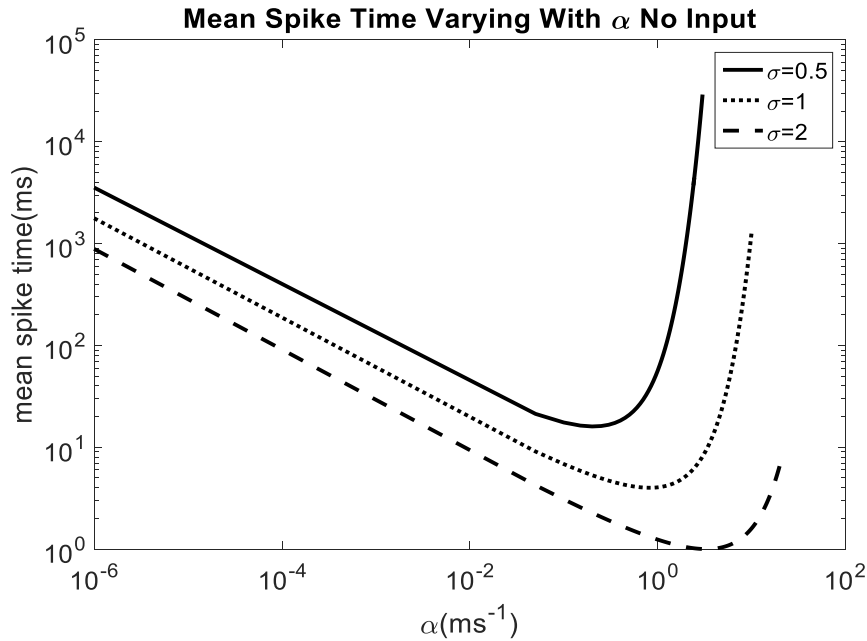
When the input is zero and the (deterministic) initial condition at rest, the mean spike time of a noisy integrate and fire neuron has an explicit, if complicated, expression, see [57]

$$T(0; \sigma) = \frac{V_{th}^2}{\sigma^2} F\left(1, 1; \frac{3}{2}, 2; \alpha \frac{V_{th}^2}{\sigma^2}\right) + \frac{\pi}{2\alpha} \operatorname{erfi}\left(\frac{V_{th}\sqrt{\alpha}}{\sigma}\right) \quad (4.1)$$

$F\left(1, 1; \frac{3}{2}, 2; x\right)$  is a generalized hypergeometric function and  $\operatorname{erfi}(x)$  is the imaginary error function. For more information about how to calculate mean exit time function, see [57]. In all figures below, I numerically evaluate the curves using Matlab.

For every neuron, it is intuitively clear that  $T(0; \sigma) = \infty$  when  $\sigma \rightarrow 0$  (no spiking in the absence of input and noise) and  $T(0; \sigma) = 0$  when  $\sigma \rightarrow \infty$  (arbitrarily high noise drives instant threshold crossings). Moreover,  $T(0; \sigma)$  is a strictly decreasing function of  $\sigma$ .

From Equation 4.1, calculation shows that  $T(0; \sigma) \rightarrow \infty$  as  $\alpha \rightarrow \infty$  and also as  $\alpha \rightarrow 0$ , with all other parameters fixed. Figure 4.1 shows  $T(0; \sigma)$  as a function of  $\alpha$  under several different noise intensities.

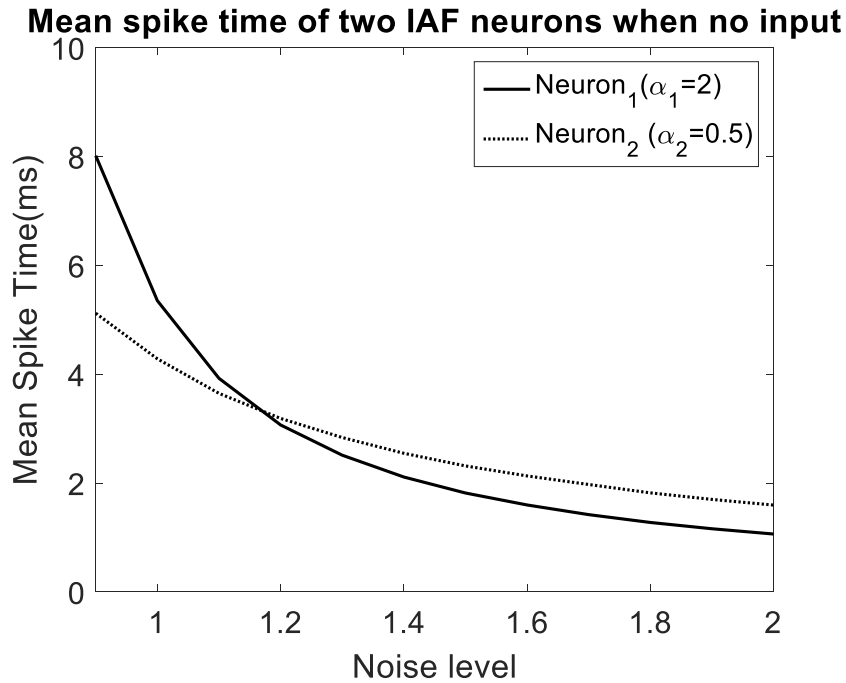


**Figure 4.1: Mean spike time under zero input  $T(0)$  for three noise levels,  $\sigma = 0.5, 1, 2$ , as a function of  $\alpha$ .**

These calculations show that, for each  $\sigma$ ,  $T(0)$  has a global minimum as a function of  $\alpha$ . Its location can be found by setting the derivative (with respect to  $\alpha$ ) of  $T(0)$  to 0, which yields the implicit equation

$$\frac{\pi}{2\alpha_{min}^2} \operatorname{erfi}\left(\frac{V_{th}\sqrt{\alpha_{min}}}{\sigma}\right) = \frac{V_{th}^4}{3\sigma^4} F\left(2, 2; \frac{5}{2}, 3; \frac{\alpha_{min}V_{th}^2}{\sigma^2}\right) + \frac{V_{th}\sqrt{\pi}}{2\sigma\sqrt{\alpha_{min}^3}} e^{\left(\frac{\alpha_{min}V_{th}^2}{\sigma^2}\right)} \quad (4.2)$$

$\alpha_{min}$  is the global minimum. Expanding Equation 4.2 in a Taylor series in powers of  $\sigma$ , one can show  $\alpha_{min} \rightarrow \infty$  as  $\sigma \rightarrow \infty$ , and  $\alpha_{min} \rightarrow 0$  as  $\sigma \rightarrow 0$ . Numerical exploration suggests the relationship between  $\alpha_{min}$  and  $\sigma$  is, in fact, monotonic. In that case, for any neuron pair with  $\alpha_1 > \alpha_2$ , there is an  $\hat{\sigma}$ , such that  $\sigma < \hat{\sigma}$  implies  $T_1(0; \sigma) > T_2(0; \sigma)$ , while  $\sigma > \hat{\sigma}$  implies  $T_1(0; \sigma) < T_2(0; \sigma)$ . In other words, the mean spike time curves cross, considered as functions of  $\sigma$ . Figure 4.2 gives an example.



**Figure 4.2:** Mean spike times of two neurons under zero input.  $\alpha_1 = 2$  and  $\alpha_2 = 0.5$  as a function of  $\sigma$  are shown. When noise is small,  $T_{u=0}(\alpha_1; \sigma) > T_{u=0}(\alpha_2; \sigma)$ , when noise is large,  $T_{u=0}(\alpha_1; \sigma) < T_{u=0}(\alpha_2; \sigma)$ .

Next, I discuss when  $u > 0$  and combine with the  $u = 0$  results. First consider when the noise is small,  $\sigma < \dot{\sigma}$ , so that  $T_1(0; \sigma) > T_2(0; \sigma)$ ; neuron 2 has an earlier mean spike time under no input. I seek criteria that allows  $T_1(u; \sigma) < T_2(u; \sigma)$  for some input  $u$ . The integrate and fire model can be rewritten as

$$\frac{dV}{dt} = -\alpha V + \sigma \bar{\varepsilon} \quad (4.3)$$

that is, as a neuron under Gaussian noise  $\bar{\varepsilon}$  with mean  $\bar{\mu} = \beta u$  ( $u$  is constant) and standard deviation  $\sigma$ . When  $u = 0$  and  $\sigma < \dot{\sigma}$ , one has  $T_1(0; \sigma) > T_2(0; \sigma)$  because the less leaky neuron 2 diffuses to  $V_{th}$  faster than neuron 1. If  $\beta_2 u \geq \beta_1 u > 0$ , neuron 2 still hits  $V_{th}$  faster because it receives noisy inputs with a higher average value. Thus only if  $\beta_1 u > \beta_2 u$  does neuron 1 have a chance to hit  $V_{th}$  faster than Neuron 2. This condition requires  $\beta_1 > \beta_2$ .

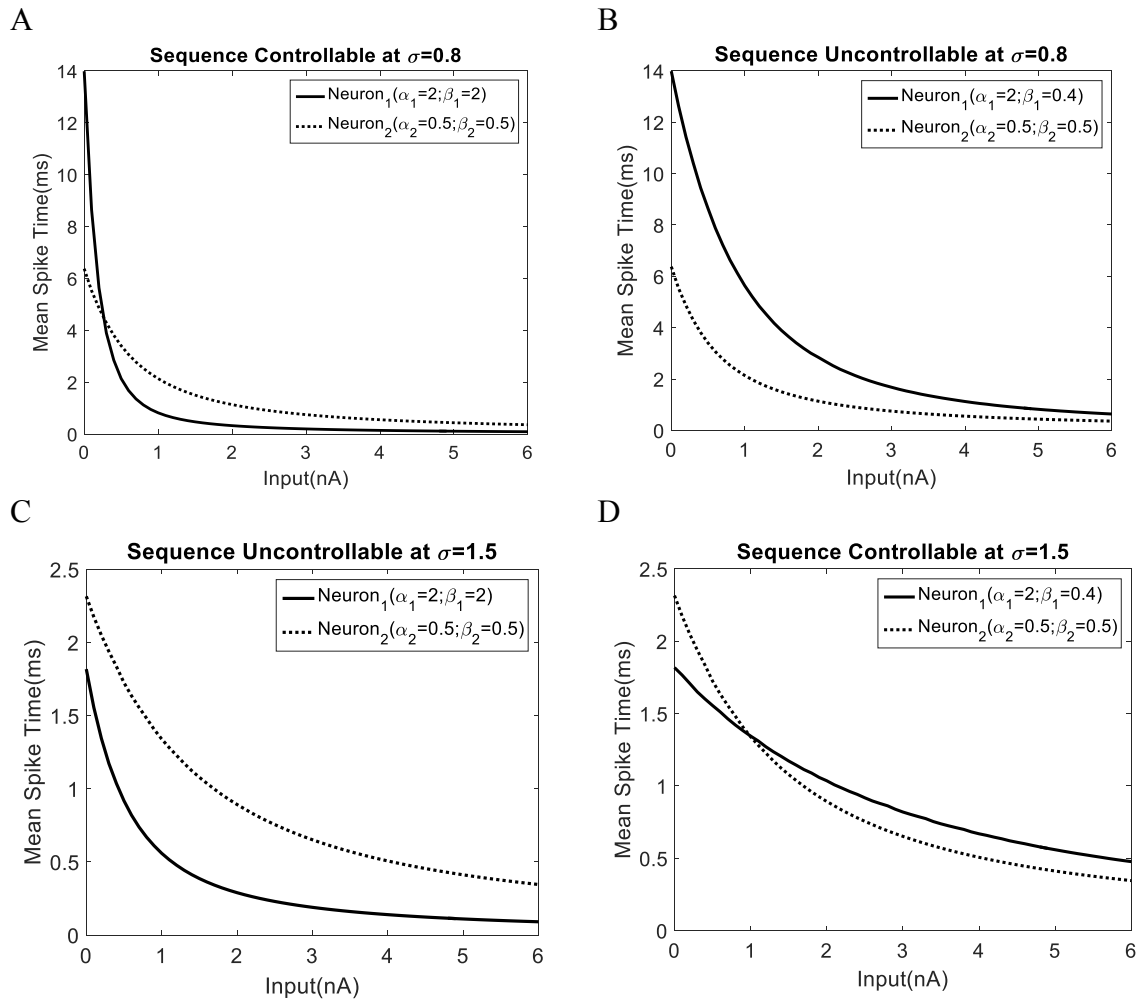
The integrate and fire neuronal model can also be rewritten as:

$$\frac{d\hat{V}}{dt} = -\alpha \hat{V} + \sigma \varepsilon, \hat{V} = V - \frac{\beta u}{\alpha} \quad (4.4)$$

where the new variable  $\hat{V}$  has initial condition:  $\hat{V}_0 = V_{rest} - \frac{\beta u}{\alpha}$ , and  $\hat{V}_{th} = V_{th} - \frac{\beta u}{\alpha}$ .

When  $\frac{\beta_1 u}{\alpha_1} \geq \frac{\beta_2 u}{\alpha_2} > V_{th}$ ,  $\hat{V}_1$  and  $\hat{V}_2$  have negative values, and  $\hat{V}_1$  has a more negative initial condition and threshold than  $\hat{V}_2$ . Note the distance between the new threshold  $\hat{V}_{th}$  and new initial condition  $\hat{V}_0$  is not changed. In this case,  $-\alpha_1 \hat{V}_1 > -\alpha_2 \hat{V}_2 > 0$ , so the leakier neuron 1 hits the threshold sooner. So if  $\frac{\alpha_1}{\beta_1} \leq \frac{\alpha_2}{\beta_2}$ , which implies  $\beta_1 \gg \beta_2$ , a large input strength  $u$  could make Neuron 1 spike first.

Following the same approach, I seek conditions when the noise is large,  $\sigma \geq \hat{\sigma}$ , such that  $T_1(0; \sigma) < T_2(0; \sigma)$ , under which a positive constant  $u > 0$  can reverse the inequality,  $T_1(u; \sigma) > T_2(u; \sigma)$ . Based on numerical evidence, I conjecture that  $\frac{\alpha_1}{\beta_1} > \frac{\alpha_2}{\beta_2}$  and  $\beta_1 \leq \beta_2$ , in contrast to the small noise case, allow the mean spike time curves to cross. Figure 4.3 shows examples in support of this conjecture. However, I have been unable to find an analytical argument in support of these conditions.



**Figure 4.3: Examples of different cases of sequence controllability using mean spike time. A.  $\alpha_1 = 2, \beta_1 = 2; \alpha_2 = 0.5, \beta_2 = 0.5; \sigma = 0.8$ , This pair is sequence controllable. B.  $\alpha_1 = 2, \beta_1 = 0.4; \alpha_2 = 0.5, \beta_2 = 0.5; \sigma = 0.8$ , This pair is sequence uncontrollable. C.  $\sigma = 1.5$ ,**

**other parameters are the same as A, The elevated noise makes the pair uncontrollable. D.  $\sigma = 1.5$ , and now the pair becomes controllable.**

Because the crossing conditions appear different for the cases  $T_1(0; \sigma) < T_2(0; \sigma)$  and  $T_1(0; \sigma) > T_2(0; \sigma)$ , the same neuron pair (in terms of  $\alpha$  and  $\beta$ ) could have different controllability status under different noise levels. For example, in Figure 4.3AC a neuron pair is sequence controllable under small noise, but become sequence uncontrollable when noise increases, but conversely in Figure 4.3BD a neuron pair becomes controllable under the larger noise.

## CHAPTER FIVE- Discussion

### 5.1 Major contributions

The primary contribution of this work is to elucidate pairwise underactuated neuron controllability in stochastic settings, by framing the problem as a bifurcation from the deterministic setting, with noise as the bifurcation parameter. I extended a pairwise controllability definition from the deterministic to stochastic case given a probabilistic parameter  $P_{th}$  which trades the probability of spiking from the target neuron with tolerance of spiking from the non-target neuron. I set  $P_{th} = 0.9$  for most of this dissertation, which implies that if a neuron pair achieves controllability, both the spike probability of the target neuron has to be larger than 0.9, and the spike probability of the non-target neuron has to be smaller than 0.1. This probabilistic definition is connected to the survival probability, which can be numerically calculated by solving a Fokker-Planck (FP) equation [57], using standard numerical schemes to solve it [1]. I incorporated the FP solutions within brute force search to delineate the pairwise controllability map under chosen noise levels.

I found neuron pairs near the deterministic boundaries are uncontrollable under noise, and I explain the failure modes through numerical simulations. Comparing to the deterministic controllability map, the stochastic one shows novel characteristics. First, the controllability map is not symmetric under a fixed noise level, as the controllable region in the lower-left quadrant ( $\hat{\alpha}_2 < 1, \hat{\beta}_2 < 1$ ) shrinks more substantially than in the upper-right quadrant. In this region, the non-nominal neuron suffers from more noisy

perturbations since the membrane potential standard deviation is larger when  $\hat{\alpha}_2$  is small. Second, if  $\hat{\alpha}_2 \approx 1$  ( $\alpha_1 \approx \alpha_2$ ), neuron pairs cannot be controllable no matter what  $\hat{\beta}_2$  is, although we can have partial isolatability at certain  $\hat{\beta}_2$ . Since  $\beta$  measures the responsiveness of neuron to stimulus, and varies with distance between neuron cell and stimulation, one can examine the stimulus location to check if  $\hat{\beta}_2$  resides within the controllable region for  $\hat{\alpha}_2$  is not close to 1. However, if  $\hat{\alpha}_2 \approx 1$ , we can only make one neuron isolatable at once by placing the stimulus closer to the target neuron. Third, I extract the stochastic controllability boundaries from the map and analyze differences and similarities with deterministic boundaries. Both of them show kinked structures, where the kinks indicate a change of input strategy, either from long-weak pulses to short-strong pulses or vice versa, to make the target neuron spike. However, instead of two segmented straight lines as the deterministic boundaries appear to be, the stochastic boundaries show distinct curvatures, especially in the lower-left quadrant, which indicates the boundary segments are no longer following linear equations.

To further analyze how controllability boundaries change with noise, I derive a set of approximate analytic equations by getting rid of the absorbing boundary conditions, which impede us solving the survival function analytically, after this simplification, the membrane potential is an Ornstein-Uhlenbeck process, it follows a Gaussian distribution at any given time. Then I approximate the survival probability by the cumulative distribution function of this OU process at  $V_{th}$  to take advantage of the Gaussian properties. These approximations would overestimate the survival probability

(underestimate the spike probability) by taking the traces that have previously been above  $V_{th}$  and under  $V_{th}$  currently as survived. However, this biasness is reduced if noise level decreases, which implies the approximated boundary equations should be more accurate as noise approaching zero, this is consented by the closeness of approximate and numerical-derived boundaries at low noise levels.

I finally expand the input space from single pulse to nonnegative input, and find a ramp strategy can recover partial isolatability for certain neurons and extend the controllable region. This finding is different with the deterministic case. If we set the initial conditions are  $V_{rest}$  in deterministic, for every controllable pair, we can always use two single pulse inputs to make either neuron spiking first, so expanding the input space from single pulse won't extend the controllable region in deterministic case, however, I found this ramp input can change certain parameter extents from partial isolatable to controllable under noise, especially the neuron pairs in the lower-left quadrant.

I now recap my major results and their most likely impacts. I have provided numerical and analytical results in a simple neural model that establish the feasibility of using a common input to achieve a degree of independent spike control of pairs of neuron in noisy environment. My results are a first step in the development of underactuated control strategies for expanding control of ensemble spiking beyond synchronous activation under noise, and they suggest partial control strategies that may be important in the development of brain-machine interface and smart neuroprosthetics.

## 5.2 Limitations and extensions

This study depends on the LIF model, which as a nearly linear model is analytically tractable but dynamically limited. For example, the model cannot capture biological phenomena, such as bursting [67], post-inhibitory rebound [68], or sodium channel inactivation [52]. However, the model simplicity allows computation of the probability density function for the noisy membrane potential through solving the associated Fokker-Planck equation [57]. Furthermore, for constant input the model is close to an Ornstein-Uhlenbeck (OU) process (ignoring the threshold), with well-known density characteristics, as used in section 2.4.

I did not include synapses in this analysis. Their inclusion would greatly complicate the control problem, but also allow for potentially more elaborate control strategies. For example, if it is desired that two neurons spike in order, and the first has an excitatory synapse onto the second, it may be possible to achieve that sequence with lower input. Conversely, in the presence of inhibitory synapses some non-target spiking might be avoided even when target neurons require higher input. While these are important problems, there are at least two reasons why such ideas may have limited benefit in applications. First, our open loop approach cannot precisely measure the membrane potential, such as in extracellular recording, this would limit the ability of system identification, introducing the synaptic connection would further increase errors in the resulting estimates. Second, electrical synaptic transmission takes place with almost no

delay, and the response in the postsynaptic neuron is in general smaller in amplitude than the source [69]. I did not consider additional limits on the input space that may arise in experimental or clinical applications, for example constraining the maximal input value or input power for safety considerations [50].

The fix-sized strength-duration pool used in numerical analysis leads to conservative bias, that I may claim a controllable neuron pair to be uncontrollable because the required single-pulse inputs are out of range, so the numerical-derived controllable region is only a subset of the “real” one. This biasness can be severe at high or tiny noise levels, at high noises, a small amount of input strength/duration change could alter the survival probability remarkably, so it requires the numerical input domain has sophisticated grid resolution; at tiny noises, the qualified inputs are close to deterministic case, which involves strong impulse  $\delta$  or extremely long duration inputs. So our numerical analysis is based on a small noise level,  $\hat{\sigma} = 0.2$ , at this level, the conservative bias should be reduced comparing to extreme noise levels. Luckily, at tiny noises, we could use the approximate analytic equations to find controllability of neuron pairs, that we have demonstrated above the approximations should be more accurate as noise approaching zero. However, I currently have no trusted method to detect stochastic controllability at high noises. In the next, a dynamic numerical input domain, the size and grid resolution varies with noises, should be perused, in order to improve accuracy while maintaining the numerical efficiency.

The ramping strategy I introduced in Chapter 3 can help recovering controllability for some single-pulse uncontrollable neuron pairs, however, the payoff is the input duration has to be long. Thus in order to include time optimal control, one has to define their desired tradeoff between faster spiking and probability of undesired spikes. Another issue I need to mention is this ramp strategy doesn't consider all possible nonnegative inputs. For example, through deterministic analysis, we know turning off the input can make membrane potentials cross the  $V_{eq}$  boarder (left side of  $V_{eq}$  in Figure 3.1), if a silence period is inserted within a stimulation (this is different from adding a silence period in the end of a input, which is used to represent the “cool down” period between single spike intervals; here, the silence is a part of stimulation design), it may lead to a better strategy to let Neuron  $i$  (with smaller  $\alpha, \beta, \alpha/\beta$ ) spike, I did not pursue it here because I did not explicitly derive the borders of all possible nonnegative inputs in the deterministic phase plane.

In Chapter 4, the mean spike time analysis depends initially on the spike order at  $u = 0$ , which I claim a neuron can spike first if it has smaller mean spike time. However, biophysics are ignored, the mean spike time could be enormously large when  $u = 0$ , see Figure 4.1, it's hard to believe the inter spike interval could be several seconds or minutes *in vivo*. Then the following  $u > 0$  analysis builds upon this  $u = 0$  initial conditions, which leads to speculative controllability conditions. This work implies that the mean exit time concept has to be combined with biophysical constrains to produce meaningful control strategies.

### APPENDIX: Matlab code used in the dissertation

I performed all computational work in Matlab (MathWorks, Inc.) version R2015b.

New\_Fokker\_Spike\_Solver.m. This function uses Crank-Nicolson method to calculate the spike densities given input and parameters of a noisy IAF neuron.

```

1      function [ Spike_Pro, Final_vec ] = New_Fokker_Spike_Solver(
alpha,beta,sigma,Vth,V_,u,dt,initial_vec,Flux_th )
2      % input 'u' has to be constant input!!
3      % 'u' is a vector, not a number, so we know the duration of input
4      %% Define parameters for Crank-Nicolson method
5      dx = 0.01;
6      v = V_:dx:Vth;
7      N = numel(u); %number of time grids
8      J = (Vth-V_)/dx; % number of voltage grids
9      %% Initialize matrix solution f and Spike_Pro
10     f = zeros(J+1,2);
11     f(:,1) = initial_vec;% assign initial vector
12     f(J+1,:) = 0;% absorbing BC
13
14     Spike_Pro = zeros(1,N+1);% P(spike) at time 't', not flux!
15     Spike_Pro_pri = 1 - dx * (sum(f(:,1))-0.5*f(1,1) -0.5*f(end,1));
16     Spike_Pro(1) = min(1,max(0,Spike_Pro_pri));
17
18     %% Calculate Fokker-Planck eq with given control input

```

```

19   % only when 'u' is constant input, A B can be calculated like
    following,
20   % otherwise A B are time dependent
21   A = sparse(J,J);
22   B = sparse(J,J);
23   for j=1:J
24       if j==1
25           A(j,j) = (sigma^2/(2*dx^2)) - (alpha/2) + (1/dt) + ((-
alpha*v_+beta*u(1))/dx) + ((-alpha*v_+beta*u(1))^2/sigma^2);
26           B(j,j) = -(sigma^2/(2*dx^2)) + (alpha/2) + (1/dt) - ((-
alpha*v_+beta*u(1))/dx) - ((-alpha*v_+beta*u(1))^2/sigma^2);
27           A(j,j+1) = -(sigma^2/(2*dx^2));
28           B(j,j+1) = (sigma^2/(2*dx^2));
29       elseif j==J
30           A(j,j) = (sigma^2/(2*dx^2)) - (alpha/2) + (1/dt);
31           B(j,j) = -(sigma^2/(2*dx^2)) + (alpha/2) + (1/dt);
32           A(j,j-1) = -(sigma^2/(4*dx^2)) - (beta*u(1))/(4*dx) +
(alpha*v(1,j))/(4*dx);
33           B(j,j-1) = (sigma^2/(4*dx^2)) + (beta*u(1))/(4*dx) -
(alpha*v(1,j))/(4*dx);
34
35       else
36           A(j,j) = (sigma^2/(2*dx^2)) - (alpha/2) + (1/dt);
37           B(j,j) = -(sigma^2/(2*dx^2)) + (alpha/2) + (1/dt);
38           A(j,j-1) = -(sigma^2/(4*dx^2)) - (beta*u(1))/(4*dx) +
(alpha*v(1,j))/(4*dx);

```

```

39         B(j,j-1) = (sigma^2/(4*dx^2)) + (beta*u(1))/(4*dx) -
(alpha*v(1,j))/(4*dx);
40         A(j,j+1) = -(sigma^2/(4*dx^2)) + (beta*u(1))/(4*dx) -
(alpha*v(1,j))/(4*dx);
41         B(j,j+1) = (sigma^2/(4*dx^2)) - (beta*u(1))/(4*dx) +
(alpha*v(1,j))/(4*dx);
42     end
43 end
44
45 old_col = 1;
46 new_col = 2;
47
48 for i=1:N
49
50     f(1:J,new_col) = A\ (B * f(1:J,old_col));
51
52
53     if i == N
54         Final_vec = f(:,new_col); % save this for future use
55     end
56
57
58     Spike_Pro_pri = 1 - dx * (sum(f(:,new_col))-0.5*f(1,new_col)
-0.5*f(end,new_col));
59     Spike_Pro(i+1) = min(1,max(0,Spike_Pro_pri));
60     f_flx = Spike_Pro(i+1) - Spike_Pro(i);
61

```

```

62     if (u(1) == 0) && (f_flux <= Flux_th) && (f_flux > 0)
63         Spike_Pro(i+2:end) = Spike_Pro(i+1);
64         Final_vec = f(:,new_col);
65         break;
66     end
67
68     old_col = 3 - old_col;
69     new_col = 3 - new_col;
70 end
71
72 end

```

NominalNeuron\_SpkPro\_SD\_Sigmas.m. This function calculates the spike probability of nominal neuron on every grid of the Strength-Duration pool at a certain noise level.

```

1     profile on
2     alpha = 1;%nominal neuron
3     beta = 1;
4     SIGMA = [0.2];
5     Vth = 1;
6     V_ = -4;
7     dx = 0.01;
8     dt = 0.01;
9
10    S_inc = 0.1;
11    D_inc = 0.04;
12

```

```

13  Umin = 0.1;% SD Full Range
14  Umax = 24;
15  Dmin = 0.04;
16  Dmax = 30;
17  Dur_sil_max = 3;
18
19  Input = Umin:S_inc:Umax;
20  Dur = Dmin:D_inc:Dmax;
21
22  n = numel(Input);
23  m = numel(Dur);
24
25  NomNeuron_Psp_SD_ext_Sigmas_02 = zeros(n,m,numel(SIGMA)); %
output mat file
26
27  for k = 1:numel(SIGMA)
28
29      Flux_th = 0;
30      sigma = SIGMA(k);
31
32      dt_sil = dt;
33      if (dx > (sigma^2)/(-alpha*V_)) && (dt_sil > (4*dx^2)/(dx*(-
alpha*V_)-sigma^2))
34          dt_sil = 0.8 * (4*dx^2)/(dx*(-alpha*V_)-sigma^2);
35          dt_sil = round(dt_sil,4);
36      end
37      u_silence = zeros(1,round(Dur_sil_max/dt_sil));

```

```

38
39     for i = 1:n
40         u = Input(i);
41
42         if (dx > (sigma^2)/(-alpha*V_+beta*u)) && (dt >
(4*dx^2)/(dx*(-alpha*V_+beta*u)-sigma^2))
43             dt = 0.8*(4*dx^2)/(dx*(-alpha*V_+beta*u)-sigma^2);
44             dt = round(dt,4);
45         end
46         u = u + zeros(1,round(D_inc/dt)-1);
47
48         for j = 1:m
49             if j == 1
50                 mu = 0;
51                 devi = 0.05;
52                 v = V_:dx:Vth;
53                 INIT = (1/sqrt(2*pi*devi^2))*exp(-(v(1,:)-
mu).^2/(2*devi^2));
54             end
55
56             if NomNeuron_Psp_SD_ext_Sigmas_02(i,j,k) == 0
57                 [ Spike_Pro, Final_vec ] =
New_Fokker_Spike_Solver( alpha,beta,sigma,Vth,V_,u,dt,INIT,Flux_th );
58
59                 INIT = Final_vec;
60

```

```

61             [ Spike_Pro, Final_vec ] =
New_Fokker_Spike_Solver(
alpha,beta,sigma,Vth,V_,u_silence,dt_sil,INIT,Flux_th );
62
63             NomNeuron_Psp_SD_ext_Sigmas_02(i,j,k) =
Spike_Pro(end);
64
65             if Spike_Pro(end) == 1
66                 NomNeuron_Psp_SD_ext_Sigmas_02(i:end,j:end,k)
= 1;
67             end
68
69
save('NomNeuron_Psp_SD_ext_Sigmas_02.mat','NomNeuron_Psp_SD_ext_Sigmas_
02');
70             end
71         end
72         dt = 0.01;
73     end
74 end
75     profile viewer
76

```

Detailed\_SD\_Analysis\_OUTPUT\_SDs.m. This function determines the controllability condition of a noisy IAF neuron pair: fully controllable; partial controllable; totally uncontrollable. If they are fully controllable, it outputs the probabilistic criteria on the SD

**pool.**

```

1      function [ N1,P_N1_NOT_N2,N2,P_N2_NOT_N1] =
Detailed_SD_Analysis_OUTPUT_SDs( alpha2,beta2,sigma,Vth,V_,P_th )
2      %UNTITLED Summary of this function goes here
3      %   Detailed explanation goes here
4      %% parameters setup
5      dx = 0.01;
6      dt = 0.01;
7      alpha_nom = 1;
8      beta_nom = 1;
9
10     S_inc = 0.1;% SD grids
11     D_inc = 0.1;
12
13     Umin = 0.1;% SD Full Range
14     Umax = 12;
15     Dmin = 0.1;
16     Dmax = 15;
17
18     Input = Umin:S_inc:Umax;
19     Dur = Dmin:D_inc:Dmax;
20
21     n = numel(Input);
22     m = numel(Dur);
23
24     SD_P_Sp_N1 = zeros(n,m);%Spike Prob of each pixel on SD space
25     SD_P_Sp_N2 = zeros(n,m);

```

```

26
27     mu = 0;
28     devi = 0.05;
29     v = V_:dx:Vth;
30     INIT_0 = (1/sqrt(2*pi*devi^2))*exp(-(v(1,:)-mu).^2/(2*devi^2));
%inital condition for F-P solver
31
32     %% get SD_P_Sp_N1 AND silence duaration
33     if alpha2 >= alpha_nom
34         % silence duration is 1*(1/alpha_nom)=1
35         Dur_sil_max = 3 * (1/alpha_nom);
36     else
37         % silence duration is 1*(1/alpha2)
38         Dur_sil_max = 3 * (1/alpha2);
39     end
40     %we have stored the nominal neuron spike prob on SD space
41     load('NomNeuron_Psp_SD_Sigmas.mat');
42     % load('NomNeuron_Psp_SD_Sigmas_021025.mat');
43     % load('NomNeuron_Psp_SD_ref_Sigmas_00501.mat');
44     % load('NomNeuron_Psp_SD_ext_Sigmas_02.mat');
45
46     if sigma == 0.2
47         SD_P_Sp_N1 = NomNeuron_Psp_SD_Sigmas(:, :, 1);
48     end
49
50     % if sigma == 0.05
51     %     SD_P_Sp_N1 = NomNeuron_Psp_SD_ref_Sigmas_00501(:, :, 1);

```

```

52 % elseif sigma == 0.1
53 %     SD_P_Sp_N1 = NomNeuron_Psp_SD_ref_Sigmas_00501(:, :, 2);
54 % end
55
56 % if sigma == 0.2
57 %     SD_P_Sp_N1 = NomNeuron_Psp_SD_Sigmas(:, :, 1);
58 % elseif sigma == 0.4
59 %     SD_P_Sp_N1 = NomNeuron_Psp_SD_Sigmas(:, :, 2);
60 % elseif sigma == 0.6
61 %     SD_P_Sp_N1 = NomNeuron_Psp_SD_Sigmas(:, :, 3);
62 % elseif sigma == 0.8
63 %     SD_P_Sp_N1 = NomNeuron_Psp_SD_Sigmas(:, :, 4);
64 % elseif sigma == 1
65 %     SD_P_Sp_N1 = NomNeuron_Psp_SD_Sigmas(:, :, 5);
66 % end
67
68 %% then get SD_P_Sp_N2
69
70 % get Flx threshold to terminate silence simulation earlier
71 sigma_min = 0.2;
72
73 [ Flux_th ] = Flx_Threshold_Calculator( alpha_nom, sigma_min, Vth
);
74 % get u_silence, dt_silence in front
75 dt_sil = dt;
76 if (dx > (sigma^2)/(-alpha2*V_) && (dt_sil > (4*dx^2)/(dx*(-
alpha2*V_)-sigma^2))

```

```

77     dt_sil = 0.8 * (4*dx^2)/(dx*(-alpha2*V_)-sigma^2);
78     dt_sil = round(dt_sil,4);
79 end
80 u_silence = zeros(1,round(Dur_sil_max/dt_sil));
81
82 for i = 1:n
83     u = Input(i);
84     if (dx > (sigma^2)/(-alpha2*V_+beta2*u)) && (dt >
(4*dx^2)/(dx*(-alpha2*V_+beta2*u)-sigma^2))
85         dt = 0.8*(4*dx^2)/(dx*(-alpha2*V_+beta2*u)-sigma^2);
86         dt = round(dt,4);
87     end
88     u = u + zeros(1,round(D_inc/dt)-1);
89
90     for j = 1:m
91         if j == 1
92             INIT = INIT_0;
93         end
94
95         if SD_P_Sp_N2(i,j) == 0
96             [ Spike_Pro, Final_vec ] = New_Fokker_Spike_Solver(
alpha2,beta2,sigma,Vth,V_,u,dt,INIT,Flux_th );
97
98             INIT = Final_vec;
99
100            [ Spike_Pro, Final_vec ] = New_Fokker_Spike_Solver(
alpha2,beta2,sigma,Vth,V_,u_silence,dt_sil,INIT,Flux_th);

```

```
101
102         SD_P_Sp_N2(i,j) = Spike_Pro(end);
103
104         if SD_P_Sp_N2(i,j) == 1
105             SD_P_Sp_N2(i:end,j:end) = 1;
106         end
107     end
108 end
109 dt = 0.01;
110 end
111 %% determine sequence controllability
112 P_N1_NOT_N2 = SD_P_Sp_N1 .* (1-SD_P_Sp_N2); % Neuron 1 is target
113 if sum(sum(P_N1_NOT_N2 >= P_th)) > 0
114     N1 = 1;
115 else
116     N1 = 0;
117 end
118
119 P_N2_NOT_N1 = SD_P_Sp_N2 .* (1-SD_P_Sp_N1); % Neuron 2 is target
120 if sum(sum(P_N2_NOT_N1 >= P_th)) > 0
121     N2 = 1;
122 else
123     N2 = 0;
124 end
125
126
127 end
```

SinglePulseControllabilityMap\_SIGMAS.m. This function derives the single pulse stochastic controllability map at certain noise level.

```

1
2   % SIGMA = [0.2, 0.4];
3   SIGMA = [0.2];
4   Vth = 1;
5   V_ = -4;
6   P_th = 0.95;
7
8   alpha2_min = 0.2;
9   alpha2_max = 5;
10  beta2_min = 0.2;
11  beta2_max = 5;
12  da2 = 0.1;
13  db2 = 0.1;
14  alpha2 = alpha2_min:da2:alpha2_max;
15  beta2 = beta2_min:db2:beta2_max;
16
17  n = numel(alpha2);
18  m = numel(beta2);
19
20  Seq_Contr_Maps_Sigmas_02_Pth_095 = zeros(m,n,numel(SIGMA));
21  P_N1_NOT_N2_SD_Sigmas_02_Pth_095 = cell(m,n,numel(SIGMA));
22  P_N2_NOT_N1_SD_Sigmas_02_Pth_095 = cell(m,n,numel(SIGMA));
23  % load('Seq_Contr_Maps_Sigmas.mat');
24  % load('P_N1_NOT_N2_SD_Sigmas.mat');
25  % load('P_N2_NOT_N1_SD_Sigmas.mat');

```

```

26
27   for k = 1:numel(SIGMA)
28
29       sigma = SIGMA(k);
30
31       for i = 1:n
32
33           for j = 1:m
34
35               if Seq_Contr_Maps_Sigmas_02_Pth_095(j,i,k) == 0
36
37                   [ N1,P_N1_NOT_N2,N2,P_N2_NOT_N1] =
Detailed_SD_Analysis_OUTPUT_SDs( alpha2(i),beta2(j),sigma,Vth,V_,P_th
);
38
39                   P_N1_NOT_N2_SD_Sigmas_02_Pth_095{j,i,k} =
P_N1_NOT_N2;
40                   P_N2_NOT_N1_SD_Sigmas_02_Pth_095{j,i,k} =
P_N2_NOT_N1;
41
42                   if (N1 == 1) && (N2 == 1)
43                       Seq_Contr_Maps_Sigmas_02_Pth_095(j,i,k) = 1;
44                   elseif (N1 == 0) && (N2 == 0)
45                       Seq_Contr_Maps_Sigmas_02_Pth_095(j,i,k) = 4;
46                   elseif (N1 == 1) && (N2 == 0)
47                       Seq_Contr_Maps_Sigmas_02_Pth_095(1:j,i:end,k)
= 2;

```

```
48             elseif (N1 == 0) && (N2 == 1)
49                 Seq_Contr_Maps_Sigmas_02_Pth_095(j:end,1:i,k)
= 3;
50             end
51
52         end
53
54
55     save('Seq_Contr_Maps_Sigmas_02_Pth_095.mat','Seq_Contr_Maps_Sigmas_02_P
th_095');
56
57     save('P_N1_NOT_N2_SD_Sigmas_02_Pth_095.mat','P_N1_NOT_N2_SD_Sigmas_02_P
th_095');
58
59     save('P_N2_NOT_N1_SD_Sigmas_02_Pth_095.mat','P_N2_NOT_N1_SD_Sigmas_02_P
th_095');
60
61         disp('i = ');
62         disp(i);
63         disp('j = ');
64         disp(j);
65         disp('k = ');
66         disp(k);
67     end
68 end
69 end
```

SeqControlMap\_Visualize.m. This function plots the single pulse controllability map as scatters.

```
1     alpha1 = 1;
2     beta1 = 1;
3     sigma = 0.3;
4     Vth = 1;
5     V_ = -4;
6     P_th = 0.9;
7
8     alpha2_min = 0.2;
9     alpha2_max = 5;
10    beta2_min = 0.2;
11    beta2_max = 5;
12    da2 = 0.1;
13    db2 = 0.1;
14    alpha2 = alpha2_min:da2:alpha2_max;
15    beta2 = beta2_min:db2:beta2_max;
16
17    n = numel(alpha2);
18    m = numel(beta2);
19
20    % load('Seq_Contr_Maps_Sigmas_0204.mat');
21    % load('Seq_Contr_Maps_Sigmas_00501_SD_Ref.mat');
22    % load('Seq_Contr_Maps_Sigmas_02_SD_Ext.mat');
23    % load('Seq_Contr_Maps_Sigmas_02_Pth_085.mat');
24    load('Seq_Contr_Maps_Sigmas_02_Pth_095.mat');
25    Seq_Contr_sigma_03_PS = Seq_Contr_Maps_Sigmas_02_Pth_095(:, :, 1);
```

```
26
27   for i = 1:m
28       for j = 1:n
29           if Seq_Contr_sigma_03_PS (i,j) == 1
30               b2 = i*db2+0.1;
31               a2 = j*da2+0.1;
32               scatter(b2,a2,40,'b','filled');
33               hold on;
34           elseif Seq_Contr_sigma_03_PS (i,j) == 2
35               b2 = i*db2+0.1;
36               a2 = j*da2+0.1;
37               scatter(b2,a2,40,'r','filled');
38               hold on;
39           elseif Seq_Contr_sigma_03_PS (i,j) == 3
40               b2 = i*db2+0.1;
41               a2 = j*da2+0.1;
42               scatter(b2,a2,40,'y','filled');
43               hold on;
44           elseif Seq_Contr_sigma_03_PS (i,j) == 4
45               b2 = i*db2+0.1;
46               a2 = j*da2+0.1;
47               scatter(b2,a2,40,'g','filled');
48               hold on;
49           end
50       end
51   end
52
```

```

53     axis([0 5 0 5]);
54     hold on;
55     scatter(1,1,250,'k','filled');
56     hold on;
57     plot([0 5],[1 1],'k',[1 1],[0 5],'k',[0 5],[0
58     5],'k','LineWidth',4);
59     ylabel('\alpha_{2}/\alpha_{1}','FontSize',20,'FontWeight','bold');
60     xlabel('\beta_{2}/\beta_{1}','FontSize',20,'FontWeight','bold');
61     % title('\sigma = 0.05','FontSize',20,'FontWeight','bold');
62     set(gca,'FontSize',20);
63     set(gca,'FontWeight','bold');

```

SeqControllability\_BCs\_Curves.m. This function gets the stochastic controllability boundaries from numerical controllability map and plots the approximated analytic boundaries.

```

1
2
3     alpha2_min = 0.02;
4     alpha2_max = 1;
5     beta2_min = 0.02;
6     beta2_max = 1;
7     da2 = 0.02;
8     db2 = 0.02;
9     alpha2 = alpha2_min:da2:alpha2_max;
10    beta2 = beta2_min:db2:beta2_max;

```

```

11
12     n = numel(alpha2);
13     m = numel(beta2);
14
15     % load('Seq_Contr_Maps_Sigmas_0204_ref.mat');
16     % SeqContrMap_Sigma_02_ref =
Seq_Contr_Maps_Sigmas_0204_ref(:,:,1);
17     % SeqContrMap_Sigma_02_ref = SeqContrMap_Sigma_02_ref'; %alpha2
is row, beta2 is column
18
19     load('Seq_Contr_Maps_Sigmas_021025_ref.mat');
20     SeqContrMap_Sigma_022_ref =
Seq_Contr_Maps_Sigmas_021025_ref(:,:,2);
21     SeqContrMap_Sigma_022_ref = SeqContrMap_Sigma_022_ref';
22
23     % RHS_alpha2_02 = []; %y-axis
24     % RHS_beta2_02 = []; %x-axis
25     % LHS_alpha2_02 = [];
26     % LHS_beta2_02 = [];
27
28     RHS_alpha2_022 = []; %y-axis
29     RHS_beta2_022 = []; %x-axis
30     LHS_alpha2_022 = [];
31     LHS_beta2_022 = [];
32
33     % for i = 1:n
34     %         for j = 1:m

```

```

35     %           if (SeqContrMap_Sigma_02_ref(i,1) == 3) ||
(SeqContrMap_Sigma_02_ref(i,1) == 4)
36     %           break;
37     %           end
38     %
39     %           if ((SeqContrMap_Sigma_02_ref(i,j) == 1) ||
(SeqContrMap_Sigma_02_ref(i,j) == 2)) &&
((SeqContrMap_Sigma_02_ref(i,j+1) == 3) ||
(SeqContrMap_Sigma_02_ref(i,j+1) == 4))
40     %           RHS_alpha2_02 = [RHS_alpha2_02, alpha2(i)];
41     %           RHS_beta2_02 = [RHS_beta2_02, beta2(j)];
42     %           break;
43     %           end
44     %       end
45     % end
46
47     % for i = 1:n
48     %     for j = 1:m
49     %         if (SeqContrMap_Sigma_02_ref(i,m) == 2) ||
(SeqContrMap_Sigma_02_ref(i,m) == 4)
50     %             break;
51     %         end
52     %
53     %         if ((SeqContrMap_Sigma_02_ref(i,j) == 1) ||
(SeqContrMap_Sigma_02_ref(i,j) == 3)) &&
((SeqContrMap_Sigma_02_ref(i,j-1) == 2) ||
(SeqContrMap_Sigma_02_ref(i,j-1) == 4))

```

```

54     %             LHS_alpha2_02 = [LHS_alpha2_02, alpha2(i)];
55     %             LHS_beta2_02 = [LHS_beta2_02, beta2(j)];
56     %             break;
57     %         end
58     %     end
59     % end
60
61
62
63     for i = 7:n
64         for j = 1:m
65             if (SeqContrMap_Sigma_022_ref(i,1) == 3) ||
66 (SeqContrMap_Sigma_022_ref(i,1) == 4)
67                 break;
68             end
69             if ((SeqContrMap_Sigma_022_ref(i,j) == 1) ||
70 (SeqContrMap_Sigma_022_ref(i,j) == 2)) &&
71 ((SeqContrMap_Sigma_022_ref(i,j+1) == 3) ||
72 (SeqContrMap_Sigma_022_ref(i,j+1) == 4))
73                 RHS_alpha2_022 = [RHS_alpha2_022, alpha2(i)];
74                 RHS_beta2_022 = [RHS_beta2_022, beta2(j)];
75                 break;
76             end
77         end
78     end
79     end
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

```
77     for i = 1:n
78         for j = 1:m
79             if (SeqContrMap_Sigma_022_ref(i,m) == 2) ||
(SeqContrMap_Sigma_022_ref(i,m) == 4)
80                 break;
81             end
82
83             if ((SeqContrMap_Sigma_022_ref(i,j) == 1) ||
(SeqContrMap_Sigma_022_ref(i,j) == 3)) &&
((SeqContrMap_Sigma_022_ref(i,j-1) == 2) ||
(SeqContrMap_Sigma_022_ref(i,j-1) == 4))
84                 LHS_alpha2_022 = [LHS_alpha2_022, alpha2(i)];
85                 LHS_beta2_022 = [LHS_beta2_022, beta2(j)];
86                 break;
87             end
88         end
89     end
90
91     % RHS_alpha2_022_1 = RHS_alpha2_022;
92     % RHS_beta2_022_1 = RHS_beta2_022;
93     % LHS_alpha2_022_1 = LHS_alpha2_022;
94     % LHS_beta2_022_1 = LHS_beta2_022;
95
96     alpha2_min = 0.2;
97     alpha2_max = 5;
98     beta2_min = 0.2;
99     beta2_max = 5;
```

```

100 da2 = 0.1;
101 db2 = 0.1;
102 alpha2 = alpha2_min:da2:alpha2_max;
103 beta2 = beta2_min:db2:beta2_max;
104
105 n = numel(alpha2);
106 m = numel(beta2);
107
108 load('Seq_Contr_Maps_Sigmas_022.mat');
109 SeqContrMap_Sigma_022 = Seq_Contr_Maps_Sigmas_022;
110 SeqContrMap_Sigma_022 = SeqContrMap_Sigma_022';
111
112 for i = 1:n
113     for j = 1:m
114         if (SeqContrMap_Sigma_022(i,1) == 3) ||
115         (SeqContrMap_Sigma_022(i,1) == 4)
116             break;
117         end
118         if ((SeqContrMap_Sigma_022(i,j) == 1) ||
119         (SeqContrMap_Sigma_022(i,j) == 2)) && ((SeqContrMap_Sigma_022(i,j+1) ==
120         3) || (SeqContrMap_Sigma_022(i,j+1) == 4)) && (alpha2(i) > 1)
121             if (alpha2(i) == 1.1) || (alpha2(i) == 1.2)
122                 RHS_alpha2_022 = [RHS_alpha2_022, alpha2(i)];
123                 RHS_beta2_022 = [RHS_beta2_022, 0.76];
124             else
125                 RHS_alpha2_022 = [RHS_alpha2_022, alpha2(i)];

```

```
124             RHS_beta2_022 = [RHS_beta2_022, beta2(j)];
125         end
126         break;
127     end
128 end
129 end
130
131 for i = 1:n
132     for j = 1:m
133         if (SeqContrMap_Sigma_022(i,m) == 2) ||
134            (SeqContrMap_Sigma_022(i,m) == 4)
135             break;
136         end
137         if ((SeqContrMap_Sigma_022(i,j) == 1) ||
138            (SeqContrMap_Sigma_022(i,j) == 3)) && ((SeqContrMap_Sigma_022(i,j-1) ==
139            2) || (SeqContrMap_Sigma_022(i,j-1) == 4)) && (beta2(j) > 1)
140             LHS_alpha2_022 = [LHS_alpha2_022, alpha2(i)];
141             LHS_beta2_022 = [LHS_beta2_022, beta2(j)];
142             break;
143         end
144     end
145 end
146
147 save('RHS_alpha2_022.mat','RHS_alpha2_022');
148 save('RHS_beta2_022.mat','RHS_beta2_022');
149 save('LHS_alpha2_022.mat','LHS_alpha2_022');
```

```
148 save('LHS_beta2_022.mat','LHS_beta2_022');
149
150 RHS_beta2_022_1 = [];
151 RHS_beta2_022_2 = [];
152 RHS_alpha2_022_1 = [];
153 RHS_alpha2_022_2 = [];
154
155 LHS_beta2_022_1 = [];
156 LHS_beta2_022_2 = [];
157 LHS_alpha2_022_1 = [];
158 LHS_alpha2_022_2 = [];
159
160 for i = 1:numel(RHS_alpha2_022)
161     if RHS_alpha2_022(i) <= 1.2
162         RHS_beta2_022_1 = [RHS_beta2_022_1, RHS_beta2_022(i)];
163         RHS_alpha2_022_1 = [RHS_alpha2_022_1, RHS_alpha2_022(i)];
164     else
165         RHS_beta2_022_2 = [RHS_beta2_022_2, RHS_beta2_022(i)];
166         RHS_alpha2_022_2 = [RHS_alpha2_022_2, RHS_alpha2_022(i)];
167     end
168 end
169
170 for i = 1:numel(LHS_alpha2_022)
171     if LHS_alpha2_022(i) <= 0.8
172         LHS_beta2_022_1 = [LHS_beta2_022_1, LHS_beta2_022(i)];
173         LHS_alpha2_022_1 = [LHS_alpha2_022_1, LHS_alpha2_022(i)];
174     else
```

```
175         LHS_beta2_022_2 = [LHS_beta2_022_2, LHS_beta2_022(i)];
176         LHS_alpha2_022_2 = [LHS_alpha2_022_2, LHS_alpha2_022(i)];
177     end
178 end
179
180 load census;
181 % f_RHS_02 = fit(RHS_beta2_02',RHS_alpha2_02','exp2');
182 % f_LHS_02 = fit(LHS_beta2_02',LHS_alpha2_02','poly1');
183
184 f_RHS_022_1 = fit(RHS_beta2_022_1',RHS_alpha2_022_1','exp2');
185 f_RHS_022_2 = fit(RHS_beta2_022_2',RHS_alpha2_022_2','poly1');
186 f_LHS_022_1 = fit(LHS_beta2_022_1',LHS_alpha2_022_1','poly1');
187 f_LHS_022_2 = fit(LHS_beta2_022_2',LHS_alpha2_022_2','poly1');
188
189 %
plot(RHS_beta2_02,RHS_alpha2_02,'r',LHS_beta2_02,LHS_alpha2_02,'y','Lin
eWidth',4);
190 % hold on;
191 % plot(f_RHS_02,'c');
192 % hold on;
193 % plot(f_LHS_02,'k');
194 % hold on;
195
plot(RHS_beta2_022,RHS_alpha2_022,'r',LHS_beta2_022,LHS_alpha2_022,'y',
'LineWidth',4);
196 hold on;
197 plot(f_RHS_022_1,'c');
```

```

198 hold on;
199 plot(f_RHS_022_2,'c');
200 hold on;
201 plot(f_LHS_022_1,'k');
202 hold on;
203 plot(f_LHS_022_2,'k');
204 hold on;
205 axis([0 5 0 5]);
206 scatter(1,1,250,'k','filled');
207 hold on;
208 % plot([0 5],[1 1],'k',[1 1],[0 5],'k',[0 5],[0
5],'k','LineWidth',2);
209
ylabel('\alpha_{2}/\alpha_{1}','FontSize',20,'FontWeight','bold');
210 xlabel('\beta_{2}/\beta_{1}','FontSize',20,'FontWeight','bold');
211 title('$$\hat{\sigma}$$ =
0.22','Interpreter','Latex','FontSize',20,'FontWeight','bold');
212 set(gca,'FontSize',20);
213 set(gca,'FontWeight','bold');

```

**New\_Euler\_Maruyama\_Sample.m.** This function applies Euler-Maruyama method to get membrane potential sample traces of noisy IAF neuron.

```

1 function [ N1_Sp_fir, N2_Sp_fir, Spike_Time, V1_Trace, V2_Trace ]
= New_Euler_Maruyama_Sample( alpha1, beta1, alpha2, beta2, sigma, Vth,
Pth, u, dt)
2 %UNTITLED Summary of this function goes here
3 % Detailed explanation goes here

```

```
4     N1_Sp_fir = 0;
5     N2_Sp_fir = 0;
6
7     N = numel(u);
8     Spike_Time = zeros(1,N+1);
9     V1_Trace = zeros(1,N+1);
10    V2_Trace = zeros(1,N+1);
11
12    v0 = 0;
13    V1_Trace(1) = normrnd(0,0.5 * sigma/sqrt(2 * alpha1));
14    V2_Trace(1) = normrnd(0,0.5 * sigma/sqrt(2 * alpha2));
15
16    if V1_Trace(1) > Vth
17        V1_Trace(1) = Vth;
18    elseif V2_Trace(1) > Vth
19        V2_Trace(1) = Vth;
20    end
21
22    for i = 1:N
23
24        if (V1_Trace(i+1) == Vth) && (V2_Trace(i+1) < Vth)
25            del_W2 = normrnd(0,sqrt(dt));
26            V2_Trace(i+1) = V2_Trace(i) + (-alpha2*V2_Trace(i) +
beta2*u(i))*dt + sigma*del_W2;
27        elseif (V1_Trace(i+1) < Vth) && (V2_Trace(i+1) == Vth)
28            del_W1 = normrnd(0,sqrt(dt));
```

```

29         V1_Trace(i+1) = V1_Trace(i) + (-alpha1*V1_Trace(i) +
beta1*u(i))*dt + sigma*del_W1;
30         elseif (V1_Trace(i+1) < Vth) && (V2_Trace(i+1) < Vth)
31             del_W1 = normrnd(0,sqrt(dt));
32             V1_Trace(i+1) = V1_Trace(i) + (-alpha1*V1_Trace(i) +
beta1*u(i))*dt + sigma*del_W1;
33             del_W2 = normrnd(0,sqrt(dt));
34             V2_Trace(i+1) = V2_Trace(i) + (-alpha2*V2_Trace(i) +
beta2*u(i))*dt + sigma*del_W2;
35         elseif (V1_Trace(i+1) == Vth) && (V2_Trace(i+1) == Vth)
36             continue
37     end
38
39     if (V1_Trace(i) == Vth) && (V2_Trace(i) < Vth)
40         V1_Trace(i+1) = v0;
41     elseif (V1_Trace(i) < Vth) && (V2_Trace(i) == Vth)
42         V2_Trace(i+1) = v0;
43     elseif (V1_Trace(i) == Vth) && (V2_Trace(i) == Vth)
44         V1_Trace(i+1) = v0;
45         V2_Trace(i+1) = v0;
46     end
47
48     if (V1_Trace(i+1) >= Vth) && (V2_Trace(i+1) < Vth)
49         V1_Trace(i+1) = Vth;
50         Spike_Time(i+1) = 1;
51         continue
52     elseif (V1_Trace(i+1) < Vth) && (V2_Trace(i+1) >= Vth)

```

```

53         V2_Trace(i+1) = Vth;
54         Spike_Time(i+1) = 2;
55         continue
56     elseif (V1_Trace(i+1) >= Vth) && (V2_Trace(i+1) >= Vth)
57         V1_Trace(i+1) = Vth;
58         V2_Trace(i+1) = Vth;
59         Spike_Time(i+1) = 3;
60         continue
61     elseif (V1_Trace(i+1) < Vth) && (V2_Trace(i+1) < Vth)
62         if i == N
63             break
64         end
65         MU1 = V1_Trace(i+1) + (-alpha1*V1_Trace(i+1) +
beta1*u(i+1))*dt;
66         MU2 = V2_Trace(i+1) + (-alpha2*V2_Trace(i+1) +
beta2*u(i+1))*dt;
67         StdDevi = sigma * sqrt(dt);
68         P1 = 1 - normcdf(Vth,MU1,StdDevi);
69         P2 = 1 - normcdf(Vth,MU2,StdDevi);
70         if (P1 >= Pth) && (P2 < Pth)
71             V1_Trace(i+2) = Vth;
72             Spike_Time(i+2) = 1;
73             continue
74         elseif (P1 < Pth) && (P2 >= Pth)
75             V2_Trace(i+2) = Vth;
76             Spike_Time(i+2) = 2;
77             continue

```

```
78         elseif (P1 >= Pth) && (P2 >= Pth)
79             V1_Trace(i+2) = Vth;
80             V2_Trace(i+2) = Vth;
81             Spike_Time(i+2) = 3;
82             continue
83         end
84     end
85 end
86
87 for i = 1:N+1
88     if Spike_Time(i) == 1
89         N1_Sp_fir = 1;
90         N2_Sp_fir = 0;
91         break
92     elseif Spike_Time(i) == 2
93         N1_Sp_fir = 0;
94         N2_Sp_fir = 1;
95         break
96     elseif Spike_Time(i) == 3
97         N1_Sp_fir = 1;
98         N2_Sp_fir = 1;
99         break
100     end
101 end
102
103 end
```

New\_SampleTrace\_Pra1.m. This function shows sample traces for 4 neuron pairs, one is fully controllable, two are partial controllable, one is totally uncontrollable.

```

1   alpha1 = 0.04;
2   beta1 = 11/270;
3   alpha2 = 1.1 * alpha1;
4   beta2 = 1.1 * beta1;
5   % epsilon = 0.4*((alpha2/beta2)/(alpha1/beta1) - 1);
6   sigma = 0.04;
7   Vth = 1;
8   V_ = -4;
9   Pth = 0.9;
10
11  % [ u,dt ] = New_Input_Generator(
alpha1,beta1,alpha2,beta2,epsilon,sigma,Vth,V_ );
12  % [ u,dt ] = Input_Generator( alpha2,beta2,epsilon,sigma,Vth,V_
);
13  % old_u = u;
14  % old_dt = dt;
15  % u = old_dt * trapz(old_u)/(old_dt * numel(old_u));%constant
input
16  % u = u + zeros(1,numel(old_u));
17  % dt = 0.1;
18  % % u = interp1(0:old_dt:old_dt * (numel(old_u)-
1),old_u,0:dt:old_dt * (numel(old_u)-1),'spline');
19  % % u = interp1(0:old_dt:old_dt * (numel(old_u)-1),u,0:dt:old_dt
* (numel(old_u)-1),'linear');
20  dt = 0.01;

```

```

21  T_sil = 1 / alpha1;
22  % T = 37.5;
23  % u = 1.57 + zeros(1,T/dt);
24  % load('u1_ramp_sigma_03.mat');
25  D = 0.1 / alpha1;
26  u = 6.6 * alpha1/beta1 + zeros(1,D/dt);
27  u = [u,zeros(1,T_sil/dt)];
28  T = numel(u) * dt;
29
30  figure
31  plot(0:dt:(numel(u)-1)*dt, u,'b','LineWidth',2);
32  ax = gca;
33  ax.FontSize = 20;
34  xlabel('Time(ms)', 'FontSize',20);
35  ylabel('Input', 'FontSize',20);
36  xlim([0,numel(u)*dt]);
37
38  [ N1_Sp, N2_Sp, Spike_Time, V1_Trace, V2_Trace ] =
New_Euler_Maruyama_Sample( alpha1, beta1, alpha2, beta2, sigma, Vth,
Pth, u, dt);
39  figure
40  plot(0:dt:T,V1_Trace,'k',0:dt:T,V2_Trace,'r','LineWidth',2);
41  legend('V1','V2','FontSize',20);
42  xlim([0,T]);
43  ylim([-Vth,Vth+0.5]);
44  hold on
45  plot([0 T],[Vth Vth],'k--','LineWidth',2);

```

```

46     hold on
47     for i = 1:T/dt+1
48         if Spike_Time(i) == 1
49             scatter((i-1)*dt, Vth+0.1, 80, 'k','filled');
50             hold on
51         elseif Spike_Time(i) == 2
52             scatter((i-1)*dt, Vth+0.1, 80, 'r','filled');
53             hold on
54         elseif Spike_Time(i) == 3
55             scatter((i-1)*dt, Vth+0.1, 80, 'k','filled');
56             hold on
57             scatter((i-1)*dt, Vth+0.2, 80, 'r','filled');
58             hold on
59         end
60     end
61     ax = gca;
62     ax.FontSize = 20;
63     xlabel('Time (ms)', 'FontSize', 20);
64     ylabel('Membrane Potential', 'FontSize', 20);

```

**Singel\_Pulse\_SDE\_Simulation\_Statistics.m.** This function does statistics of spike outcomes for every fully controllable neuron pairs in a stochastic controllability map.

```

1     sigma = 0.2;
2     Vth = 1;
3     V_ = -4;
4     Pth = 0.9;
5     alpha1 = 1;

```

```
6     beta1 = 1;
7     Iter_Num = 100;
8     Silence_Duration_max = 15;
9     dt = 0.01;
10
11    alpha2_min = 0.02;
12    alpha2_max = 1;
13    beta2_min = 0.02;
14    beta2_max = 1;
15    da2 = 0.02;
16    db2 = 0.02;
17    alpha2 = alpha2_min:da2:alpha2_max;
18    beta2 = beta2_min:db2:beta2_max;
19
20    n = numel(alpha2);
21    m = numel(beta2);
22
23    load('Seq_Contr_Maps_Sigmas_0204_ref.mat');
24
25    Num_NeuPairs = 0;
26
27    for i = 1:n
28        for j = 1:m
29            if (Seq_Contr_Maps_Sigmas_0204_ref(j,i,1) == 1)
30                Num_NeuPairs = Num_NeuPairs + 1;
31            end
32        end
33    end
```

```
33     end
34
35     V1_S1_D1_PS = cell(1,Num_NeuPairs);
36     V2_S1_D1_PS = cell(1,Num_NeuPairs);
37     Spk_Time_S1_D1_PS = cell(1,Num_NeuPairs);
38     V1_S2_D2_PS = cell(1,Num_NeuPairs);
39     V2_S2_D2_PS = cell(1,Num_NeuPairs);
40     Spk_Time_S2_D2_PS = cell(1,Num_NeuPairs);
41     ALPHA2_PS = cell(1,Num_NeuPairs);
42     BETA2_PS = cell(1,Num_NeuPairs);
43
44     N1_Sp_N2_sigma_02_PS = zeros(m,n);
45     N1_Sp_N1_0_sigma_02_PS = zeros(m,n);
46     N1_Sp_N1_1_sigma_02_PS = zeros(m,n);
47     N1_Sp_N1_2_sigma_02_PS = zeros(m,n);
48     N1_Sp_N1_3_sigma_02_PS = zeros(m,n);% N1 is spike neuron, N1
fires more than twice
49     N2_Sp_N1_sigma_02_PS = zeros(m,n);
50     N2_Sp_N2_0_sigma_02_PS = zeros(m,n);
51     N2_Sp_N2_1_sigma_02_PS = zeros(m,n);
52     N2_Sp_N2_2_sigma_02_PS = zeros(m,n);
53     N2_Sp_N2_3_sigma_02_PS = zeros(m,n);
54
55     Indx_NeuPairs = 0;
56
57     load('P_N1_NOT_N2_SD_Sigmas_0204_ref.mat');
58     load('P_N2_NOT_N1_SD_Sigmas_0204_ref.mat');
```

```

59
60   for i = 1:m
61       for j = 1:n
62           if (Seq_Contr_Maps_Sigmas_0204_ref(i,j,1) == 1)
63
64               P_N1_NOT_N2 = P_N1_NOT_N2_SD_Sigmas_0204_ref{i,j,1};
65               [M,I] = max(P_N1_NOT_N2(:));
66               [I_row, I_col] = ind2sub(size(P_N1_NOT_N2),I);
67
68               S1 = 0.1 * I_row;
69               D1 = 0.1 * I_col;
70
71               P_N2_NOT_N1 = P_N2_NOT_N1_SD_Sigmas_0204_ref{i,j,1};
72               [M,I] = max(P_N2_NOT_N1(:));
73               [I_row, I_col] = ind2sub(size(P_N2_NOT_N1),I);
74
75               S2 = 0.1 * I_row;
76               D2 = 0.1 * I_col;
77
78               u1 = S1 + zeros(1,round(D1/dt));
79               u2 = S2 + zeros(1,round(D2/dt));
80
81               Silence_Duration = 3 / alpha2(j);
82               Silence_Duration =
min(Silence_Duration,Silence_Duration_max);
83
84               u1 = [u1, zeros(1,round(Silence_Duration/dt))];

```

```

85         u2 = [u2, zeros(1,round(Silence_Duration/dt))];
86
87         v1_s1_d1 = zeros(Iter_Num, numel(u1)+1);
88         v2_s1_d1 = zeros(Iter_Num, numel(u1)+1);
89         sp_time_s1_d1 = zeros(Iter_Num, numel(u1)+1);
90
91         v1_s2_d2 = zeros(Iter_Num, numel(u2)+1);
92         v2_s2_d2 = zeros(Iter_Num, numel(u2)+1);
93         sp_time_s2_d2 = zeros(Iter_Num, numel(u2)+1);
94
95         Indx_NeuPairs = Indx_NeuPairs + 1;
96
97         ALPHA2_PS{1,Indx_NeuPairs} = alpha2(j);
98         BETA2_PS{1,Indx_NeuPairs} = beta2(i);
99
100        for k = 1:Iter_Num
101            [ N1_Sp, N2_Sp, Spike_Time, V1_Trace, V2_Trace ]
= New_Euler_Maruyama_Sample( alpha1, beta1, alpha2(j), beta2(i), sigma,
Vth, Pth, u1, dt);
102
103            v1_s1_d1(k,:) = V1_Trace;
104            v2_s1_d1(k,:) = V2_Trace;
105            sp_time_s1_d1(k,:) = Spike_Time;
106
107            if (sum(Spike_Time == 2) > 0) || (sum(Spike_Time
== 3) > 0)

```

```

108             N1_Sp_N2_sigma_02_PS(i,j) =
N1_Sp_N2_sigma_02_PS(i,j) + 1;
109             elseif sum(Spike_Time == 1) == 0
110                 N1_Sp_N1_0_sigma_02_PS(i,j) =
N1_Sp_N1_0_sigma_02_PS(i,j) + 1;
111             elseif sum(Spike_Time == 1) == 1
112                 N1_Sp_N1_1_sigma_02_PS(i,j) =
N1_Sp_N1_1_sigma_02_PS(i,j) + 1;
113             elseif sum(Spike_Time == 1) == 2
114                 N1_Sp_N1_2_sigma_02_PS(i,j) =
N1_Sp_N1_2_sigma_02_PS(i,j) + 1;
115             elseif sum(Spike_Time == 1) > 2
116                 N1_Sp_N1_3_sigma_02_PS(i,j) =
N1_Sp_N1_3_sigma_02_PS(i,j) + 1;
117             end
118
119             [ N1_Sp, N2_Sp, Spike_Time, V1_Trace, V2_Trace ]
= New_Euler_Maruyama_Sample( alpha1, beta1, alpha2(j), beta2(i), sigma,
Vth, Pth, u2, dt);
120
121             v1_s2_d2(k,:) = V1_Trace;
122             v2_s2_d2(k,:) = V2_Trace;
123             sp_time_s2_d2(k,:) = Spike_Time;
124
125             if (sum(Spike_Time == 1) > 0) || (sum(Spike_Time
== 3) > 0)

```

```
126             N2_Sp_N1_sigma_02_PS(i,j) =
N2_Sp_N1_sigma_02_PS(i,j) + 1;
127             elseif sum(Spike_Time == 2) == 0
128                 N2_Sp_N2_0_sigma_02_PS(i,j) =
N2_Sp_N2_0_sigma_02_PS(i,j) + 1;
129             elseif sum(Spike_Time == 2) == 1
130                 N2_Sp_N2_1_sigma_02_PS(i,j) =
N2_Sp_N2_1_sigma_02_PS(i,j) + 1;
131             elseif sum(Spike_Time == 2) == 2
132                 N2_Sp_N2_2_sigma_02_PS(i,j) =
N2_Sp_N2_2_sigma_02_PS(i,j) + 1;
133             elseif sum(Spike_Time == 2) > 2
134                 N2_Sp_N2_3_sigma_02_PS(i,j) =
N2_Sp_N2_3_sigma_02_PS(i,j) + 1;
135             end
136
137
save('N1_Sp_N2_sigma_02_PS.mat','N1_Sp_N2_sigma_02_PS');
138
save('N1_Sp_N1_0_sigma_02_PS.mat','N1_Sp_N1_0_sigma_02_PS');
139
save('N1_Sp_N1_1_sigma_02_PS.mat','N1_Sp_N1_1_sigma_02_PS');
140
save('N1_Sp_N1_2_sigma_02_PS.mat','N1_Sp_N1_2_sigma_02_PS');
141
save('N1_Sp_N1_3_sigma_02_PS.mat','N1_Sp_N1_3_sigma_02_PS');
```

```
142
save('N2_Sp_N1_sigma_02_PS.mat','N2_Sp_N1_sigma_02_PS');
143
save('N2_Sp_N2_0_sigma_02_PS.mat','N2_Sp_N2_0_sigma_02_PS');
144
save('N2_Sp_N2_1_sigma_02_PS.mat','N2_Sp_N2_1_sigma_02_PS');
145
save('N2_Sp_N2_2_sigma_02_PS.mat','N2_Sp_N2_2_sigma_02_PS');
146
save('N2_Sp_N2_3_sigma_02_PS.mat','N2_Sp_N2_3_sigma_02_PS');
147         end
148         V1_S1_D1_PS{1,Indx_NeuPairs} = v1_s1_d1;
149         V2_S1_D1_PS{1,Indx_NeuPairs} = v2_s1_d1;
150         Spk_Time_S1_D1_PS{1,Indx_NeuPairs} = sp_time_s1_d1;
151         V1_S2_D2_PS{1,Indx_NeuPairs} = v1_s2_d2;
152         V2_S2_D2_PS{1,Indx_NeuPairs} = v2_s2_d2;
153         Spk_Time_S2_D2_PS{1,Indx_NeuPairs} = sp_time_s2_d2;
154
155         save('V1_S1_D1_PS.mat','V1_S1_D1_PS');
156         save('V2_S1_D1_PS.mat','V2_S1_D1_PS');
157         save('Spk_Time_S1_D1_PS.mat','Spk_Time_S1_D1_PS');
158         save('V1_S2_D2_PS.mat','V1_S2_D2_PS');
159         save('V2_S2_D2_PS.mat','V2_S2_D2_PS');
160         save('Spk_Time_S2_D2_PS.mat','Spk_Time_S2_D2_PS');
161         save('ALPHA2_PS.mat','ALPHA2_PS');
162         save('BETA2_PS.mat','BETA2_PS');
163
```

```
164
165     end
166     disp('i = ');
167     disp(i);
168     disp('j = ');
169     disp(j);
170     end
171 end
172
173 N1_Sp_N2_sigma_02_PS = N1_Sp_N2_sigma_02_PS / Iter_Num;
174 N1_Sp_N1_0_sigma_02_PS = N1_Sp_N1_0_sigma_02_PS / Iter_Num;
175 N1_Sp_N1_1_sigma_02_PS = N1_Sp_N1_1_sigma_02_PS / Iter_Num;
176 N1_Sp_N1_2_sigma_02_PS = N1_Sp_N1_2_sigma_02_PS / Iter_Num;
177 N1_Sp_N1_3_sigma_02_PS = N1_Sp_N1_3_sigma_02_PS / Iter_Num;
178
179 N2_Sp_N1_sigma_02_PS = N2_Sp_N1_sigma_02_PS / Iter_Num;
180 N2_Sp_N2_0_sigma_02_PS = N2_Sp_N2_0_sigma_02_PS / Iter_Num;
181 N2_Sp_N2_1_sigma_02_PS = N2_Sp_N2_1_sigma_02_PS / Iter_Num;
182 N2_Sp_N2_2_sigma_02_PS = N2_Sp_N2_2_sigma_02_PS / Iter_Num;
183 N2_Sp_N2_3_sigma_02_PS = N2_Sp_N2_3_sigma_02_PS / Iter_Num;
184
185 save('N1_Sp_N2_sigma_02_PS.mat', 'N1_Sp_N2_sigma_02_PS');
186 save('N1_Sp_N1_0_sigma_02_PS.mat', 'N1_Sp_N1_0_sigma_02_PS');
187 save('N1_Sp_N1_1_sigma_02_PS.mat', 'N1_Sp_N1_1_sigma_02_PS');
188 save('N1_Sp_N1_2_sigma_02_PS.mat', 'N1_Sp_N1_2_sigma_02_PS');
189 save('N1_Sp_N1_3_sigma_02_PS.mat', 'N1_Sp_N1_3_sigma_02_PS');
190 save('N2_Sp_N1_sigma_02_PS.mat', 'N2_Sp_N1_sigma_02_PS');
```

```

191 save('N2_Sp_N2_0_sigma_02_PS.mat','N2_Sp_N2_0_sigma_02_PS');
192 save('N2_Sp_N2_1_sigma_02_PS.mat','N2_Sp_N2_1_sigma_02_PS');
193 save('N2_Sp_N2_2_sigma_02_PS.mat','N2_Sp_N2_2_sigma_02_PS');
194 save('N2_Sp_N2_3_sigma_02_PS.mat','N2_Sp_N2_3_sigma_02_PS');

```

**New\_Input\_Generator.m.** This function gets the ramp input strategy for a neuron pair and decide whether this ramp input can help the target neuron spiking alone.

```

1 function [ ut,dt,N1_Sp ] = New_Input_Generator(
alpha1,beta1,alpha2,beta2,epsilon,sigma,Vth,V_,P_th )
2 %UNTITLED Summary of this function goes here
3 % Detailed explanation goes here
4 %% Define parameters
5 dt = 0.01;
6 dx = 0.01;
7
8 n = (alpha2/beta2)/(alpha1/beta1) - epsilon;
9 c = (alpha2*beta1 - alpha1*beta2*n)/(beta2*n - beta1);
10 v1_init = 0.01;
11 T = 30;
12 V1 = 0;
13
14 while V1(end) <= Vth
15     [t,y] = ode45(@(t,y) c*y, [0 T], v1_init);
16     V1 = y;
17     T = T + 10;
18 end
19

```

```

20   ut = (alpha2 - alpha1)/(beta2*n - beta1)*V1;
21   if (dx > (sigma^2)/(-alpha2*V_+beta2*ut(end))) && (dt >
(4*dx^2)/(dx*(-alpha2*V_+beta2*ut(end))-sigma^2))
22       dt = 0.8*(4*dx^2)/(dx*(-alpha2*V_+beta2*ut(end))-sigma^2);
23       dt = round(dt,4);
24   end
25   new_t = 0:dt:t(end);
26   ut = interp1(t,ut,new_t,'spline');
27
28   N1_Sp = 0;
29
30   [ N1_Sp_Pro ] = New_Fokker_Spike_Solver(
alpha1,beta1,sigma,Vth,V_,ut,dt );
31   [ N2_Sp_Pro ] = New_Fokker_Spike_Solver(
alpha2,beta2,sigma,Vth,V_,ut,dt );
32
33   Pro_Crit = N1_Sp_Pro.*(1-N2_Sp_Pro);
34   N1_Sp = sum(Pro_Crit >= P_th);
35
36   if N1_Sp > 0
37       N1_Sp = 1;
38   end
39
40   if N1_Sp == 1
41       I = find(Pro_Crit >= P_th,1);
42   end
43

```

```
44     if N1_Sp == 0
45         [M,I] = max(Pro_Crit);
46     end
47
48     ut = ut(1:I);
49
50
51     end
```

**PWC\_RUN.m.** This function iterates single pulse uncontrollable pairs and decides which neuron pair can be saved by ramp input strategy.

```
1
2     alpha_nom = 1;
3     beta_nom = 1;
4     sigma = 0.2;
5     Vth = 1;
6     V_ = -4;
7     P_th = 0.9;
8     Epsilon_Rat = 0.1;
9
10    alpha2_min = 0.02;
11    alpha2_max = 1;
12    beta2_min = 0.02;
13    beta2_max = 1;
14    da2 = 0.02;
15    db2 = 0.02;
16    alpha2 = alpha2_min:da2:alpha2_max;
```

```

17  beta2 = beta2_min:db2:beta2_max;
18
19  n = numel(alpha2);
20  m = numel(beta2);
21
22  % load('Seq_Contr_Maps_Sigmas_0204.mat');
23  load('Seq_Contr_Maps_Sigmas_0204_ref.mat');
24  PWC_U = cell(m,n);
25  PWC_dt = cell(m,n);
26  % load('PWC_U.mat');
27  % load('PWC_dt.mat');
28
29  for i = 1:m
30      for j = 1:n
31          b2 = beta2(i);
32          a2 = alpha2(j);
33          a2b2_rat = a2 / b2;
34          if (b2 < beta_nom) && (a2 < alpha_nom) && (a2b2_rat < 1)
&& (Seq_Contr_Maps_Sigmas_0204_ref (i,j,1) == 2)
35
36              epsilon = Epsilon_Rat * (1/a2b2_rat - 1);
37
38              [ ut,dt,N1_Sp ] = New_Input_Generator(
a2,b2,alpha_nom,beta_nom,epsilon,sigma,Vth,V_,P_th );%target Neuron's
alpha,beta put first two positions
39
40              if N1_Sp == 1

```

```
41         for k = i:m
42             for h = 1:j
43                 if Seq_Contr_Maps_Sigmas_0204_ref (k,h,1)
44                     == 2
45                         Seq_Contr_Maps_Sigmas_0204_ref
46                         (k,h,1) = 5; %Neuron 2 controllable
47                         PWC_U{k,h} = ut;
48                         PWC_dt{k,h} = dt;
49                     end
50                 end
51             end
52         elseif N1_Sp == 0
53             for k = 1:i
54                 for h = j:n
55                     if Seq_Contr_Maps_Sigmas_0204_ref (k,h,1)
56                         == 2
57                             Seq_Contr_Maps_Sigmas_0204_ref
58                             (k,h,1) = 22; % same as 2, only Neuron 1 controllable
59                         end
60                     end
61                 end
62             end
63         end
```

```
64
65
save('Seq_Contr_Maps_Sigmas_0204_ref.mat','Seq_Contr_Maps_Sigmas_0204_r
ef');
66         save('PWC_U.mat','PWC_U');
67         save('PWC_dt.mat','PWC_dt');
68         disp('i = ');
69         disp(i);
70         disp('j = ');
71         disp(j);
72     end
73 end
```

## Bibliography

- [1] P. Dayan and L. F. Abbott, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems* (Computational Neuroscience Series), The MIT Press, 2005.
- [2] S. Flesher, J. Collinger, S. Foldes, J. Weiss, J. Downey and R. Gaunt, "Intracortical microstimulation of human somatosensory cortex," *Science Translational Medicine*, vol. 8, 2016.
- [3] G. Tabot, J. Dammann, J. Berg, F. Tenore, J. Boback, J. Vogelstein and S. Bensmaia, "Restoring the sense of touch with a prosthetic hand through a brain interface," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 110, no. 45, pp. 18279-18284, 2013.
- [4] J. Carmena, M. Lebedev, R. Crist, J. O'Doherty, D. Santucci, D. Dimitrov, P. Patil, C. Henriquez and M. Nicolelis, "Learning to control a brain-machine interface for reaching and grasping by primates," *PLoS Biology*, vol. 1, no. 2, 2003.
- [5] A. Yazdan-Shahmorad, C. Diaz-Botia, T. Hanson, V. Kharazia, P. Ledochowitsch, M. Maharbiz and P. Sabes, "A large-scale interface for optogenetic stimulation and recording in nonhuman primates," *Neuron*, vol. 89, no. 5, pp. 927-939, 2016.
- [6] J. Bernstein and E. Boyden, "Optogenetic tools for analyzing the neural circuits of behavior," *Trends in Cognitive Sciences*, vol. 15, no. 12, pp. 592-600, 2011.
- [7] L. Fenno, O. Yizhar and K. Deisseroth, "The Development and Application of Optogenetics," *Annual Review of Neuroscience*, vol. 34, pp. 389-412, 2011.
- [8] K. Deisseroth, "Optogenetics," *Nature Methods*, vol. 8, no. 1, pp. 26-29, 2011.
- [9] J. Zhang, F. Laiwalla, J. Kim, H. Urabe, R. V. Wagenen, Y.-K. Soon, B. W. Connors, F. Zhang, K. Deisseroth and A. Nurmikko, "Integrated device for optical stimulation and spatiotemporal electrical recording of neural activity in light-sensitized brain tissue," *Journal of Neural Engineering*, vol. 6 no. 5, p. 055007, 2009.
- [10] P. Anikeeva, A. S. Andalman, I. Witten, M. Warden, I. Goshen, L. Grosenick, L. A. Gunaydin, L. M. Frank and K. Deisseroth, "Optetrode: a multichannel readout for optogenetic control in freely moving mice," *Nature Neuroscience*, vol. 15, no. 1, pp. 163-170, 2012.

- [11] S. Peron and K. Svoboda, "From cudgel to scalpel: toward precise neural control with optogenetics," *Nature Methods*, vol. 8, no. 1, pp. 30-34, 2011.
- [12] B. Chow and E. Boyden, "Optogenetics and Translational Medicine," *Science Translational Medicine*, vol. 5, no. 177, 2013.
- [13] J. Robinson, M. Jorgolli, A. Shalek, M.-H. Yoon, R. Gertner and H. Park, "Vertical nanowire electrode arrays as a scalable platform for intracellular interfacing to neuronal circuits," *Nature Nanotechnology*, vol. 7, no. 3, pp. 180-184, 2012.
- [14] C. McIntyre, W. Grill, D. Sherman and N. Thakor, "Cellular Effects of Deep Brain Stimulation: Model-Based Analysis of Activation and Inhibition," *Journal of Neurophysiology*, vol. 91, no. 4, pp. 1457-1469, 2004.
- [15] R. Plonsey and R. C. Barr, *Bioelectricity A Quantitative Approach*, Springer, 2007.
- [16] C. Butson and C. McIntyre, "Current Steering to Control the Volume of Tissue Activated During Deep Brain Stimulation," *Brain Stimulation*, vol. 1, no. 1, pp. 7-15, 2008.
- [17] J. Ritt and S. Ching, "Neurocontrol: Methods, Models and Technologies for Manipulating Dynamics in the Brain," in *2015 American Control Conference*, Chicago, IL, USA, 2015.
- [18] J. MacGregor and D. T. Fogal, "Closed-loop identification: the role of the noise model and prefilters," *Journal of Process Control*, vol. 5, no. 3, pp. 163-171, 1995.
- [19] W. Bequette, *Process Control: Modeling, Design and Simulation*, Prentice Hall, 2003.
- [20] J. S. Perlmutter and J. W. Mink, "Deep brain stimulation," *Annual Review of Neuroscience*, vol. 29, 2006, pp. 229-257, 2006.
- [21] S. J. Groiss, L. Wojtecki, M. Sudmeyer and A. Schnitzler, "Deep brain stimulation in Parkinson's disease," *Therapeutic Advances in Neurological Disorders*, vol. 2, no. 6, pp. 379-391, 2009.
- [22] M. C. Rodriguez-Oroz, J. A. Obeso, A. E. Lang, J.-L. Houeto and N. V. Blercom, "Bilateral deep brain stimulation in Parkinson's disease: a multicentre study with 4 years follow-up," *Brain*, vol. 128, no. 10, pp. 2240-2249, 2005.
- [23] J. P. Rauschecker and R. V. Shannon, "Sending Sound to the Brain," *Science*, vol. 295, no. 5557, pp. 1025-1029, 2002.

- [24] L. Friesen, R. Shannon, D. Baskent and X. Wang, "Speech recognition in noise as a function of the number of spectral channels: Comparison of acoustic hearing and cochlear implants," *Journal of the Acoustical Society of America*, vol. 110, no. 2, 2001.
- [25] T. Ching, P. Incerti and M. Hill, "Binaural Benefits for Adults Who Use Hearing Aids and Cochlear Implants in Opposite Ears," *Ear and Hearing*, vol. 25, no. 1, pp. 9-21, 2004.
- [26] J. A. Assad, L. Battelli and M. Aghdaee, "Relative timing: From behaviour to neurons," *Philosophical Transaction of The Royal Society B*, vol. 369, no. 1637, p. 20120472, 2014.
- [27] P. J. Drew and L. F. Abbott, "Extending the effects of spike-timing-dependent plasticity to behavioral timescales," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 103, no. 23, pp. 8876-8881, 2006.
- [28] V. Gauck and D. Jaeger, "The Control of Rate and Timing of Spikes in the Deep Cerebellar Nuclei by Inhibition," *Journal of Neuroscience*, vol. 20, no. 8, pp. 3006-3016, 2000.
- [29] E. Brown, J. Moehlis and P. Holmes, "On the Phase Reduction and Response Dynamics of Neural Oscillator Populations," *Neural Computation*, vol. 16, p. 673-715, 2004.
- [30] E. Brown, J. Moehlis, P. Holmes, E. Clayton, J. Rajkowski and G. Aston-Jones, "The Influence of Spike Rate and Stimulus Duration on Noradrenergic Neurons," *Journal of Computational Neuroscience*, vol. 17, no. 1, p. 13-29, 2004.
- [31] G. Lajoie and E. Shea-Brown, "Shared Inputs, Entrainment, and Desynchrony in Elliptic Bursters: From Slow Passage to Discontinuous Circle Maps," *SIAM Journal on Applied Dynamical Systems*, vol. 10, no. 4, p. 1232-1271, 2011.
- [32] S. Gu, F. Pasqualetti, M. Cieslak, Q. Telesford, A. Yu, A. Kahn, J. Medaglia, J. Vettel, M. Miller, S. Grafton and D. Bassett, "Controllability of structural brain networks," *Nature Communications*, vol. 6, p. 8414, 2015.
- [33] I. Dasanayake and J.-S. Li, "Optimal design of minimum-power stimuli for phase models of neuron oscillators," *Physical Review E*, vol. 83, no. 6, 2011.
- [34] P. Danzl, R. Hansen, G. Bonnet and J. Moehlis, "Partial phase synchronization of neural populations due to random Poisson inputs," *Journal of Computational Neuroscience*, vol. 25, no. 1, pp. 141-157, 2008.

- [35] P. Danzl and J. Moehlis, "Event-based feedback control of nonlinear oscillators using phase response curves," in *2007 46th IEEE Conference on Decision and Control*, New Orleans, LA, USA, 2007.
- [36] T. Kano and S. Kinoshita, "Control of individual phase relationship between coupled oscillators using multilinear feedback," *Physical Review E*, vol. 81, no. 2, pt. 2, p. 026206, 2010.
- [37] J. Moehlis, E. Shea-Brown and H. Rabitz, "Optimal inputs for phase models of spiking neurons," *Journal of Computational and Nonlinear Dynamics*, vol. 1, no. 4, pp. 358-367, 2006.
- [38] R. Agarwal and S. Sarma, "Restoring the Basal Ganglia in Parkinson's disease to Normal via Multi-Input Phase-Shifted Deep Brain Stimulation," in *Conference Proceedings: IEEE Engineering in Medicine and Biology*, vol. 2010, pp. 1539-1542, 2010.
- [39] C. J. Wilson, B. Beverlin II and T. Netoff, "Chaotic desynchronization as the therapeutic mechanism of deep brain stimulation," *Frontiers in Systems Neuroscience*, vol. 5, p. 50, 2011. doi: 10.3389/fnsys.2011.00050
- [40] P. Danzl, J. Hespanha and J. Moehlis, "Event-based minimum-time control of oscillatory neuron models," *Biological Cybernetics*, vol. 110, no. 5-6, pp. 387-399, 2009.
- [41] P. Danzl, A. Nabi and J. Moehlis, "Charge-balanced spike timing control for phase models of spiking neurons," *Discrete and Continuous Dynamical Systems*, vol. 28, no. 4, p. 1413-1435, 2010.
- [42] D. Wilson and J. Moehlis, "A Hamilton-Jacobi-Bellman approach for termination of seizure-like bursting," *Journal of Computation Neuroscience*, vol. 37, no. 2, p. 345-355, 2014.
- [43] A. Nabi and J. Moehlis, "Single input optimal control for globally coupled neuron networks," *Journal of Neural Engineering*, vol. 8, no. 6, 2011.
- [44] A. Nabi, M. Mirzadeh, F. Gibou and J. Moehlis, "Minimum energy desynchronizing control for coupled neurons," *Journal of Computational Neuroscience*, vol. 34, no. 2, pp. 259-271, 2013.
- [45] M. Zeitler and P. A. Tass, "Computationally Developed Sham Stimulation Protocol for Multichannel Desynchronizing Stimulation," *Frontiers in Physiology*, vol. 9, p. 512, 2018.

- [46] S. Ching and J. T. Ritt, "Control strategies for underactuated neural ensembles driven by optogenetic stimulation," *Frontiers in Neural Circuits*, vol. 7, article 54, 2013.
- [47] O. Miranda-Dominguez, J. Gonía and T. I. Netoff, "Firing rate control of a neuron using a linear proportional-integral controller," *Journal of Neural Engineering*, vol. 7, no. 6, 2010.
- [48] T. Stigen, P. Danzl, J. Moehlis and T. Netoff, "Controlling spike timing and synchrony in oscillatory neurons," *Journal of Neurophysiology*, vol. 105, no. 5, pp. 2074-2082, 2011.
- [49] A. Nabi, T. Stigen, J. Moehlis and T. Netoff, "Minimum energy control for in vitro neurons," *Journal of Neural Engineering*, vol. 10, no. 3, article 036005, 2013.
- [50] Y. Ahmadian, A. M. Packer, R. Yuste and L. Paninski, "Designing optimal stimuli to control neuronal spike timing," *Journal of Neurophysiology*, pp. 1038-1053, 2011.
- [51] A. Lolov, S. Ditlevsen and A. Longtin, "Stochastic optimal control of single neuron spike trains," *Journal of Neural Engineering*, vol. 11, article 046004, 2014.
- [52] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of Physiology*, vol. 117, no. 4, pp. 500-544, 1952.
- [53] C. McIntyre, W. Grill, D. Sherman and N. Thakor, "Cellular effects of deep brain stimulation: model-based analysis of activation and inhibition," *Journal of Neurophysiology*, vol. 91, no. 4, pp. 1457-1469, 2004.
- [54] E. M. Izhikevich, "Which Model to Use for Cortical Spiking Neurons?" *IEEE Transactions on Neural Networks*, vol. 15, no. 5, p. 1063-1070, 2004.
- [55] A. Nandi, J. T. Ritt and S. Ching, "Non-negative Inputs for Underactuated Control of Spiking in Coupled Integrate-and-Fire Neurons," in *53rd IEEE Conference on Decision and Control*, Los Angeles, 2014.
- [56] A. Nandi, H. Schattler, J. T. Ritt and S. Ching, "Fundamental Limits of Forced Asynchronous Spiking with Integrate and Fire Dynamics," *The Journal of Mathematical Neuroscience*, vol. 7, article 11, 2017.
- [57] C. W. Gardiner, *Handbook of Stochastic Methods for Physics, Chemistry and the Natural Sciences*, Berlin, Heidelberg, New York, Tokyo: Springer - Verlag, 1985.

- [58] P. E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*, Berlin: Springer-Verlag Berlin Heidelberg, 1992.
- [59] W. N. Frost, L.-M. Tian, T. A. Hoppe, D. L. Mongeluzi and J. Wang, "A Cellular Mechanism for Prepulse Inhibition," *Neuron*, vol. 40, no. 5, pp. 991-1001, 2003.
- [60] C. M. Tweedie, "Statistical properties of inverse Gaussian distributions – I," *Annals of Mathematical Statistics*, vol. 28, no. 2, pp. 362–377, 1957.
- [61] M. C. Tweedie, "Statistical properties of inverse Gaussian distributions – II," *Annals of Mathematical Statistics*, vol. 28, no. 3, pp. 696–705, 1957.
- [62] G. A. Whitmore and A. H. Neufeldt, "An application of statistical models in mental health research," *The Bulletin of Mathematical Biophysics*, vol. 32, no. 4, pp. 563-579, 1970.
- [63] T. Lancaster, "A Stochastic Model for the Duration of a Strike," *Journal of the Royal Statistical Society*, vol. 135, no. 2, pp. 257-271, 1972.
- [64] V. Giorno, P. Lansky, A. G. Nobile and L. M. Ricciardi, "Diffusion approximation and first-passage-time problem for a model neuron," *Biological Cybernetics*, vol. 58, no. 6, p. 387–404, 1987.
- [65] W. Braun, P. C. Matthews and R. Thul, "First passage times in integrate-and-fire neurons with stochastic thresholds," *Physical Review E*, vol. 91, no. 5, 2015.
- [66] P. Lansky and C. E. Smith, "The effect of a random initial value in neural first-passage-time models," *Mathematical Biosciences*, vol. 93, no. 2, pp. 191-215, 1989.
- [67] B. J. Breen and W. C. Gerken, "Hybrid Integrate-and-Fire Model of a Bursting Neuron," *Neural Computation*, vol. 15, no. 12, pp. 2843-2862, 2003.
- [68] V. S. Sohal, . S. Pangratz-Fuehrer, U. Rudolph and J. R. Huguenard, "Intrinsic and Synaptic Dynamics Interact to Generate Emergent Patterns of Rhythmic Bursting in Thalamocortical Neurons," *Journal of Neuroscience*, vol. 26, no. 16, p. 4247–4255, 2006.
- [69] J. R. Gibson, M. Beierlein and B. Connors, "Functional Properties of Electrical Synapses Between Inhibitory Interneurons of Neocortical Layer 4," *Journal of Neurophysiology*, vol. 93, no. 1, p. 467– 480, 2005.
- [70] M. Devor and V. Zalkind, "Reversible analgesia, atonia, and loss of consciousness on bilateral intracerebral microinjection of pentobarbital," *Pain*, vol. 94, no. 1, pp. 101-112, 2001.

- [71] E. Brown, P. Holmes and J. Moehlis, *Globally Coupled Oscillator Networks*, New York: Springer-Verlag, 2003.
- [72] A. H. Cohen, P. Holmes and R. Rand, "The nature of the coupling between segmental oscillators of the lamprey spinal generator for locomotion: A mathematical model," *Journal of Mathematical Biology*, vol. 13, no. 3, pp. 345-369, 1982.
- [73] P. Ashwin and J. Swift, "The dynamics of n weakly coupled identical oscillators," *Journal of Nonlinear Science*, vol. 2, no. 1, pp. 69-108, 1992.
- [74] K. Nancy and G. B. Ermentrout, "Phase Transitions and Other Phenomena in Chains of Coupled Oscillators," *SIAM Journal on Applied Mathematics*, vol. 50, no. 4, p. 1014–1052, 1990.
- [75] L. Bachelier, "Théorie de la Spéculation," *Annales Scientifiques de l'École Normale Supérieure*, vol. 3, no. 17, pp. 21-86, 1900.
- [76] V. E. Schrodinger, "Zur Theorie der Fall- und Steigversuche an Teilchen mit Brownscher Bewegung," *Physikalische Zeitschrift*, vol. 16, p. 289–295, 1915.

**CURRICULUM VITAE**

