University of California
Santa Barbara

Adaptive Strategies for Oscillatory Systems: Integrating Machine Learning and Control Techniques with Applications in Neuroscience

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Mechanical Engineering

by

Timothy Dandeneau Matchen

March 2021

The Dissertation of Timothy Dandeneau Matchen is approved.

Professor Igor Mezic

Professor Bassam Bamieh

Professor Greg Ashby

Professor Jeff Moehlis, Committee Chair

December 2020

Adaptive Strategies for Oscillatory Systems: Integrating Machine Learning and Control Techniques with Applications in Neuroscience

To my parents and my sibling for their steadfast support.

To my friends for being there for me when I need them.
To my D&D groups for alternately keeping me sane and driving me insane.

Acknowledgements

Thank you first and foremost to Professor Jeff Moehlis, whose advice throughout my graduate career has been indispensable. I would also like to thank my committee for their feedback in my advancement to candidacy and for taking the time to read this dissertation. Lastly, thank you to the Starr Laboratory at UCSF for their guidance in analyzing Parkinsonian patient data and generating meaningful results.

Curriculum Vitæ
Timothy Dandeneau Matchen

Education

2021	Ph.D. in Mechanical Engineering (Expected) with Certificate in Neuroengineering, University of California, Santa Barbara.
2014	B.S. in Mechanical and Aerospace Engineering with Certificate in Engineering Physics, Princeton University.

Publications

- T. D. Matchen, R. Gilron, S. Little, P. Starr, and J. Moehlis, *A data-driven, machine-learning based approach to adaptive deep brain stimulation* (2021) (*In preparation*)
- A. Yates, T. D. Matchen, and J. Moehlis, *Extraction of phase and isostable response curves via deep neural networks* (2021) (*In preparation*)
- T. D. Matchen and J. Moehlis, *Leveraging deep learning to control neural oscillators* (2021) (*Submitted*)
- B. Monga, D. Wilson, T. Matchen, and J. Moehlis, *Phase reduction and phase-based optimal control for biological systems: a tutorial*, *Biological Cybernetics* **113** (2019), no. 1-2.
- T. D. Matchen and J. Moehlis, *Phase model-based neuron stabilization into arbitrary clusters*, *Journal of Computational Neuroscience* **44** (2018), no. 3 363–378.
- T. Matchen and J. Moehlis, *Real-Time Stabilization of Neurons into Clusters*, in *American Controls Conference*, (Seattle), pp. 2805–2810, 2017.
- N. Govindarajan, H. Arbabi, L. Van Blargian, T. Matchen, E. Tegling, and I. Mezic, *An operator-theoretic viewpoint to non-smooth dynamical systems: Koopman analysis of a hybrid pendulum*, in *2016 IEEE 55th Conference on Decision and Control, CDC 2016*, 2016.
- T. Matchen and B. Nadler, *Image-based target tracking using least-squares trajectory estimation without a priori knowledge*, in *IEEE Aerospace Conference Proceedings*, 2014.

Abstract

Adaptive Strategies for Oscillatory Systems: Integrating Machine Learning and Control Techniques with Applications in Neuroscience

by

Timothy Dandeneau Matchen

Oscillators are ubiquitous in nature. As such, a significant body of literature has been devoted to studying their dynamics and how to control those dynamics. Many systems, however, do not maintain the core assumptions guiding the development of this literature–systems that are either too complex or too poorly understood to allow for simple mathematical representations. Machine learning can serve as a powerful tool to supplement our understanding of dynamical systems in situations where traditional methods fail. In this dissertation, we develop first a control strategy for oscillators using standard techniques and assumptions about our dynamical system, then explore the ways in which machine learning can replace some of the strictest requirements on developing control strategies. We demonstrate how machine learning can extract meaningful information from complex systems in neuroscience and use that information as the basis for control. Lastly, we discuss some emerging strategies for further marrying the disciplines of dynamics and control and machine learning.

Contents

Chapter 1

Dynamics, Control, and Neuroscience

1.1 Computational Modeling of the Brain

The brain is complicated. This is by no means a revelatory statement, but it is an important one nonetheless, and even accepting that the brain is complicated feels as though it does not do justice to the sheer complexity of the brain – it contains a mind-boggling 100 billion neurons, each with thousands of synaptic connections, amounting to a total of roughly 10^{15} total connections within the brain [1]. While advances in neuroimaging allow us to more accurately map the brain than ever before, empirical study presents a number of ongoing challenges, such as difficulties in reproducibility of results with small cohort sizes or low statistical power, as well as the ongoing need for invasive strategies for study, which frequently cannot be carried out on human patients [2]. Beyond the not-so-simple act of studying the brain, developing novel techniques for therapeutic procedures or other active forms of modification to the brain can be incredibly costly and resource-intensive, potentially requiring hardware, live creatures to study, and more.

All of these factors make neuroscience in many ways an ideal discipline for leveraging computational modeling coupled with a firm understanding of the principles of dynamics and control. While *in vivo* testing is inherently limited by requiring live patients or test subjects, a robust computational model can allow us to consider infinite variations, parameter adjustments, and algorithms for a fraction of the cost and time commitment. Moreover, an understanding of dynamics can allow us to take these models a step further, providing the possibility for further simplification thanks to intelligent application of the principles of dynamics and control. In turn, the lessons we learn in the context of neuroscience can typically be generalized to a far broader class of problems: just as neuroscience can learn from dynamics, so too can dynamics learn from neuroscience.

Scope in Computational Modeling

As already noted, there are hundreds of billions of neurons to contend with in the brain. Trying to build a model from first principles of the entire brain is nearly computationally prohibitive, especially so without the aid of supercomputing. At the same time, important information can be derived from the interactions between individual neurons within the brain. As a result, a key idea in computational modeling of the brain is that of *scope*: namely, the trade-off between granularity in a model and computational complexity. Although simulating 100 billion neurons is typically infeasible, simulating 100 is not, and this may be the appropriate approach at times. Other times, a researcher may wish to study intently a single neuron and may implement a level of granularity that makes even interconnecting two neurons computationally taxing. Broadly speaking, we can describe four levels of granularity in computational modeling:

1. The analysis of a single neuron or sub-component of a neuron, considering both time and spatial evolutions of chemical concentrations, electric potential, and protein activation/inactivation;
2. A system of interconnected neurons, considering synaptic effects and coupling;
3. Modeling of a local brain region, such as a specific structure or lobe; and
4. System-level modeling and analysis.

The analysis in this dissertation will primarily focus on the second and third tiers of granularity – either small populations of oscillatory neurons or modeling of local field potentials for a specific region of the brain.

1.2 Core Dynamical Systems Concept: Phase Reduction

More specifically, in general we will be considering *oscillatory* systems: those with a stable limit cycle and that exhibit periodic behavior. Oscillators are an important component of numerous biological processes beyond just neurons in the brain, including circadian rhythms, cardiac pacemaker cells, and motor control, and developing effective methods of controlling these oscillators is an important goal. This is especially true in neuroscience, where pathological activity may be linked to improper functioning of neural oscillators, cf. [3].

One particularly useful technique when studying oscillators is phase model reduction. Because dynamical systems (especially those modeling neurons) can be high-dimensional, designing control algorithms for them from first principles can be difficult or computationally intensive. We can instead leverage the fact that the dynamical system has stable, recurrent behavior to simplify the dynamics to a lower-dimensional representation. We can replace the dynamics in $\mathbb{R}^n$ with a one-dimensional system instead, where the state is represented instead by its *phase* θ. This phase ranges from 0 to 2π and represents the oscillator's progression through its periodic orbit. Absent any perturbations, because the oscillator inhabits a stable periodic orbit, we can additionally characterize it by its *natural frequency* ω, equal to 2π divided by its period. In the context of our work in neuroscience, because the firing neuron has a fixed, stable limit cycle, following the work in [4, 5, 6] we can therefore reduce the dynamics when the oscillator's state is near the limit cycle to the representation:

$$\dot{\theta} = \omega + Z\left(\theta\right) u\left(t\right), \tag{1.1}$$

where $\dot{\theta}$ describes the evolution of the oscillator and $u\left(t\right)$ is the control input. $Z\left(\theta\right)$ is known as the *phase response curve* and describes the sensitivity of the phase to a stimulus. By implementing the phase model reduction, we can design control strategies by studying the effects on a far simpler model. For example, once the phase model reduction is derived, it is straightforward to apply traditional optimization techniques, such as utilizing variational calculus to control an oscillator's period [7, 8] or applying control to manipulate a neural population's distribution [9, 10, 11]. Provided we do not stray too far from the limit cycle, the results will reliably transfer to the full, un-reduced model.

What qualifies as “sufficiently close” to the limit cycle, however, is a challenging question. Fortunately, we can augment our phase model reduction to attempt to quantify this distance from the limit cycle and utilize that information effectively. We do this by supplementing the phase model reduction with a technique known as isostable reduction, developed primarily in [12]. We can measure the relative distance to the limit cycle by evaluating a second differential equation, this time for the value of the isostable coordinate Ψ, which is equal to 0 on the limit cycle:

$$\dot{\Psi} = k\Psi + \mathcal{I}(\theta)\, u(t), \tag{1.2}$$

where k describes the rate at which the system returns to the limit cycle and $\mathcal{I}$ is an analog to the phase response curve known as the *isostable response curve*.

As implied by the name, the isostable response curve determines the response of the system's isostable coordinate to an input stimulus. We have neglected thus far to define what this isostable coordinate is, however, and we should note as well it possesses an analog in the traditional phase model reduction known as the isochron. Put simply, for a system of differential equations in $\mathbb{R}^n$ an isostable is the set of all points in $\mathbb{R}^n$ that approach the limit cycle at the same rate and asymptotically approach the limit cycle at the same time; the isochron represents a similar concept – it is the set of all points that converge to the same phase on the limit cycle as $t \to \infty$. For example, we may displace a state from the limit cycle, but it will eventually asymptotically near the limit cycle and continue oscillating. The isostable coordinate tells us how long it will take to return to the proximity of the limit cycle, while the isostable coordinate (or phase) tells us where on the limit cycle the system would be were it to be on the limit cycle. Understanding the interplay between these two concepts can allow us to generalize our definition of the phase response and isostable response to the entirety of $\mathbb{R}^n$ space, allowing us to consider the dynamics at any distance from the limit cycle. In doing so, we adjust our phase response curve to be functions of both coordinates instead of just phase:

$$\dot{\theta} = \omega + Z(\theta, \Psi)\, u(t) \tag{1.3}$$

$$\dot{\Psi} = k\Psi + \mathcal{I}(\theta, \Psi)\, u(t). \tag{1.4}$$

These are known instead as the *global phase response curve* and the *global isostable response curve*, respectively.

There are a number of methods for calculating the phase and isostable response curves, both numeric and analytical; we will discuss some of these later in this dissertation, but some strategies are the so-called “adjoint method” cf. [5], direct numerical methods [13, 14], and methods derived from Koopman operator theory [15, 16].

1.3 Oscillators in the Brain: Parkinson’s Disease and Deep Brain Stimulation

The primary motivation for this emphasis on oscillatory activity in the context of neuroscience is improving our understanding of the relationship between Parkinson’s Disease (PD) and deep brain stimulation. Parkinson’s Disease is a neurodegenerative disease associated with the early death of dopamine neurons, specifically in the substantia nigra. The primary treatment strategy for PD is levodopa (L-Dopa), a dopamine precursor. L-Dopa medication, however, is not without drawbacks, with potential issues in long-term use arising from both fluctuations in medication level over the course of a day (”on-off” effects) and emergent symptoms such as dyskinesia [17]. Additionally, L-Dopa does not arrest progression of PD, and symptoms may deteriorate despite regular medication. In these cases, deep brain stimulation (DBS) may be used as an additional treatment strategy in conjunction with continuing medication. An effective DBS implementation confers numerous benefits that both augment the effectiveness of L-Dopa and ameliorate certain side effects [18, 19, 20, 21].

If computational modeling represents the core goal of assessing the dynamics of the brain, then deep brain stimulation is one of our most promising strategies for control.

DBS in particular has seen numerous applications, from the aforementioned reduction of symptoms in PD [22, 23, 24, 25] to treatment for Tourette Syndrome [26], essential tremor, and various other disorders. In DBS, neural dynamics are modulated by an electrode implanted in the brain tissue; pulsatile stimuli are sent via the electrode into the target brain region. The tuning of stimulation parameters, such as frequency and amplitude, is carried out by a neurologist over the course of several sessions in a manual, time-intensive process [27]. Despite its proven effectiveness, the mechanisms by which DBS alleviates symptoms are not fully understood. Additionally, there are risks associated with DBS, both related to the surgical procedure and hardware as well as to the chronic usage in combating the symptoms of PD [28, 29].

For these reasons, there have been various attempts in recent years to not only better understand the processes that allow for the success of DBS, but also to understand ways to reduce the possible negative side-effects. While the exact mechanisms regulating deep brain stimulation are as yet unclear, many studies point to one possible explanation being pathological oscillatory behavior in the affected brain regions, specifically, elevated levels of synchrony. Here it is again important to recall the varying levels of granularity in computational modeling and indeed in data collection: terms such as synchronization will have different meanings depending on what level of granularity we are considering.

On a neuronal level, synchrony may correspond to populations of neurons firing simultaneously or nearly simultaneously, and based on this framework there has been experimental and theoretical evidence [30, 31, 32] that the reduction of this synchrony is correlated to the alleviation of symptoms. One approach to achieve partial desynchronization is to split the oscillator neurons into clusters, in which only a subpopulation of the neurons are spike-synchronized. In fact, [31] suggests that the standard DBS protocol leads to a clustering of oscillators, in which the population of neurons divides into synchronized subpopulations. Additionally, we note that the beneficial effects of clustering for Parkinson's relief can also be inferred from [33].

This "clustering" approach has found success in coordinated reset, which involves using multiple electrode implants delivering a series of identical impulses separated by a time delay between implants. This has been studied extensively [30, 34, 25, 24] with preliminary clinical success [23]. Modeling and clinical results for coordinated reset suggest that relatively strongly clustered groups of neurons do not lead to pathological outcomes in the user and can be effective in treatment for Parkinson's Disease. Coordinated reset does, however, possess some of the same challenges of traditional DBS methods, namely an inflexibility in stimulation strategy and the requirement of hand-tuning appropriate stimulation parameters.

In the context of a local brain region, synchrony is instead the inferred explanation for an elevation in the activity of a particular region of the frequency domain. Increased power in the so-called "beta band" (approximately 15-35 Hz) is associated with Parkinsonian symptoms. Elevated activity in this frequency range has been correlated to slowness of movement [35] and has more generally been observed in numerous studies of Parkinson's, see [36, 37, 38, 39].

A more flexible approach based on this analysis, adaptive deep brain stimulation, is presented in [40, 41]. Here, stimulation is triggered by a spiking of power in the beta band, with the stimulation turning on once the power exceeds a threshold. This holds the appeal of significantly reducing energy usage by DBS, which in turn minimizes the likelihood of adverse side effects as well as the requirements of battery replacement. A core drawback and core challenge presented by this strategy, however, is that it is inherently reactive, not proactive; ideally, a stimulation strategy would be rooted in identifying impending pathological activity prior to its onset rather than *post hoc.*

With a host of challenges associated with present implementations and a variety of modeling options, Parkinson's Disease and DBS are a fertile test bed for novel approaches to analyzing dynamics and designing control in complex systems. While the focus of the dissertation is broader than just the question of Parkinson's Disease, it will serve as a useful set of benchmarks that can effectively highlight the advantages and drawbacks of different methods we have explored.

1.4 Machine Learning and Artificial Neural Networks

In particular, the challenges of PD will present in stark relief the limitations of employing classical methods from dynamics and control in the context of increasingly complex systems. While concepts such as optimal control and phase model reduction are powerful tools for approaching novel dynamical systems, it became clear that we needed to develop new methods to add to our toolbox if we were to effectively tackle the problems of PD and other open questions, and machine learning (ML) provides effective strategies for answering these. That being said, our roots remain firmly planted in the world of dynamics and control, and the core tenet throughout our forays into the world of machine learning was supplementing rather than supplanting classical approaches to dynamics and control.

This dissertation will primarily consider a particular branch of ML known as artificial neural networks (ANNs). Despite the moniker, neural networks are not yet widely-used by the neural control community, though interest is growing; recent results [42, 43] have begun leveraging reinforcement learning to generate control strategies.

An ANN consists of a system that maps an input vector to an output vector via one or more layers; the system is a "deep" network if at least one of these layers is "hidden," meaning it does not map directly to the output. Each layer is comprised of a number of neurons, each consisting of a matrix that linearly transforms the output from the previous layer followed by a nonlinear activation function. Training a neural network consists of running repeated iterations of two steps: a forward propagation step followed by a backward propagation step. In the forward propagation step, the current parameters of the neural network are used to make a prediction of the output based on the input, which is then compared to the actual output. This provides a loss function which is then fed back into the neural network to compute appropriate derivatives for updating the weights on the matrices in each layer of the neural network. A schematic of this process is provided in Fig 1.1. An appealing aspect of neural networks is that, for some finite number of artificial neurons, any smooth function can be accurately approximated by a neural network consisting of only a single hidden layer [44]. In practice, however, it is typically more efficient to sacrifice breadth – a large number of neurons in a layer – for depth, a larger number of hidden layers. Effectively, the deep neural network architecture allows nonlinearities to compound at each layer, providing for the approximation of highly nonlinear functions with less computational cost than a simpler, single-hidden layer network.

One key drawback of ANNs is that they are typically considered to be "black box" solutions – an input goes in, and output comes out, but what happens in between is difficult to extract. As mechanical engineers, this can be unnerving – we like to know what's happening, and the usefulness of an answer without context is severely limited. Throughout this research, whenever machine learning is used, we have attempted to approach it from a "grey box" perspective instead, where core principles of dynamics and control are preserved despite the implementation of a neural network.

1.5 What Lies Ahead

The rest of this dissertation will be structured as follows. In Chapter 2, we will explore the problems of clustering and control using the phase model reduction and

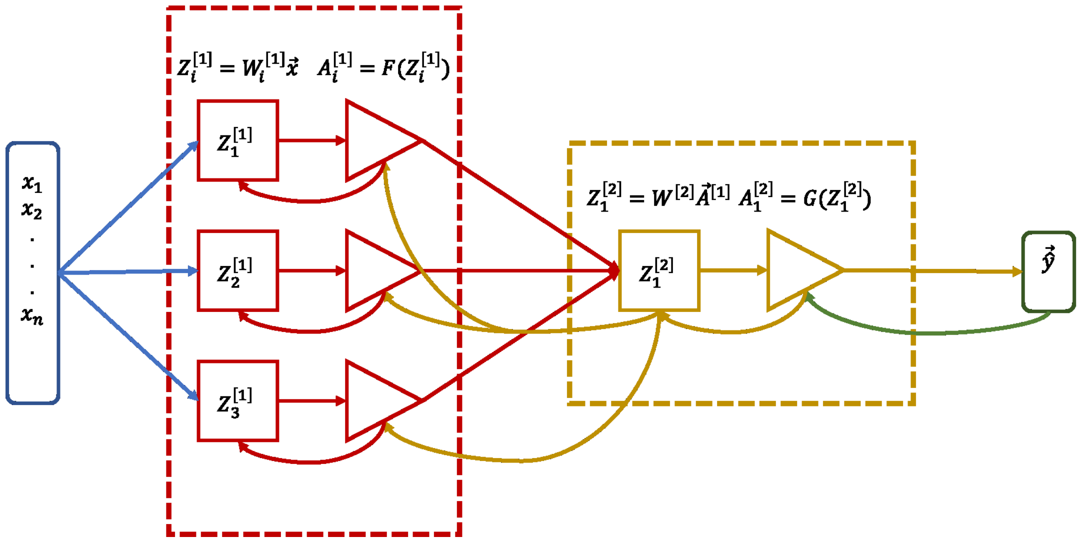

$Z_i^{[1]} = W_i^{[1]}\vec{x}$ $A_i^{[1]} = F(Z_i^{[1]})$
x_1
x_2
x_n
$Z_1^{[1]}$
$Z_2^{[1]}$
$Z_3^{[1]}$
$Z_1^{[2]} = W^{[2]}\vec{A}^{[1]}$ $A_1^{[2]} = G(Z_1^{[2]})$
$Z_1^{[2]}$
$\vec{\hat{y}}$

Figure 1.1: **Schematic of a neural network with one hidden layer.** This neural network contains one hidden layer (in red) with three neurons and an output layer (in yellow) with one neuron. The input $\vec{x}$ (blue) is applied to each neuron in the hidden layer. The outputs of the hidden layer's activation function are passed to the output layer, which yields the system's estimate $\hat{y}$ (green). The result is then fed backward for back propagation.

isostable techniques, then provide some commentary on the advantages and drawbacks of this strategy. In Chapter 3, we will approach the problem of clustering again, but this time, we will make use of ANNs and generate control strategies based on both the results from Chapter 2 and another classical strategy in dynamics and control, dynamic programming. Chapter 4 will then serve as a case study for the power of machine learning in approaching data-driven problems in dynamics and control. Using real patient data, we will develop a predictive model for the future state of our observable. Following this, we will use the same core structure to create an adaptive control strategy via artificial neural networks. In Chapter 5, we will conclude by reviewing our efforts to marry traditional dynamics and control with machine learning and discuss our ongoing work in both phase model reduction and system identification, which both seek to further bridge the gap between the two disciplines.

1.6 Permissions and Attributions

1. Portions of Chapter 5 were produced in collaboration with Andrew Yates, with select figures courtesy of *Extraction of Phase and Isostable Response Curves via Deep Neural Networks* (Yates, Matchen, and Moehlis, *in preparation*).

Chapter 2

A Starting Point: Oscillator Control Design in a Classical Setting

The bulk of this dissertation will focus on the implementation of machine learning in dynamics and control to overcome the challenges and limitations of more traditional approaches. It would be remiss, however, to do so without first contextualizing the unique strengths and weaknesses of non-machine learning approaches and understanding how ML may be integrated with these strategies. To this end, we begin with an analysis of a control strategy for identical oscillators (specifically, in this example, neurons), with the goal of achieving partial desynchronization to a clustered state, where neurons are locally in phase with other neurons but out of phase with other groups of neurons. We consider as a starting point the case of identical, uncoupled neurons subject to a common input; such a scenario, with sufficiently small input amplitudes, is an ideal candidate for the phase model reduction technique described in Chapter 1. The material in this chapter has been published previously in [45] and [46].

2.1 Stabilizability of Clusters

2.1.1 Terminology and Overview

Before describing the specific control design we will employ, we start by showing that it is in general feasible to achieve a clustered state with a population of identical neurons. In particular, we demonstrate that any order-preserving clustering scheme for uncoupled, identical neurons is asymptotically stabilizable with an appropriate control input provided minor restrictions are placed on the phase response curve. Here we understand asymptotically stabilizable to mean that, for an appropriate choice of input u, the system of neurons approaches our desired state as $t \to \infty$. To do this, it must be shown that the control system is *passive* with a *radially unbounded positive definite storage function* and *zero-state observable* [47]. These requirements are summarized as follows:

Radially Unbounded Positive Definite Storage Function

A storage function $\mathcal{V}$ is any function that converts the state of the system into a scalar measure of the "energy" stored in the system; a simple physical example might be a function converting the position and velocity of an object into a total energy consisting of potential and kinetic energy components. In this case, we desire the storage function to equal 0 at exactly one point: our target state. The requirement that the function is positive definite means that everywhere else in the state space, the storage function's value is greater than 0. Because the function is radially unbounded, we further require that as we move farther from this state, the value continually increases.

Passive

A system is passive [47] if, for a given observable vector y, storage function $\mathcal{V}$, and any choice of u:

$$u^T y \geq \dot{\mathcal{V}}, \tag{2.1}$$

where $\dot{\mathcal{V}}$ denotes the first time derivative of $\mathcal{V}$. This requires, for example, that if $u^T = \vec{0}$, $\dot{\mathcal{V}} \leq 0$. Physically, this corresponds to the system not producing energy and instead being energy neutral or an energy consumer.

Zero-State Observable

For a vector observable y to be zero-state observable, the target state must be the only point in state-space where $y = \vec{0}$ and remains zero for all future times. Although y may equal zero at other points in state-space, it must become nonzero in finite time. For example, if we used height as our observable for a bouncing ball, we would say the system is zero-state observable because, unless the velocity of the ball stays at zero (i.e., the ball has stopped bouncing), the height will not remain zero (the ball will bounce back up).

2.1.2 Derivation of Stabilizability

A system that meets these three criteria can be shown to be stabilizable with an appropriate choice of input [47]. We demonstrate these requirements all generally hold for the case of N identical, uncoupled neurons in the reduced phase model formulation. We label the neurons such that, at time $t = 0$, the neuron phases are ordered as $\theta_1 < \theta_2 < \theta_3 < ... < \theta_N$. Note that if the phases of two neurons are exactly the same, because the neurons are identical and receive identical inputs, they are impossible to separate; therefore, we exclude the possibility of two phases being equal by assumption.

Furthermore, because the neurons are identical, the response of a neuron is bounded by the neurons of phase initially less than the neuron and those greater than the neuron, so for $t > 0$, it follows from these assumptions that $\theta_1(t) < \theta_2(t) < ... < \theta_N(t)$ (here we do not use the modulo 2π value for θ_j, so θ_j is allowed to be greater than 2π) [7].

Typically when discussing stabilizability, the target state would be a specific coordinate in state-space, such as the origin. Here, however, we do not want the neurons to stop oscillating, so we do not wish to drive the system to specific values of θ. Instead, we wish to instead reach a target state describing the relations between their phases as they continue to oscillate. It is therefore natural to define our storage function in terms of the *differences* between the phases of neurons rather than the individual phases (which are constantly evolving). More precisely, we construct our storage function as the linear combination of positive semidefinite functions, each prescribing the target separation for the phases of two neurons:

$$v_i = v_i(\theta_j - \theta_k), \qquad \mathcal{V}(\theta_1, ..., \theta_N) = \sum_{i=1}^{l} \beta_i v_i, \tag{2.2}$$

with $\beta_i > 0$ and where θ_j and θ_k are the phases of any two neurons whose separation is to be prescribed by the function v_i. The value of l is arbitrary in this context; in Section 2.2, for the specific problem of clustering $l = K$. The individual storage function candidates have three properties:

1. At the target separation $\theta_j - \theta_k = \Delta\theta^*$, $v_i(\Delta\theta^*) = 0$;
2. For $\theta_j - \theta_k \neq \Delta\theta^*$, $v_i(\theta_j - \theta_k) > 0$ and grows unbounded away from $\Delta\theta^*$ within the interval $\theta_j - \theta_k \in (0, 2\pi)$; and
3. $\left.\frac{\partial v_i}{\partial \Delta\theta}\right|_{\Delta\theta^*} = 0$, $\left.\frac{\partial v_i}{\partial \Delta\theta}\right|_{\Delta\theta \neq \Delta\theta^*} \neq 0$.

Fig. 2.1 illustrates the case of $l = 2$, $\Delta\theta^* = \pi$ to demonstrate how these control objectives translate into a stabilized clustered configuration.

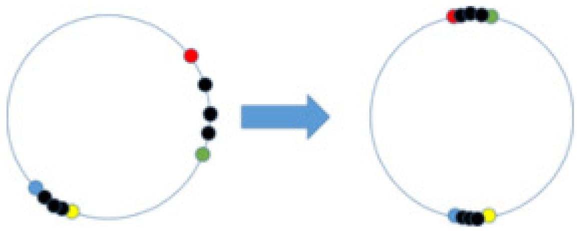


Figure 2.1: Visualization of the control objective design. Each circle represents a neuron, and they oscillate around the unit circle. Here, $l = 2$, so two target separations are specified: the separation between the yellow and green neurons and the separation between the blue and red neurons. $\mathcal{V} = 0$ if and only if both of these separations are π (as is nearly the case in the circle to the right). Because the neurons are identical, the positions of the black neurons are bounded by the non-black neurons, and so they are guaranteed to be present in this clustered arrangement.

We now calculate the value of $\dot{V}$. As each individual storage function is dependent on only one phase difference, we write $\dot{V}$ as:

$$\dot{V} = \sum_{i=1}^{l} \beta_i \frac{\partial v_i}{\partial \Delta\theta_i} \dot{\Delta\theta_i} = \sum_{i=1}^{l} \beta_i \frac{\partial v_i}{\partial \Delta\theta_i} \left(\dot{\theta}_j - \dot{\theta}_k \right). \tag{2.3}$$

Substituting in from (1.1), $\dot{V}$ can be rewritten as:

$$\dot{V} = u^* \sum_{i=1}^{l} \beta_i \frac{\partial v_i}{\partial \Delta\theta_i} \left(Z\left(\theta_j\right) - Z\left(\theta_k\right) \right), \tag{2.4}$$

where u^* is the common input received by every neuron. To satisfy passivity, we choose our observable to be a vector $y = [y_1, y_2, \cdots, y_l]^T$ such that:

$$y_i = \beta_i \frac{\partial v_i}{\partial \Delta\theta_i} \left(Z\left(\theta_j\right) - Z\left(\theta_k\right) \right). \tag{2.5}$$

Recognizing identical inputs as a special case of $u^T = [u_1, u_2, \cdots, u_l]$ where $u_i = u^* \forall i$, it follows that $u^T y = \dot{V}$ everywhere in the state-space. Therefore, the system as constructed is not only passive but also lossless. Additionally, y is zero-state observable: at the target state, $\frac{\partial v_i}{\partial \Delta\theta_i} = 0$; otherwise $y = 0$ only if $Z(\theta_j) - Z(\theta_k) = 0$, but no such pair of neurons can stay indefinitely in the set $y = 0$. We can see this by considering:

$$\frac{d}{dt}\left(Z(\theta_j) - Z(\theta_k)\right) = \left.\frac{\partial Z}{\partial \theta}\right|_{\theta_j} \dot{\theta}_j - \left.\frac{\partial Z}{\partial \theta}\right|_{\theta_k} \dot{\theta}_k. \tag{2.6}$$

Because $Z(\theta_j) = Z(\theta_k)$, it follows from (1.1) that $\dot{\theta}_j = \dot{\theta}_k$ instantaneously, so the right side of (2.6) equalling 0 would require $\left.\frac{\partial Z}{\partial \theta}\right|_{\theta_j} = \left.\frac{\partial Z}{\partial \theta}\right|_{\theta_k}$. For y to equal 0 at all future times, this would further imply that this equality must hold over the entire period, i.e. $\exists \delta x \in (0, 2\pi)$ such that $\left.\frac{\partial Z}{\partial \theta}\right|_{x} = \left.\frac{\partial Z}{\partial \theta}\right|_{x+\delta x}$ $\forall x$. Graphically, this would mean that horizontally shifting the phase response curve reproduces the original curve. Because $\left.\frac{\partial Z}{\partial \theta}\right|_{0} = \left.\frac{\partial Z}{\partial \theta}\right|_{2\pi}$ and $Z(0) = Z(2\pi)$, this is true if and only if $Z(\theta)$ is constant or has periodicity greater than 2π, which is physically not realized. Therefore, as the system is both passive with an unbounded storage function and zero-state observable, we can conclude that the system can be stabilized by the choice of $u = -\phi(y)$ where $\phi(y)$ is locally Lipschitz and $y\phi(y) > 0$ [47]. We note that this does not strictly hold for the case of an identical input; while $u^T = [u, u, \cdots, u]$ does allow for locally Lipschitz solutions, there is a measure-0 set in which $y\phi(y) = 0$. In practice, however, we find this only forms an invariant set when the phase response curve is in some way degenerate or not physically realizable (such as having a higher than 2π periodicity) or the control objectives are poorly defined (such as when reaching the control state would require neurons to cross each other). Other instances of $y\phi(y) = 0$ are solely instantaneous and do not affect the computational outcome; we expand on the additional circumstances in which this may occur and demonstrate they are not zero-state observable in Appendix B.

2.2 Control Strategy for K Clusters of Neurons

Having shown that clustered states for identical neurons can be stabilized, we will now outline our design strategy for doing so. Our goals in developing a control strategy for clustering are threefold:

1. Create a flexible method such that the strategy functions in a way that is agnostic both to the specific neuron model used and the desired number of clusters K;
2. Require as little precomputing as possible so the method is robust to inaccuracies in modeling; and
3. Allow for the control to be easily tuned for parameters of interest, such as maximum input amplitude and the speed with which clustering is achieved.

These three conditions can be seen as measures of robustness for the method. A control scheme that meets these three criteria can be altered on the fly by changing only a small number of target parameters, allowing the input to rapidly be tuned to the performance specifications desired. Additionally, deviations from expected results can be compensated for if the input is not constrained to precomputed values, as would be the case with optimal control strategies derived from, for example, variational principles.

The approach proposed here consists of considering what we propose to call the input of maximal instantaneous efficiency (IMIE) rather than precomputed data. Although not necessarily as efficient as true optimization strategies, IMIE requires only knowledge of the phase response curve of the neurons and the current state of the system.

The rest of this section will be structured as follows: first, we will define the two necessary functions for IMIE: a state function and a cost function. Next, we will lay out the details of the control strategy. Lastly, we will see how the reduction of the model for special cases returns results that agree with intuition and past results.

State Function

The state function r, to be defined below, is functionally equivalent to a specific storage function that we will use to generate our control. Control of a system of N neurons into K clusters requires the direct control of $2K$ neurons, split into pairs, with each pair of neurons adjacent to each other in phase order. The control is generated in such a way that each pair is driven apart to a target separation of $\frac{2\pi}{K}$ radians. In this way, K clusters are formed by exploiting the boundedness of response described in Section 2.1 and illustrated in Fig. 2.1. For example, if we wished to subdivide a population of 16 identical neurons into 4 clusters and the neurons were ordered by initial phase ($\theta_1 < \theta_2 < ... < \theta_{16}$), the K control pairs would be {2, 3}, {6, 7}, {10, 11}, and {14, 15}, and the final clusters would be {15, 16, 1, 2}, {3, 4, 5, 6}, {7, 8, 9, 10}, and {11, 12, 13, 14}. We define a positive semidefinite function $r_{i,j}$ for each control pair; this function is dependent only on the phase difference $\Delta\theta_{i,j} = \theta_j - \theta_i$ and is identically zero at $\Delta\theta_{i,j} = \frac{2\pi}{K}$.

To allow for consistency in the definition of $r_{i,j}$ across choices of K, the value of $\Delta\theta_{i,j}$ is mapped by the function $g\left(\Delta\theta_{i,j}\right)$ so that $g\left(\frac{2\pi}{K}\right) = \pi$. This is done using the piecewise definition:

$$g\left(\Delta\theta_{i,j}\right) = \begin{cases} \frac{K}{2}\Delta\theta_{i,j} & 0 \leq \Delta\theta_{i,j} \leq \frac{2\pi}{K} \\ K\frac{(\Delta\theta - 2\pi/K)}{(2(K-1))} + \pi & \frac{2\pi}{K} < \Delta\theta_{i,j} \leq 2\pi \end{cases}. \tag{2.7}$$

With this mapping, we define the positive-definite function for each pair as follows:

$$r_{i,j} = \begin{cases} \frac{1}{g(\Delta\theta_{i,j})^p} - \frac{1}{\pi^p} & 0 < \Delta\theta_{i,j} \leq \frac{2\pi}{K} \\ \frac{1}{(2\pi - g(\Delta\theta_{i,j}))^p} - \frac{1}{\pi^p} & \frac{2\pi}{K} < \Delta\theta_{i,j} < 2\pi \end{cases}, \tag{2.8}$$

which is continuous and differentiable everywhere on the domain $(0, 2\pi)$ except at $\Delta\theta_{i,j} = \frac{2\pi}{K}$. The value of the parameter p can be adjusted to meet control objectives; in the simulations in Section IV, $p = 0.7$. The function $r_{i,j}$ can be made first-order differentiable by the replacement of the constant $\frac{1}{\pi^p}$ with a term that is linear in g, though in practice this is not necessary. This replacement generates a function that does, however, serve as a valid storage function candidate in (2.2), while maintaining derivatives with the same sign as in (2.8). Clearly, (2.8) is greater than zero for all choices of p with $\Delta\theta_{i,j} \neq \frac{2\pi}{K}$ and grows unbounded as $\Delta\theta_{i,j} \to 0$ or 2π.

From here we can define a state function of the system as:

$$r = \frac{1}{K}\sum_{l=1}^{K} r_{2l-1,2l}. \tag{2.9}$$

Note that here we have omitted the neurons that are not being directly controlled, and as such our control pairs are relabelled as {1,2}, {3,4},...,{$2K-1$,$2K$}. Because each component of the summation is greater than zero everywhere except at the desired target state, the combined function is also positive-definite and only equal to zero when all pairs of neurons achieve the target separation.

Cost Function

With the state function defined, we turn our attention to the cost function. The purpose of the cost function is to prescribe the important characteristics of the control by penalizing undesired behavior. While any cost function can be used, we select one that penalizes energy usage and the time required to reach the target state. This can be accomplished by defining the cost function:

$$C(t) = \int_0^t \left[u(\tau)^2 + \alpha r(\tau)\right] \mathrm{d}\tau. \tag{2.10}$$

The first term introduces a quadratic cost to increased input amplitude; this is inspired by the general fact that the square of the input amplitude is proportional to power. The second term penalizes the value of r being large. The value of α can be adjusted to increase or decrease the relative importance of this penalty; the higher the value of α, the greater emphasis the control places on reducing the value of the state function quickly. The instantaneous cost associated with the state and input at a given time t can be given by taking the derivative and evaluating:

$$\frac{dC}{dt} = u\left(t\right)^2 + \alpha r\left(t\right). \tag{2.11}$$

With the state and cost functions defined, the input of maximal instantaneous efficiency can be generated as follows. An optimal path is one that minimizes $C\left(t\right)$ as $t \to \infty$. While the time-dependent input would need to be computed in advance to truly optimize, IMIE aims to produce a near-optimal input by minimizing the cost incurred at each time step instead. We rewrite $C\left(t\right)$ in terms of the value of r, which in the uncoupled case monotonically decreases at all times with the appropriate choice of u. Then the total cost as $t \to \infty$ is equal to:

$$\lim_{t\to\infty} C\left(t\right) = \int_{r(0)}^{0} \frac{dC}{dr} \mathrm{d}r; \tag{2.12}$$

by exploiting the chain rule, we equate $\frac{dC}{dr}$ to:

$$\frac{dC}{dr} = \frac{{}^{dC}\!/_{dt}}{{}^{dr}\!/_{dt}}. \tag{2.13}$$

In this formulation, the input we choose is designed so that, at all times, the instantaneous magnitude of $\frac{dC}{dr}$ is minimized. This can be interpreted as the input that is most efficient in terms of cost relative to change in r. The value of $\frac{dC}{dt}$ is given by (2.11). Differentiating

r with respect to time yields:

$$\frac{dr}{dt} = \sum_{l=1}^{2K} \frac{\partial r}{\partial \theta_l} \frac{d\theta_l}{dt}. \tag{2.14}$$

As in (2.4), we can reorganize this and exploit the fact that in each neuron phase control pair $\theta_{2k}, \theta_{2k+1}$ the partial derivatives with respect to phase satisfy $\frac{\partial r}{\partial \theta_{2k}} = -\frac{\partial r}{\partial \theta_{2k+1}}$ and express $\frac{dr}{dt}$ as:

$$\frac{dr}{dt} = u \sum_{l=1}^{K} \frac{\partial r}{\partial \theta_{2l}} \left(Z\left(\theta_{2l}\right) - Z\left(\theta_{2l-1}\right) \right), \tag{2.15}$$

which has the characteristic form $-a\left(\theta_1, ..., \theta_{2K}\right) u\left(t\right)$. Therefore, $\frac{dC}{dr}$ is equal to:

$$\frac{dC}{dr} = -\frac{u^2 + \alpha r}{au}, \tag{2.16}$$

where the dependence of a and r on the state $\left[\theta_1, ..., \theta_{2K}\right]$ is omitted from the equation for simplicity.

From this, the extrema can be found by differentiating with respect to u; the input used is then set equal to this calculated minimum. Differentiating and rearranging yields:

$$u\left(t\right) = \frac{\sqrt{a^2 \alpha r\left(t\right)}}{a} = \mathrm{sign}(a)\sqrt{\alpha r\left(t\right)}. \tag{2.17}$$

Note here the positive root is taken because $\frac{dC}{dr}$ is negative (since $\frac{dC}{dt}$ is always positive and $\frac{dr}{dt}$ is negative by construction), and therefore the quantity au must be positive for the entire expression to be negative.

Recalling the definition of $r\left(t\right)$, u evolves as a function of the average separation $\overline{\Delta\theta}$ of the control pairs as approximately $\overline{\Delta\theta}^{-p/2}\sqrt{\alpha}$. From this it can be seen that, holding p constant, increasing α corresponds to a $\sqrt{\alpha}$ increase of the maximum amplitude of the input signal. This in turn decreases the response time of the system at the cost, generally, of increasing total power usage and maximum input amplitude. In contrast, increasing the value of p while holding maximum amplitude constant (by adjusting α accordingly) will cause a sharper decline in the input signal, reducing power usage but increasing response time. As such, the system can be tuned to meet the desired control specifications – power usage, maximum amplitude, response time – simply by varying α and p accordingly, regardless of the neuronal model being used.

We now turn our attention to the case where $\alpha = 0$ and demonstrate that the method returns a result that is consistent with intuition. In this case, no weight is placed on fast response time, and the cost is entirely connected to minimizing the energy usage of the system. With $\alpha = 0$, the original formulation of $\frac{dC}{dr}$ can be simplified greatly, yielding:

$$\frac{dC}{dr} = \frac{-u}{a}. \tag{2.18}$$

Unlike the case where $\alpha \neq 0$, this is linear and therefore has no minimum; since the only constraint is that $\frac{-u}{a}$ should be positive, a lower-cost control is always achieved by decreasing the magnitude of u. It can be seen that, as predicted by this result, using constant-amplitude control takes longer (but requires less energy) when a smaller amplitude is used, thereby agreeing that the optimal control from an energy perspective is to use as small an input as possible.

In practice, we do not want the state to be reached in infinite time. If we abstract away from the physical representations of the phase model (which breaks down at high amplitudes of u) and consider only what will allow us to reach the target state in as little time as possible, we would expect that the solution would be to allow the input signal to be as large as possible for all times. We can model this by removing u^2 from the cost function so that $C^*(t) = \alpha r(t)$. Now, $\frac{dC}{dr}$ is given as:

$$\frac{dC}{dr} = \frac{-\alpha r}{au}. \tag{2.19}$$

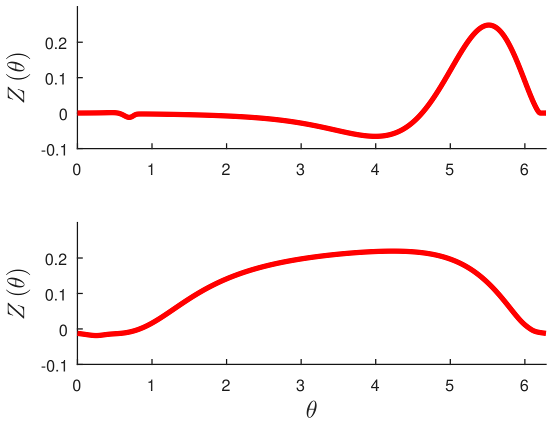


Figure 2.2: Phase response curves for the reduced Hodgkin-Huxley equations (*top*) and the thalamic model (*bottom*)

Table 2.1: Default Simulation Parameters

N	50
K	4
α	0.1
p	0.7

As in the case where $\alpha = 0$, this has no minimum, and instead approaches 0 as $u \to \infty$. Therefore, IMIE correctly predicts that for the fastest possible response, u should be allowed to be as large as allowed by the constraints on the system at all times. This trend, as well as the minimal-energy trend, are demonstrated in simulation and shown as solid lines in Fig. 2.5. While we do not propose IMIE as a fully optimal control strategy, this demonstrates that the method matches basic sanity checks in its application.

2.2.1 Application to Uncoupled Identical Neurons

We now apply the IMIE approach to two different neural models: a two-dimensional reduced Hodgkin-Huxley model [48, 49] and a three-dimensional model for periodically firing thalamic neurons [33], both presented in Appendix A. Unless otherwise stated, all simulations we present in this section utilized the parameters listed in Table 2.1. The dynamics for both models were initially represented using the phase model reduction, with all the neurons treated as identical (possessing the same natural frequencies and phase response curves). The dynamics for all neurons were given by:

$$\dot{\theta}(t) = \omega + Z(\theta)\, u(t)\,, \tag{2.20}$$

where the natural frequencies and phase response curves are appropriate to the models. The phase response curves for the two models were derived from the adjoint equation using XPPAUT and can be seen in Fig. 2.2. In addition to the distinctions between the initial dimensionality of the two models, they also differ in that the Hodgkin-Huxley model is an example of a Type II neuron, whereas the thalamic model is representative of a Type I neuron [50]. This can be seen by the qualitative differences in their phase response curves: whereas Type II neurons have PRCs with positive and negative portions, Type I neurons have PRCs that are typically nonnegative [51].

By using models of two different types of periodically firing neurons, we aim to show that the qualitative results of this control scheme are similar across qualitatively different base models. This agreement can be seen in Fig. 2.3, which shows the response of a population of neurons of each type to the proposed control strategy.

Fig. 2.4 shows in finer detail the control signal applied to the Hodgkin-Huxley phase model to achieve clustering. Initially, when the neurons are still in a one-cluster configuration, the signal varies comparatively rapidly, both in amplitude and direction. However, as the system approaches the 4-cluster state, the signal becomes increasingly regular with a frequency four times that of the system's natural frequency and with a signal that only slowly decreases in amplitude. We view these "maintenance" signals when the system is near the clustered state as especially feasible with current hardware. The results for the thalamic model are qualitatively equivalent to those presented here.

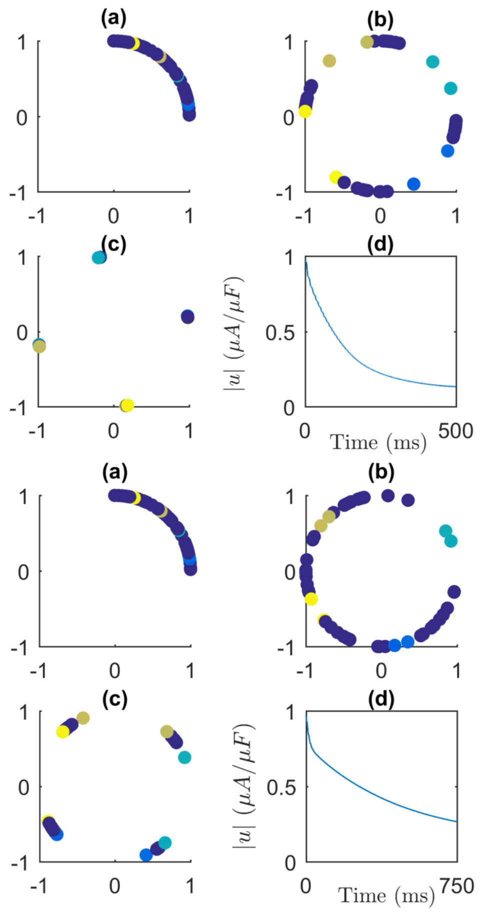


Figure 2.3: Evolution of reduced Hodgkin-Huxley (*top*) and thalamic (*bottom*) phase models at three times for $K = 4$. **(a)**, **(b)**, and **(c)** show the projection of the phases onto the unit circle at times $t = 0$, $t = 125$, and $t = 500$ ms (0, 187.5, and 750 for thalamus), respectively. **(d)** shows the absolute value of the input over the length of the simulation. The differently-colored pairs represent neurons that are being actively controlled.

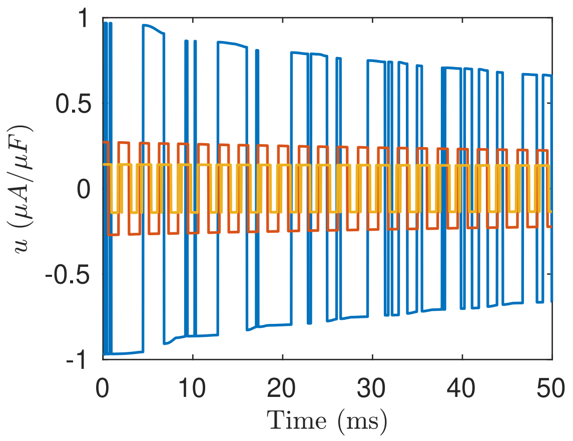


Figure 2.4: Control signal over three time intervals, Hodgkin-Huxley model. The actual values of the signal (in contrast to the absolute values presented in Fig. 2.6) are presented over three 50-ms time intervals. The blue interval begins at $t = 0$ ms, the red at $t = 200$ ms, and the yellow at $t = 450$ ms. We note that the signal becomes increasingly regular as the system approaches a clustered state.

Furthermore, IMIE improves upon the performance of strategies that similarly require only instantaneous state data to calculate a control. Using the same methodology of selecting whether to apply a positive or negative input based on the derivative of the state function r, the Hodgkin-Huxley neuron population was also simulated with the application of a constant-amplitude "bang-bang" control. In "bang-bang" control, the amplitude of the control signal is fixed while its direction (positive or negative) is allowed to switch. For a range of amplitudes, the constant-amplitude control and IMIE were simulated until the values of their state functions were within a tolerance of a 0 ($r_{tol} = 0.01$). The "settling" time and value of $\int_0^T u^2 dt$ was recorded for each trial; these results are shown in Fig. 2.5. For comparison between the two methods, in these simulations p was held constant and α was varied such that the initial value of u was equal to the listed max amplitude ($\alpha = \frac{u_{max}^2}{r(0)}$). Without an appreciable sacrifice of response time, IMIE achieves clustering at a dramatically reduced energy cost when compared to bang-bang control.

With the effectiveness of the control established for phase models, the control was

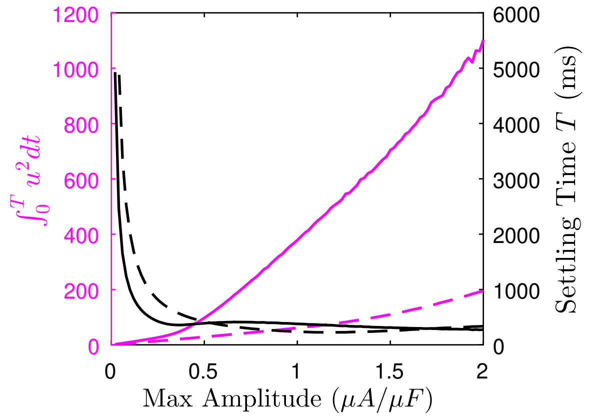


Figure 2.5: Comparison of constant-amplitude control and IMIE. The solid lines denote the constant-amplitude control. The dashed lines represent the implementation of IMIE on the population of uncoupled, identical Hodgkin-Huxley neurons reduced to a phase model for control to the 4-cluster state. Note that the energy cost (in red) is always lower for IMIE than the analogous constant-amplitude control, and despite these massive energy discrepancies the response time (in black) is always either better or approximately equal to that of constant-amplitude control, except in the extreme low-amplitude case. This demonstrates that IMIE represents a performance improvement over bang-bang control in terms of both energy cost and response time.

extended and applied to a full state model for both the Hodgkin-Huxley and thalamic neurons. The value of the state function $r(t)$ was calculated based on an estimation of the phase corresponding to a neuron's position in state-space. This rough approximation of the phase was found by identifying the point on the curve nearest to the neuron's position in state-space. Because the variation in V differs far more significantly than the variation in n or other gating variables, the distance to the curve was normalized in each dimension by the span of the limit cycle in that dimension. Example simulations for these full state-space models can be seen in Fig. 2.6. Note that there is a "jitter" in (d) in both plots; this can be attributed to inaccuracies inherent in estimating the phase of the neurons, which can result in large movement in the value of $r(t)$ because of its large derivative at small values of $\Delta\theta$.

Additionally, adjusting the values of p and α can alter the response characteristics of the populations of neurons in a consistent fashion. To illustrate this, a “settling time” and energy cost were calculated for different pairs of parameter values. Because p was varied between trials, to maintain a consistent benchmark of performance the system was simulated not to a specific value of r but rather to a specific average deflection from the target state. $\Delta\theta_{tol}$ was defined as:

$$\Delta\theta_{tol} = \frac{9}{10}\Delta\theta^*, \tag{2.21}$$

which subsequently was used to define a specific value of r_{tol} dependent on the value of p as:

$$r_{tol} = \frac{1}{g\left(\Delta\theta_{tol}\right)^p} - \frac{1}{\pi^p}. \tag{2.22}$$

The settling time T was defined as the time to reach r_{tol}, and the energy cost was calculated as $\int_0^T u^2\left(t\right)dt$.

The effect of variations in p and α on the phase model results can be seen in Fig. 2.7. As can be expected, for a given value of p, increasing α leads to a decreased response time but increased energy cost. Decreasing p corresponds to the input decreasing less sharply in time; despite a steeper rate of decrease, however, between 0 and $\Delta\theta^*$ the value of r increases for a given $\Delta\theta$ as p increases. Therefore, increasing p corresponds to an increase in energy usage as well as a decrease in response time.

In practice, we may wish to prescribe a maximum amplitude for the control signal rather than just utilizing a scaling value of α arbitrarily. In the uncoupled phase model, we assume that r decreases at all times; from this, it follows that $|u|_{max} = |u(0)|$, as a decrease in r corresponds to a decrease in $|u|$. If, instead of varying α, $|u|_{max}$ is varied instead, the trend from Fig. 2.7 reverses. Here, the value of α is prescribed by its

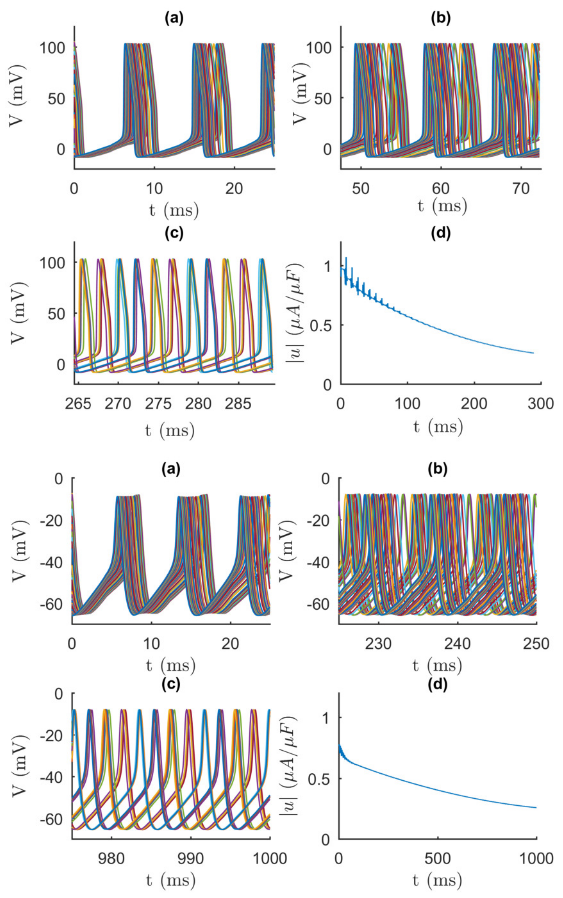


Figure 2.6: Evolution of reduced Hodgkin-Huxley (*top*) and thalamic (*bottom*) state-space models over three time intervals. **(a)**, **(b)**, and **(c)** show three 25-millisecond voltage traces of the 50 neurons being simulated subject to the described control without coupling. As in the phase model simulations, the neurons initially become desynchronized, spreading out around the limit cycle before coalescing into clusters. **(d)** shows the absolute value of the input over the duration of the simulation, with general qualitative agreement to the plots in Fig. 2.3.

relationship to u and r:

$$\sqrt{\alpha r\left(0\right)} = |u|_{max} \qquad \Rightarrow \qquad \alpha = \frac{{u_{max}}^2}{r\left(0\right)}, \tag{2.23}$$

where r is dependent on the choice of p. For a given maximum amplitude, an increase in the value of p now corresponds to a state function whose value drops more sharply, which in turn means an input whose magnitude drops more sharply. As such, energy usage decreases with increased p for a fixed u_{max} and response time increases in turn. This can be seen in Fig. 2.8.

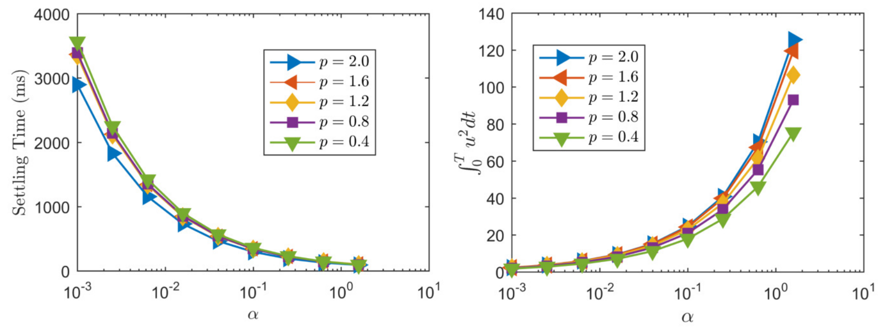


Figure 2.7: Effect of variations in p and α. The effect of varying the two parameters on settling time (with r_{tol} set by the value of p) is shown in the *left* panel, while the integral of u^2 is shown to the *right*. Each curve represents a constant value of p, with α allowed to vary. These results maintain the correspondence between increasing energy cost and decreasing response time illustrated in Fig. 2.5.

2.2.2 Application to Coupled Identical Neurons

We now introduce coupling to the phase model reduction and adapt IMIE to accommodate its effects. We consider all-to-all electrotonic coupling; in the state-space model, this is introduced into the value of $\dot{V}$ as [52]:

$$\Delta \dot{V}_i = a_e \sum_{j=1}^{N} \left(V_j - V_i\right). \tag{2.24}$$

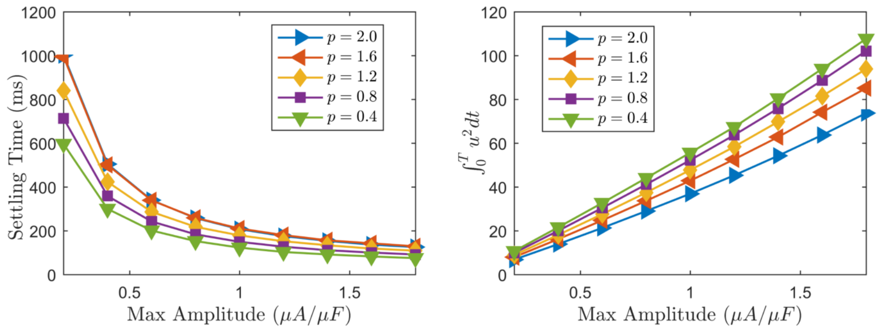


Figure 2.8: Effect of variations in p and $|u|_{max}$. For a given maximum amplitude, increasing p leads to an increase in settling time (*left*) but a decrease in energy usage (*right*).

In the context of the phase model reduction, this can be rewritten as:

$$\dot{\theta}_i = \omega + Z(\theta_i) u + a_e \sum_{j=1}^{N} \left(V(\theta_j) - V(\theta_i)\right) Z(\theta_i). \tag{2.25}$$

Here, a_e is the strength of the electrotonic coupling. This, however, is not a particularly convenient notation, as we would like to omit V from the phase model reduction. Assuming a_e is small (coupling is weak), we can further simplify this equation by selectively averaging the value of the coupling over one period [53], $\bar{f}(\theta_j - \theta_i) = \bar{f}(\Delta\theta_{i,j})$. This can be calculated as:

$$\bar{f}(\Delta\theta_{i,j}) = \frac{1}{2\pi}\int_0^{2\pi} \left[\left(V(\theta_j + \Theta) - V(\theta_i + \Theta)\right) Z(\theta_i + \Theta)\right] d\Theta; \tag{2.26}$$

$\dot{\theta}_i$ can then be rewritten in terms of this averaged function instead.

Returning to (2.14), the expression for $\frac{d\theta_l}{dt}$ has changed, and (2.15) must include an additional term; $\frac{dr}{dt}$ is now computed as:

$$\frac{dr}{dt} = \sum_{l=1}^{K} \frac{\partial r}{\partial \theta_{2l}} \left\{ u\left(Z(\theta_{2l}) - Z(\theta_{2l-1})\right) + a_e \sum_{m=1}^{N} \left[\bar{f}(\theta_m - \theta_{2l}) - \bar{f}(\theta_m - \theta_{2l-1})\right] \right\}, \tag{2.27}$$

which now has the characteristic form $-au + b$ instead of simply $-au$. Rederiving the instantaneously optimal input yields:

$$u(t) = \frac{b + \sqrt{b^2 + a^2 \alpha r}}{a}. \tag{2.28}$$

The problem presented by coupling is that a core assumption of the control design (namely, that $\frac{dr}{dt}$ monotonically decreases with the appropriate choice of sign for u) no longer holds; there is a measure-0 probability that the value of a is identically 0 at some time. If a approaches 0, the mandate that $\frac{dr}{dt}$ monotonically decrease proves excessively restrictive: the value of u grows unbounded, which both violates the regime in which the phase model reduction can be considered valid and runs counter to the goal of a cost-minimized strategy. To address this, we instead approach the problem by considering an averaged value of $\frac{dr}{dt}$, much as was done for the coupling function:

$$\begin{aligned} \frac{dr}{dt}_{avg} = \frac{1}{2\pi} \int_0^{2\pi} \sum_{l=1}^{K} \frac{\partial r}{\partial \theta_{2l}} \Big\{ & u\left(Z\left(\theta_{2l} + \Theta\right) - Z\left(\theta_{2l-1} + \Theta\right)\right) \\ & + a_e \sum_{m=1}^{N} \left[\bar{f}\left(\theta_m \Theta - \theta_{2l} - \Theta\right) - \bar{f}\left(\theta_m + \Theta - \theta_{2l-1} - \Theta\right) \right] \Big\} d\Theta. \end{aligned} \tag{2.29}$$

Rearranging and recognizing that $\frac{1}{2\pi}\int_0^{2\pi} \bar{f}\left(\Delta\theta\right) d\Theta = \bar{f}\left(\Delta\theta\right)$, we simplify as:

$$\frac{dr}{dt}_{avg} = -\bar{a}u + a_e \sum_{l=1}^{K} \frac{\partial r}{\partial \theta_{2l}} \sum_{m=1}^{N} \left[\bar{f}\left(\theta_m - \theta_{2l}\right) - \bar{f}\left(\theta_m - \theta_{2l-1}\right) \right], \tag{2.30}$$

where:

$$\bar{a} = \frac{-1}{2\pi} \int_0^{2\pi} \left| \sum_{l=1}^{K} \frac{\partial r}{\partial \theta_{2l}} \left(Z\left(\theta_{2l} + \Theta\right) - Z\left(\theta_{2l-1} + \Theta\right)\right) \right| d\Theta. \tag{2.31}$$

The absolute value is taken to reflect the ability to choose at each time increment the appropriate sign of u, a feature that is otherwise not captured by this equation. The sign of u is still computed from the unaveraged value of a. As such, this equation still reduces to (2.17) in the limit $b \to 0$.

Additionally, we can consider special cases that arise with the introduction of coupling and ensure that our intuition hold for these circumstances. We consider three cases: the limit where α is large compared to a and b, the limiting behavior as $a^2\alpha r \to 0$, and the resulting equation for $\alpha = 0$. If α is large, the magnitude of the input will similarly be large as the control objective will consist of approaching $r(t) = 0$ as quickly as possible. In this case, $|u| \to \left|\frac{b}{\bar{a}} + \text{sign}(a)\sqrt{\alpha r}\right|$ when r is large. We include the absolute value on u to acknowledge the influence of a on the instantaneous sign of u. In this case only a relatively small correction term is applied to the control for the $b = 0$ case (amounting to, on average, matching the strength of the coupling), as the control dynamics are dominated by α instead of b. As we approach the target state, however, $a^2\alpha r \to 0$ regardless of the choice of α, and the dynamics become instead dominated by the coupling; in this limit $|u| \to \left|\frac{b+|b|}{\bar{a}}\right|$.

This limiting behavior would be demonstrative of the response if our response time was irrelevant and energy was the only consideration, i.e. if $\alpha = 0$. In that case, the response at all times would evolve as $|u| = \left|\frac{b+|b|}{\bar{a}}\right|$. When $b < 0$, coupling serves to pull the system toward the target state; in the presence of this favorable coupling, $u = 0$ as the most energy-efficient strategy is to input no additional energy. If $b > 0$, then $|u| = \left|\frac{2b}{a}\right|$. Whereas in the uncoupled case the optimal control input when $\alpha = 0$ is to asymptotically approach a magnitude of 0, the presence of coupling allows for the presence of a local minimum instead with a value slightly higher than the averaged offset value of $\frac{b}{a}$.

Simulations with weak coupling (here, $a_e = 0.01$) were conducted for both the Hodgkin-Huxley and thalamic models. Results for the coupled state-space model can be seen in Fig. 2.9. Note that while the general shape of the input signal is consistent with the uncoupled case, the presence of weak coupling in addition to the uncertainty

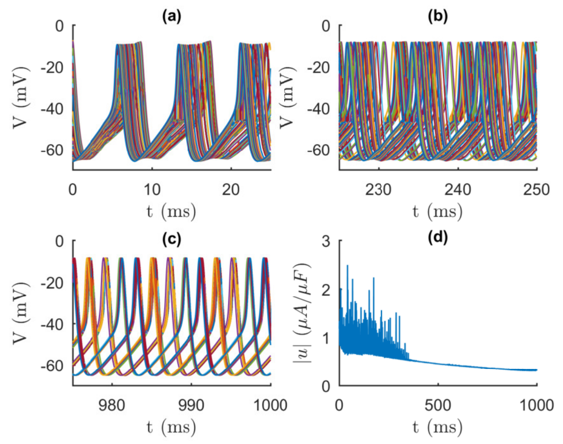


Figure 2.9: Simulation of thalamic state-space model with weak ($a_e = 0.01$) coupling for $K = 4$. Despite large fluctuations in the amplitude of the control signal, clustering proceeds similarly to in the uncoupled case: first, the system desynchronizes, and then strong clustering emerges.

Table 2.2: Energy Cost to Reach K-Cluster State

	$K = 2$	$K = 3$	$K = 4$	$K = 5$	$K = 6$
Hodgkin-Huxley	13.8691	14.1803	18.1086	23.2431	29.2406
Hodgkin-Huxley ($a_e = 0.01$)	10.7016	19.1343	27.2689	32.4005	35.6388
Thalamic	29.064	71.7944	70.1354	69.6513	92.1052
Thalamic ($a_e = 0.01$)	24.337	76.6713	82.9656	78.5948	100.763

caused by the phase estimation leads to significant fluctuations in the amplitude of the control signal. Despite these fluctuations, the control scheme still is able to successfully cluster the population of neurons at a relatively low cost, albeit higher than in the uncoupled case.

As can be seen by comparing Figs. 2.6 and 2.9, weak coupling does not significantly affect the response time of the system, only the energy cost. To demonstrate this, the population of neurons was simulated until reaching a value of $r_{tol} = 0.05$ for different values of K, with p and α held constant. The results for the energy cost associated with these separate trials are shown in Table 2.2, while the settling times are shown in Table 2.3.

As coupling strength moves out of the weak regime where the above averaging as-

Table 2.3: Settling Time (ms) to Reach K-Cluster State

	$K=2$	$K=3$	$K=4$	$K=5$	$K=6$
Hodgkin-Huxley	191.275	207.2	279.825	386.2	518.426
Hodgkin-Huxley ($a_e = 0.01$)	191.55	200.45	267.825	379.5	518.65
Thalamic	387.25	1074.15	1128.05	1167.45	1634.25
Thalamic ($a_e = 0.01$)	382.775	1073.52	1127.82	1168.07	1644.55

sumptions hold, the cost associated with achieving clustering increases unboundedly as the value of a_e approaches some asymptotic limit. To demonstrate this, the system was simulated for different coupling strengths varying from the weak regime to the moderate coupling regime. This was done for two different values of α to demonstrate robustness with respect to the choice of α. Fig. 2.10 shows the energy cost for the reduced Hodgkin-Huxley phase model for $\alpha = 0.1$ and $\alpha = 0.01$ as a_e is varied. We consider $\log a_e$ to demonstrate it approaches an asymptotic limit; by using the log value, we treat the uncoupled case as the asymptotic behavior as $\log(a_e) \to -\infty$. The resulting plot is characteristic of a model of the form $\frac{1}{(a\log(x)+b)^c} + d$, suggesting an asymptotic limit in a_e rather than growth governed by an exponential or power law model. This is consistent with simulations failing to converge to the target state for a_e sufficiently large.

The inability to reach the target value of r_{tol} should not, however, be seen as a complete failure to achieve clustering. Rather, as the coupling strength increases, the system reaches a state wherein the value of r fluctuates in a complicated manner about some average value. The stronger coupling is in this regime, the generally larger the variations between the maximum and minimum of the cycle; similarly, the range of inputs grows increasingly large as well. These phenomena are shown in Fig. 2.11. As can be seen, beyond a certain value for a_e, increasing the coupling strength actually causes the minima of $|u|$ and r to decrease; this is likely the result of the neurons reaching a configuration where the coupling actually aids in clustering instead of working against it, with the strength being high enough to reach much more closely to the ideal state before coupling

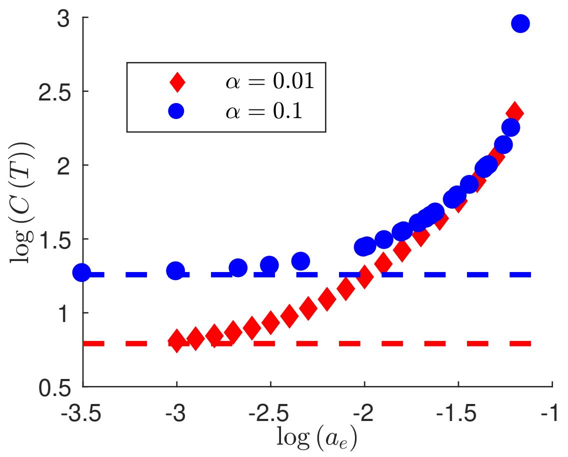


Figure 2.10: Cost to reach r_{tol} for varying coupling strengths at two different values of α for the Hodgkin-Huxley model ($K = 4$). As can be seen, in the weak coupling regime ($a_e \lesssim 0.01$) the system asymptotically approaches the uncoupled dynamics of the system (denoted by the dashed lines), but as moderate coupling is approached the required energy grows unbounded. This is consistent with the underlying assumptions used in the process of averaging, namely that perturbations are small.

pulls the neurons away from the target state once more. This is consistent with [54], which found that depending on the initial configuration, coupled neuron networks could end in smeared one-cluster or multi-cluster states if there was sufficient connectivity between neurons, depending on the initial conditions. Here, the application of control effectively allows the system to transition from the basin of attraction of one of these configurations (the one-cluster state) to a different configuration (the multi-cluster state).

Beyond a certain value of a_e, IMIE as described cannot achieve clustering and the neurons instead coalesce. This is to be expected, as the fundamental assumption of its formulation – namely, that the effects of coupling are weak – is no longer accurate. As can be seen in Fig. 2.12, at higher coupling pronounced peaks and troughs can be seen in each cycle of the neuron population. The disparity between the peaks and troughs continues increasing as the coupling strength is increased. Below a critical value, these oscillations in r are damped as time progresses, leading to a relatively steady solution. Above a critical value, however, the oscillations instead increase, with each peak reaching

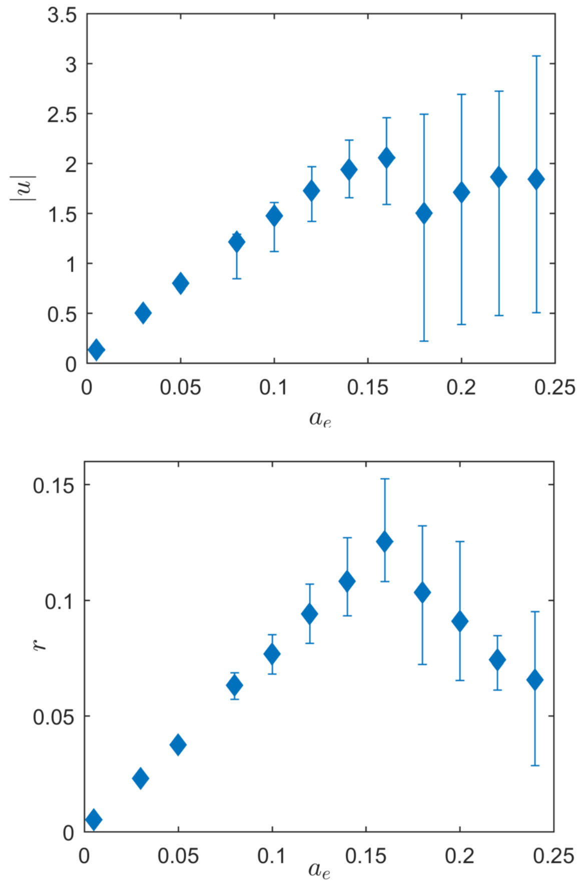


Figure 2.11: Asymptotic behavior for Hodgkin-Huxley phase model, varying values of coupling strength. A population of Hodgkin-Huxley neurons ($N = 50$, $K = 4$) evenly spaced in phase space was simulated for 3000 ms; the final 1000 ms were averaged and analyzed to determine asymptotic behavior. The variation in the maximum and minimum magnitude for the input, as well as the average, is shown in the *top* panel, whereas the variation in the value of r is shown in the *bottom* panel.

a higher value of r than the previous peak, eventually tending toward ∞ and indicating clustering cannot be achieved with the averaged technique described.

2.3 Some Thoughts on IMIE

We have demonstrated a potentially effective strategy for designing controls for achieving clustering of populations of neuronal oscillators. Simulations of both phase models and full state models have demonstrated a significant improvement in cost when employing our input of maximal instantaneous efficiency (IMIE) as compared to other methods that make use of the same level of information when generating a closed-loop control strategy. Additionally, this method works not only for identical uncoupled neurons, but can be extended to accommodate weak to moderate coupling as well.

The use of two separate neural models, the Hodgkin-Huxley and thalamic models, demonstrates the robustness of the control strategy. While the Hodgkin-Huxley model is not physically relevant to the specific research areas of interest, the thalamic model provides a direct link to problems such as essential tremor. Research such as [55] shows the correspondence between the firing behavior of singular neurons in the thalamus and the tremors in essential tremor, and [56] indicates that DBS modifies and entrains the firing of oscillators in the thalamus. The flexibility of the control strategy we have presented allows it to achieve the same or similar results as conventional DBS, potentially at a fraction of the cost.

The control strategy developed here in Chapter 2 is not without its limits, however, and its limitations motivate our move into machine learning in the remainder of this dissertation. In the case of PD, the behavior of the basal ganglia is complex and involves the interplay of neurons in the STN, GPi, and other portions of the basal ganglia. As such, developing a control to generate desynchronization of firing activity is more complicated

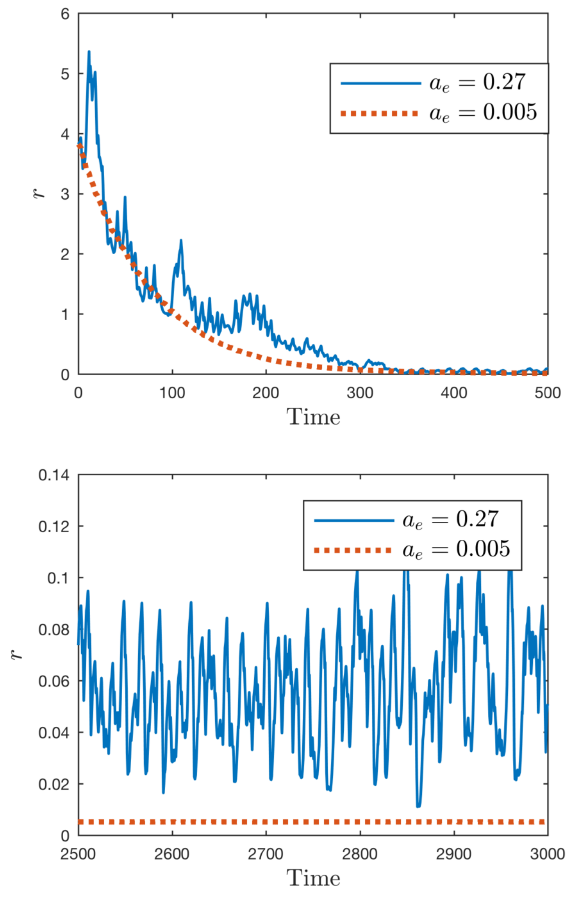


Figure 2.12: Evolution of Hodgkin-Huxley phase state models for weak ($a_e = 0.005$) and moderately strong ($a_e = 0.27$) coupling. The first 500 ms of simulation are shown in the *top* panel; The *bottom* panel expands the region of $2500 < t < 3000$. When coupling is weak, the evolution of r is relatively smooth; however, as a_e increases the corresponding evolution of r becomes increasingly jagged, with each cycle containing peaks and troughs. Eventually, as a_e is increased farther, successive troughs and corresponding peaks increase in value instead of decreasing, and clustering cannot be achieved. This occurs in the Hodgkin-Huxley phase models around $a_e \approx 0.275$.

than in the case of the two models presented here. In principle, if we can develop an accurate phase model reduction to represent the various components of the basal ganglia, we can apply all of the techniques we have just developed. That, however, is a decidedly nontrivial problem. We may take as a starting point the computational model of the basal ganglia presented in [33, 57] or [58]. In either case, we are, as a starting point, working with not one type of oscillator, but three – the STN, globus pallidus interna (GPi) and globus pallidus externa (GPe). These three neurons have distinct behavior, with excitatory and inhibitive effects connecting these three subpopulations to each other. These connections further invalidate our assumptions of light to moderate coupling; the coupling is so strong, in fact, that it is necessary for actually producing the oscillatory patterning we see in all three neuron subtypes (the STN spontaneously fires periodically, but the GPi does not).

Putting all this aside, another core issue we face is a practical one. Namely, this approach requires possession of a great deal of knowledgee about the state of the system, including the positions in phase space of all neurons at all times. While this may be feasible in simulation or *in vitro*, extending this to *in vivo* applications is not possible. If we want to tackle these challenges, we need to further generalize our approach while retaining the understanding and insight this investigation of phase models provides us – the control strategies here will be revisited with the tools of neural networks in Chapter 3.

2.4 Optimal Control with Global Phase Reduction

Before tackling these questions, however, we will look at another application of phase model reduction and again consider its benefits and drawbacks. The primary reason we bother with phase model reduction at all is that it allows us to take complex systems and simplify them significantly, and doing so allows us to use problem-solving methods that may be infeasible in higher-dimension domains to achieve control objectives. An example of this is [8], which uses optimal control in conjunction with phase model reduction to design stimuli to target specific firing times for the oscillator. For large input stimuli that may drive the system far from the limit cycle, however, these results may be inaccurate, as demonstrated in [59]. There the researchers avoided the problem by adding an additional constraint to the optimal control solution in the form of the isostable coordinate, to which they assigned a cost for being nonzero.

There are circumstances, however, where we may wish to deviate from the limit cycle. First, the optimal control signals generated in [59] required significantly more energy than their corresponding (but, admittedly, less accurate) isochron-only strategies. Giving the control the freedom to deviate from the limit cycle may allow for more cost-efficient solutions to the Euler-Lagrange equations. Second, there are circumstances where we may specifically want to move away from the limit cycle and toward some other feature of the system's state space, such as an unstable fixed point. This is the case, for example in [60], which seeks to drive neurons to the so-called "phaseless set" adjacent to the unstable fixed point encircled by the limit cycle in the Hodgkin-Huxley equations. Traditional augmented phase reduction does not give us the tools to do so; in that paper, the authors instead resort to the Hamilton-Jacobi-Bellman equation [61], which is computationally complex to solve, especially compared to optimal control via the variational principle and associated Euler-Lagrange equations. Using a globalized version of the phase model reduction, we can avoid the PDEs required of the HJB equation and greatly simplify designing a control that moves us toward the unstable fixed point.

2.4.1 Analytic Derivation of Global Isostables and Isochrons

We consider here the class of dynamical systems known as λ-ω systems, which are characterized by dynamics given by:

$$\frac{dr}{dt} = G\left(r\right), \qquad \frac{d\phi}{dt} = H\left(r\right). \tag{2.32}$$

Following the example of [62], we can write out a general form for the global isochrons following from (1.3):

$$\frac{d\theta\left(\mathbf{x}\right)}{dt} = \omega = \frac{\partial\theta}{\partial r}\dot{r} + \frac{\partial\theta}{\partial\phi}\dot{\phi}. \tag{2.33}$$

We now note that λ-ω systems are radially symmetric and that $\theta\left(r_1, \phi_1\right) = \theta\left(r_1, \phi_1 + 2\pi\right)$; these two facts taken together imply $\frac{\partial\theta}{\partial\phi} = 1$. This allows us to rearrange (2.33) and (substituting in for $\dot{r}$ and $\dot{\phi}$) convert it to an ODE:

$$\frac{d\theta}{dr} = \frac{\omega - H\left(r\right)}{G\left(r\right)}, \tag{2.34}$$

which has the solution:

$$\theta\left(\mathbf{x}\right) = \phi + \int \frac{\omega - H\left(r\right)}{G\left(r\right)} dr. \tag{2.35}$$

We can carry out a similar analysis for the global isostable coordinates, starting from (1.4):

$$\frac{d\psi\left(\mathbf{x}\right)}{dt} = k\psi\left(\mathbf{x}\right) = \frac{\partial\psi}{\partial r}G\left(r\right) + \frac{\partial\psi}{\partial\phi}H\left(r\right). \tag{2.36}$$

Because of radial symmetry, $\frac{\partial\psi}{\partial\phi} = 0$, and we may again rewrite to find an equation for ψ:

$$\frac{d\psi\left(r\right)}{dr} = \frac{k\psi\left(r\right)}{G\left(r\right)}. \tag{2.37}$$

2.4.2 Example: Supercritical Hopf Bifurcation

For the supercritical Hopf bifurcation, $G(r) = ar + cr^3$ and $H(r) = b + dr^2$; this is slightly more general than Example 4.1.1 from [63], which uses a different but related approach to find what we refer to as global isochrons and isostables. The ODE (2.37) for the global isostable coordinates is:

$$\frac{d\psi\left(r\right)}{dr} = -\frac{2a\psi\left(r\right)}{ar + cr^3}, \tag{2.38}$$

which has the general solution

$$\psi\left(r\right) = c_1\left(\frac{a}{r^2} + c\right), \tag{2.39}$$

where c_1 is an arbitrary scaling coefficient that can be chosen to normalize as desired. Note that rearranging (2.39) allows us to represent r in terms of ψ instead:

$$r = \sqrt{\frac{c_1 a}{\psi - c_1 c}}. \tag{2.40}$$

Now, substituting (2.39) into (2.38) and simplifying yields the IRC directly:

$$\mathbf{I}_{r,\phi}\left(r\right) = -\frac{2ac_1}{r^3}\hat{r} \qquad \Rightarrow \qquad \mathbf{I}_{r,\phi}\left(\psi\right) = -2\sqrt{\frac{\left(\psi - c_1 c\right)^3}{c_1 a}}\hat{r},$$

$$\mathbf{I}_{x,y} = -2\sqrt{\frac{\left(\psi - c_1 c\right)^3}{c_1 a}}\left(\cos\phi\hat{x} + \sin\phi\hat{y}\right).$$

We now carry out a similar analysis to find the global isochron coordinates. Our ODE (2.34) is

$$\frac{d\theta\left(\mathbf{x}\right)}{dr} = -\frac{d\left(r^2 + \frac{a}{c}\right)}{ar + cr^3},$$

which is solved as:

$$\theta\left(r,\phi\right)=\phi-\frac{d}{c}\log r+c_2. \tag{2.41}$$

To enforce $\phi=\theta$ on the periodic orbit, we select $c_2=\frac{d}{2c}\log\left(-\frac{a}{c}\right)$. The relation between ϕ and θ can then be flipped and written as:

$$\phi=\theta+\frac{d}{2c}\left(\log\left(\frac{c_1 a}{\psi-c_1 c}\right)-\log\left(-\frac{a}{c}\right)\right). \tag{2.42}$$

Lastly, by taking the gradient of (2.41), the global PRC is given by

$$\mathbf{Z}_{r,\phi}\left(r,\phi\right)=-\frac{d}{c}\frac{1}{r}\hat{r}+\frac{1}{r}\hat{\phi},$$

$$\mathbf{Z}_{x,y}\left(r,\phi\right)=-\frac{1}{r}\left(\frac{d}{c}\cos\phi+\sin\phi\right)\hat{x}+\frac{1}{r}\left(\cos\phi-\frac{d}{c}\sin\phi\right)\hat{y}.$$

In general, the PRC can be put in terms of the coordinates ψ and θ by substituting for r and ϕ using (2.40) and (2.42), respectively.

2.4.3 Example: Control to an Unstable Fixed Point

We now consider a control application which makes use of the above global isostable and isochron results. Typically, when we consider the phase reduction, one of the assumptions is that the trajectory remains close to the stable periodic orbit, i.e. $\psi\approx 0$. There may be situations, however, where this is not desirable; in this case, the underlying assumptions of the non-global approach no longer hold and it is necessary to use global coordinates. For example, for the supercritical Hopf bifurcation there is an unstable fixed point interior to the stable periodic orbit; attempting to control to this point using the traditional augmented phase reduction analysis would yield an inaccurate result, but with the global coordinates we can effectively drive the system to the neighborhood of this unstable point. In terms of the stable periodic orbit's isostable coordinates, the unstable fixed point corresponds to $\psi \to \infty$; this can be seen from (2.39). Therefore, designing a control to reach the unstable point at time T_1 would correspond to maximizing the value of ψ at T_1.

To illustrate this, we consider the Hopf bifurcation normal form with the parameters $a = 0.1, b = 1, c = d = -1$. Additionally, we assume we can only apply a control input in the $\hat{x}$ direction. The components of the PRC and IRC in the $\hat{x}$ direction are, respectively,

$$Z\left(r, \phi\right) = -\frac{1}{r}\left(\frac{d}{c}\cos\phi + \sin\phi\right) \tag{2.43}$$

and

$$I\left(r, \phi\right) = -\frac{2a}{r^3}\cos\phi. \tag{2.44}$$

Note that for convenience, for (2.44) we have taken $c_1 = 1$.

We now design a minimum-energy control that brings the system close to the fixed point. Our cost function is given by:

$$\mathcal{C} = \int_0^{T_1} u\left(t\right)^2 dt, \tag{2.45}$$

where $u(t)$ is the control input, and T_1 is the time at which we want the trajectory to be within a specified Euclidean distance from the unstable fixed point. We apply calculus of variations to minimize

$$C[u(t)] = \int_0^{T_1} [u(t)]^2 + \lambda_1\left(\dot{\theta} - \omega - Z(\psi, \theta)u(t)\right) + \lambda_2\left(\dot{\psi} - k\psi - I\left(\psi, \theta\right)u\left(t\right)\right) dt. \tag{2.46}$$

The associated Euler-Lagrange equations are:

$$2u\left(t\right)-\lambda_1 Z\left(\psi,\theta\right)-\lambda_2 I\left(\psi,\theta\right)=0 \quad \Rightarrow \quad u\left(t\right)=\frac{\lambda_1 Z\left(\psi,\theta\right)+\lambda_2 I\left(\psi,\theta\right)}{2}, \tag{2.47}$$

$$\frac{d\lambda_1}{dt}=-\lambda_1\frac{\partial Z}{\partial\theta}u\left(t\right)-\lambda_2\frac{\partial I}{\partial\theta}u\left(t\right), \tag{2.48}$$

$$\frac{d\lambda_2}{dt}=-\lambda_1\frac{\partial Z}{\partial\psi}u\left(t\right)-\lambda_2\frac{\partial I}{\partial\psi}u\left(t\right)-\lambda_2 k. \tag{2.49}$$

Because it is more convenient in general to represent the PRC and IRC in terms of the original polar coordinates, we calculate the partial derivatives with respect to θ and ψ by utilizing the chain rule:

$$\frac{\partial Z}{\partial\psi}=\frac{\partial Z}{\partial r}\frac{\partial r}{\partial\psi}+\frac{\partial Z}{\partial\phi}\frac{\partial\phi}{\partial\psi}, \qquad \frac{\partial I}{\partial\psi}=\frac{\partial I}{\partial r}\frac{\partial r}{\partial\psi}+\frac{\partial I}{\partial\phi}\frac{\partial\phi}{\partial\psi}, \tag{2.50}$$

$$\frac{\partial Z}{\partial\theta}=\frac{\partial Z}{\partial\phi}\frac{\partial\phi}{\partial\theta}, \qquad \frac{\partial I}{\partial\theta}=\frac{\partial I}{\partial\phi}\frac{\partial\phi}{\partial\theta}. \tag{2.51}$$

The partial derivatives of r and ϕ are computed from (2.40) and (2.42). A shooting method on the reduced model was employed to obtain the values of λ_1 and λ_2 required to give the value of ψ corresponding to $r=0.01$ at time T_1; Figure 2.13 shows the results of a realization for $T_1=12$. Then, the original Hopf bifurcation normal form model was simulated using the input corresponding to these initial values for λ_1 and λ_2 for the reduced model. At the target time T_1, the control was turned off ($u=0$) and the system was allowed to relax back to the periodic orbit. As can be seen, this control strategy effectively forces the system into the vicinity of the unstable fixed point.

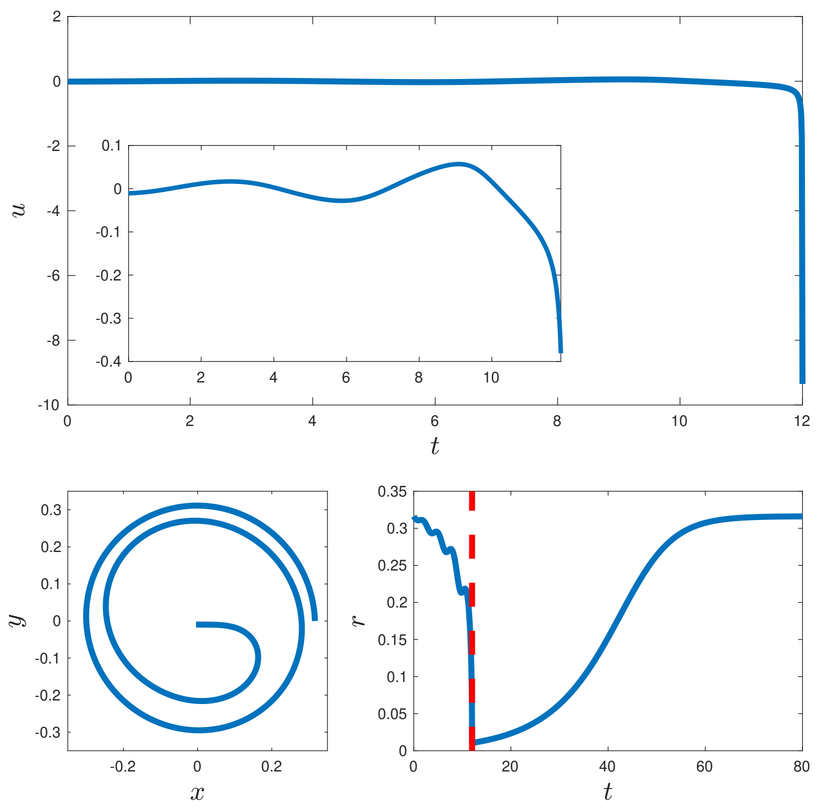


Figure 2.13: Control to an unstable fixed point for the Hopf bifurcation normal form: the *top* panel shows the computed optimal input to reach $r = 0.01$ at $T_1 = 12$, with the inset showing more detail. The *bottom left* panel shows the trajectory of the system which starts at $(x, y) = (r_{po}, 0)$ until it reaches the target for r, and the *bottom right* panel shows the slow relaxation back to the periodic orbit after the control is turned off at $T_1 = 12$.

2.5 Final Thoughts on the Global Phase Reduction

We again note that this method is computationally simpler than using a Hamilton-Jacobi-Bellman approach to achieve a similar goal of steering a trajectory to an unstable fixed point [60, 64]; however, it requires knowledge of the global PRC and IRC, which might not always be possible. If we have the right-hand side of the dynamics for our system, it may be possible to derive an analytic equation for these global coordinates;

existing methods, however, do not provide an effective strategy for handling cases where we do not already know the dynamics of our system. We will revisit this in Chapter 5, in which we will explore how machine learning can solve these problems.

The greatest hurdle in designing controls for systems optimally has been, as we have seen now several times, the value of information. The more information we have about these systems, the more we can do to develop clever, insightful, or simple solutions that leverage the tools of classical dynamics and control analysis. But often, this information hinges on assumptions about the system that, in practice, may not be realizable: either assumptions about the interactions between components of the systems (such as in the case of coupling) or near-perfect information about either the underlying dynamics of the system or state information at any given time. The focus in the remainder of this dissertation will be on showing how machine learning can allow us to relax these constraints and work with our foundation in dynamics and control to develop more effective techniques for controlling systems where we may not be able to observe every detail of the system we would prefer to.

Chapter 3

Toward Machine Learning: Using ML to Solve Traditional Control Problems

We revisit the core ideas from Chapter 2 in Chapter 3, with the crucial distinction that we will now move away from the phase reduction model and narrow our scope slightly. This allows us to reduce the expectation we have of the information available – whereas in the previous work we assumed we essentially knew every state of the system, the work presented here assumes far less about the information available to us in our efforts to achieve desynchronization. In fact, we may say that here we will observe less than one state, as we will see.

3.1 Artificial Neural Network Design and Training

A primary goal of this research was to utilize as little information as possible when training the neural network and designing the control. To this end, we assumed that the only measurement we could recover from the neuron was the timing of its spikes. The control signal used, however, was assumed to be known, and is characterized by three variables: signal amplitude I_0, signal width t_{width}, and signal delay time t_{delay}. The delay time is the length of time after the previous spike that the input stimulus is applied. The system's output was the expected value of the neuron's next firing time, t_{final}. The input $\vec{x}$ and output $\hat{y}$ of our neural network can therefore be written as:

$$\vec{x} = [I_0,\ t_{width},\ t_{delay}\ ]^T\,, \tag{3.1}$$

$$\hat{y} = E\left[t_{final}|\vec{x}\right]. \tag{3.2}$$

The neural network was trained via random trials of $\vec{x}$ selected uniformly from the ranges $0 \leq I_0 \leq 25$ mA, $1 \leq t_{width} \leq 9$ ms, and $0 \leq t_{delay} \leq 33$ ms. For each $\vec{x}^i$, the neural model was simulated (here, a conductance-based model of the STN adapted from [33]; the STN model used is provided in A.3) and the next spike time $y^i = t_{final}$ was recorded. Here, the superscript notation denotes the ith training example from the data.

Prior to being used to train the neural network, $\vec{x}$ was normalized to account for differences in the ranges of I_0, t_{width}, and t_{delay}. The transformed $\vec{x}$ was computed as:

$$\hat{x}_j = \frac{x_j - \mu_{x_j}}{\sigma_{x_j}}, \tag{3.3}$$

where μ_{x_j} and σ_{x_j} are the arithmetic mean and standard deviation, respectively, of the jth element of $\vec{x}$.

The model was trained using 10,000 random trials, with 70% of the trials used as the training set and the remaining 30% held out for cross-validation. Parameters for the neural network were initialized using the Xavier initialization protocol [65] and updated via batch gradient descent. Hyperparameters for the number of hidden layers, the number of neurons per layer, and the learning rate were tuned using a Bayesian search optimization and optimizing the harmonic mean HM of the quadratic loss function $\mathcal{L}$ applied to both the training set and the cross-validation set. More concretely:

$$\mathcal{L}\left(\hat{y}, y\right) = \frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^i - y^i\right)^2 , \tag{3.4}$$

$$HM = \frac{2}{\frac{1}{\mathcal{L}_{train}} + \frac{1}{\mathcal{L}_{C-V}}}. \tag{3.5}$$

Here, $m = 7000$ for the training set and $m = 3000$ for the cross-validation set. This process yielded a neural network with 5 hidden layers and neuron numbers whose values are presented in Table 3.1. The hidden layers utilized *tanh* activation functions, while the output layer contained a linear activation function to account for the complete range of possible period measurements. A learning rate of $\eta = 0.0015$ was used to achieve convergence, and the model was trained for 50,000 epochs.

Table 3.1: **Neural network hyperparameters.**

Layer #	1	2	3	4	5
# of Neurons	80	100	60	30	80

After confirming the model's accuracy on the training set, we cross-validated against the held out data to check for overfitting. As can be seen in Fig. 3.1, the model maintained a high degree of accuracy in this cross-validation.

We note that the use of a neural network to perform this regression has two primary advantages over other common regression strategies we considered. First and foremost, the neural network allows us to achieve a far greater degree of complexity in the nonlinearities than other general regression methods, such as polynomial regression or Fourier coefficient regression. Because neuron dynamics exist in a high-dimensional, highly com-

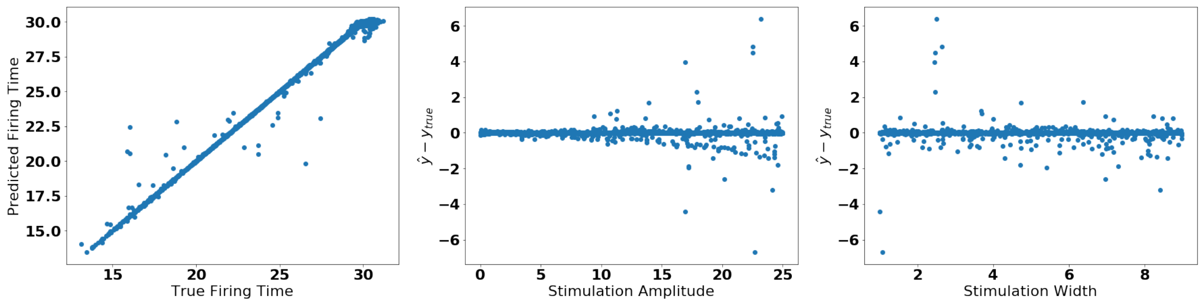


Figure 3.1: Analysis of cross-validation data prediction. In the *left* panel, predictions are plotted against true values; perfect accuracy would return a straight line along $y = x$. As can be seen, almost all predictions have low error, with some outliers. The *center* and *right* panels plot the errors against amplitude and stimulation width, respectively, revealing that the most challenging stimuli to model tend to be those with high amplitude and short duration. These represent stimulations near the firing threshold for the neuron – sufficient stimulation will generate an all-or-nothing action potential, while insufficient stimulation will simply return to the limit cycle, meaning a small change in stimulation parameters can have a strong, divergent effect on the resulting firing time.

plex space, this generality that the deep neural network provides is especially valuable in the context of modeling stimulus responses in neurons. A comparison of the accuracy of our neural network to two other regression strategies can be found in Table 3.2. Second, because future estimations are provided by an explicit, feed-forward model based on provided input values, the neural network produces an estimate that is efficient to compute once the initial training is complete. Other strategies, such as interpolating from the scattered data, are necessarily slower; this difference in processing speed can be significant in the context of on-line neural control, where computational efficiency is of particular importance.

3.1.1 Evaluation of Multiple Neurons Simultaneously

To extend the ANN to consider a larger population of neurons, the firing times of all neurons in the population were defined relative to a reference time, selected as the firing time of the latest-firing neuron in the population. The appropriate result estimation,

Table 3.2: **Comparison of performance of neural network to other regression methods.**

	Training Error	C-V Error
Polynomial Fit	0.7787	0.9297
MATLAB scatteredInterpolant	N/A	0.4674
Neural Network	0.02688	0.0490

The error was calculated according to the loss function (3.4). The polynomial interpolation was estimated using a least-squares analysis of polynomial terms of the type $a_{j,k,l} I_0^j t_{width}^k t_{delay}^l$.

then, was evaluated as t_{delay} with this offset subtracted. For example, if the reference neuron fired at 0 ms and a second neuron fired at -4 ms, then an input that occurs at $t_{delay} = 5$ for the reference neuron occurs at $t_{delay} = 9$ for the second neuron. The ANN therefore can be efficiently used for larger neural populations.

As an extension of this process, an "event horizon" mapping was also generated from the ANN's regression analysis. To extend the neural network to the case of multiple inputs within a single cycle, it was necessary to estimate not just when the neuron was stimulated but also when the stimulus ends relative to the limit cycle. This was accomplished by making the simplifying assumption that the neuron rapidly returns to the limit cycle following the end of a given stimulus. An event horizon time t_{hor} disallows subsequent input stimuli with start times $t_{delay} < t_{hor}$. The event horizon time was computed as:

$$t_{hor} = T_0 - (\hat{y} - t_{delay} - t_{width}), \tag{3.6}$$

where T_0 is the natural period of the neuron. For example, if a stimulus at $t_{delay} = 15$ with $t_{width} = 3$ causes the period to change from 24 to 21, although the stimulus ends at $t = 18$, its event horizon time would be 21 because it ends 4 ms prior to the neuron's firing time. Any subsequent stimuli that may be applied would need to have a value of

$t_{delay} > 21$ since all delay values were considered relative to the original firing time. This is illustrated in Fig 3.2. Additionally, we note T_0 can itself be derived from the neural

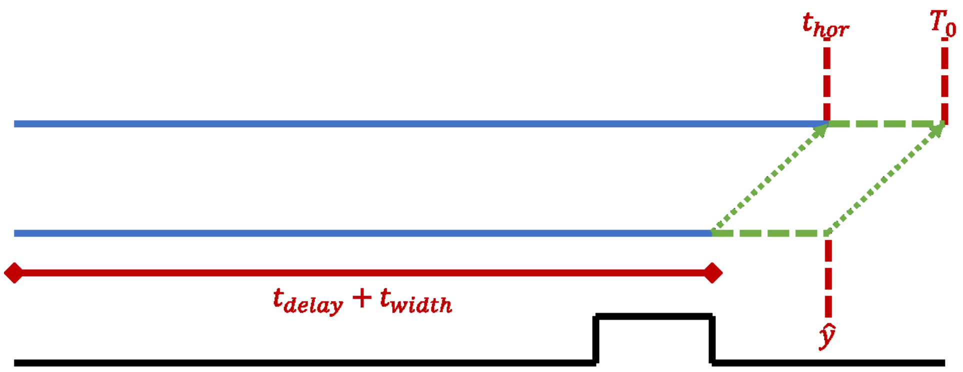


Figure 3.2: Illustration of event horizon mapping. Time increases from 0 going left to right. The event horizon time is computed by translating the remaining time until the neuron fires after the completion of the input stimulus (*black solid line*) to the corresponding remaining time relative to the original firing time T_0. This lines in this figure are to scale with the example described in the paper ($t_{delay} = 15$, $t_{width} = 3$, $T_0 = 24$, $\hat{y} = 21$).

network by setting $t_{delay} \gg T_0$ for a candidate stimulus (if $t_{delay} > T_0$, the stimulus occurs *after* the next spike, and the network should predict a firing time that precedes the value of t_{delay}). A realization of this event horizon mapping for $t_{width} = 1$ is shown in Fig 3.3.

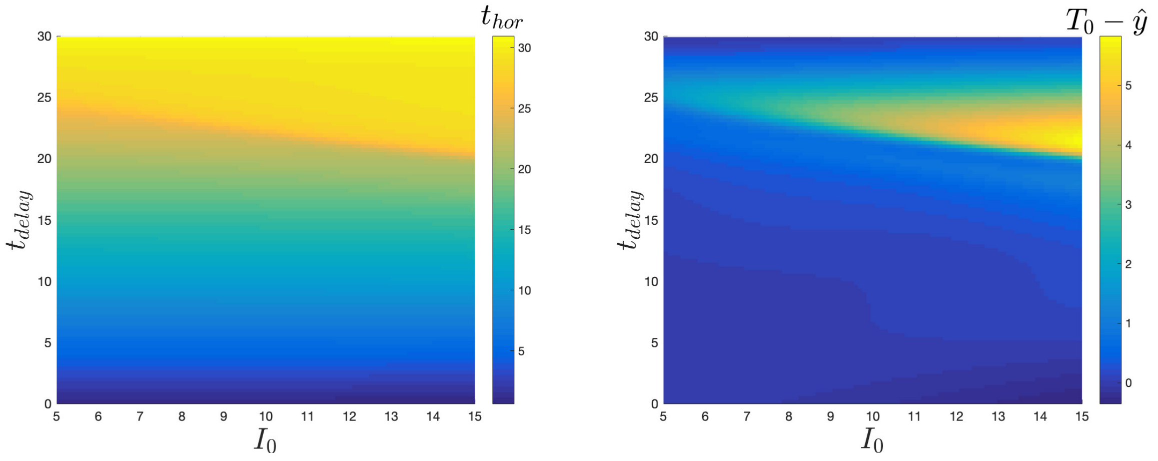


Figure 3.3: Event horizon mapping (*left*) and period gain mapping (*right*) for $\mathbf{t_{width} = 1}$. The event horizon mapping is calculated via (3.6); the period gain mapping shows the amplitude of the reduction in period generated by stimulating at the given t_{delay} and I_0.

3.2 Maximally Efficient Desynchronization

We now turn our attention to application of our regression model to the desynchronization of two neural oscillators. We consider the voltage traces of two identical uncoupled STN neurons, V_1 and V_2. We define the spike time as the time at which V_i is locally maximal and above a threshold voltage (here, 0). The initial spike times for the neurons are separated by some small increment Δt; because the neurons are identical, uncoupled, and at steady state, this spike time difference will persist from cycle to cyle unless the neurons receive an input. We define, without loss of generality, V_2 to be the "leading" neuron, meaning V_2 fires before V_1 and the firing times can be related by:

$$t^{V_1}_{spike} = t^{V_2}_{spike} + \Delta t, \ \Delta t > 0. \tag{3.7}$$

If during the subsequent cycle the inter-firing times of V_1 and V_2 differ by Δt_{final}, the total separation between the two neurons is given by:

$$\Delta t_{total} = \Delta t + \Delta t_{final}. \tag{3.8}$$

Our goal is to increase the value of Δt_{final} as efficiently as possible when the two neurons are subject to a common stimulus. Following our previous work in Chapter 2, we therefore design our control to minimize our value function $Q(x, y)$:

$$Q = \frac{-\left(t^{V_1}_{final} - t^{V_2}_{final}\right)}{\alpha \int_0^t I(t)^2 \, dt + \epsilon}, \tag{3.9}$$

where α is a weighting term, $\epsilon > 0$ is an offset to prevent division by zero if $I = 0$, and $t^{V_1}_{final} - t^{V_2}_{final} = -\Delta t_{final}$. The values of $t^{V_1}_{final}$ and $t^{V_2}_{final}$ are computed assuming the neuron previously fired at $t = 0$. We additionally note that, since we are only considering square waves, the integral can be replaced:

$$\int_0^t I(t)^2\, dt = I_0^2 t_{width}. \tag{3.10}$$

Because the only difference between the two neurons is the initial offset Δt, the final time estimation is just the difference in the expected values:

$$t_{final}^{V_1} - t_{final}^{V_2} = E\left[y|\vec{x_1}\right] - E\left[y|\vec{x_2} = \vec{x_1} + [0,\ 0,\ \Delta t]^T\right]. \tag{3.11}$$

We minimized Q via gradient descent for multiple initial separations Δt subject to two restrictions:

1. Q was minimized while holding amplitude constant, then amplitude was varied; and
2. t_{width} was not allowed to decrease below $t_{width} = 0.5$.

The first condition was applied to recognize that control may be subject to specific constraints, such as a particular amplitude or performance characteristics (desynchronization time or total energy). The second condition was to ensure the model did not minimize the cost by reducing t_{width} to a physically-unrealizable negative number.

The most energy-efficient signal for desynchronizing a pair of neural oscillators as a function of amplitude I_0 is shown in Fig 3.4. To validate the results of the control, the error in the output when applied to the original ODE was calculated as:

$$err_{rel} = \frac{\sqrt{\left(\Delta t_{final,est} - \Delta t_{final,sim}\right)^2}}{\left|\Delta t_{final,sim}\right|}, \tag{3.12}$$

where $\Delta t_{final,sim}$ and $\Delta t_{final,est}$ are the simulation and machine learning-derived values of $t_{final}^{V_1} - t_{final}^{V_2}$, respectively. Both this error and the absolute error $(\Delta t_{final,est} - \Delta t_{final,sim})$

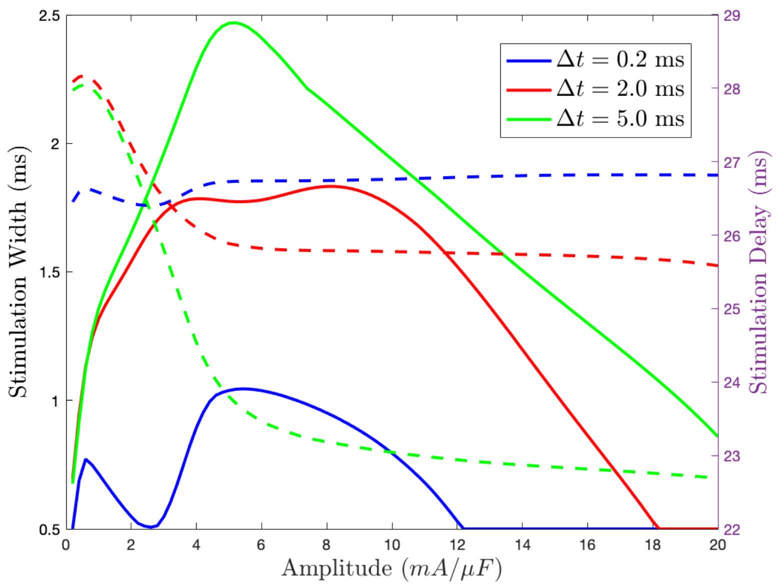


Figure 3.4: Optimal t_{width} (*solid line*, scaling on left y-axis) and t_{delay} (*dashed line*, scaling on right y-axis) for desynchronizing a pair of neural oscillators as a function of stimulus amplitude I_0 for three different initial separations. As can be seen, all three separations show the same general trend in terms of the optimal signal, with shorter signals typically preferred at higher amplitude.

are shown in Fig 3.5. The high relative error demonstrates the limits of the neural network's modeling: for low-separation systems (here represented by the $\Delta t = 0.2$ curves), the model underperforms, most likely due to limitations in the resolution of the model. With the size of the training set and the hyperparameters chosen, the model does not distinguish well between the responses of two neurons that start close together (here, separated by less than 1% of the natural period of the neuron). A similar problem arises for small-amplitude signals regardless of the separation: because the response to small-amplitude signals is relatively small, the relative effectiveness is limited by the underlying accuracy of the model. We note, however, that the model performs exceedingly well outside of these extreme cases, with relative errors under 10% and absolute errors on the order of 1% of the STN neuron's period.

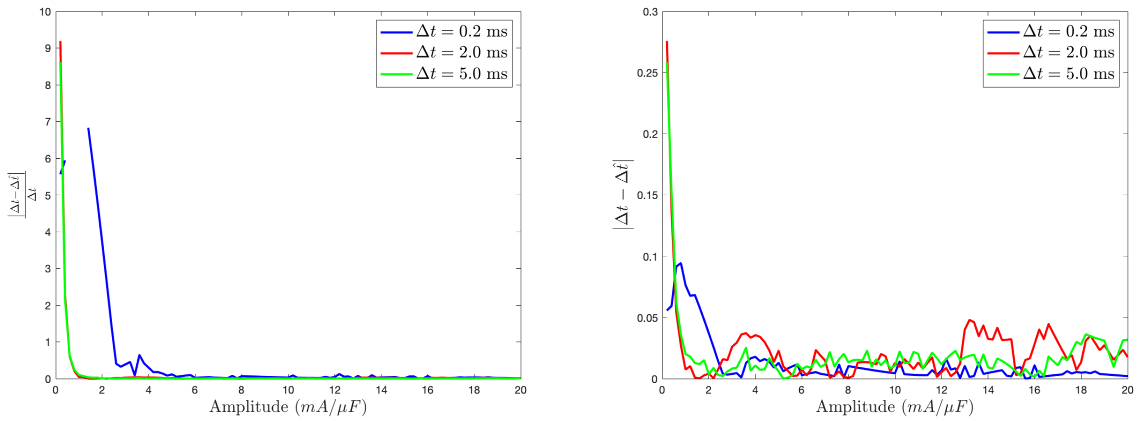


Figure 3.5: Relative (*left*) and absolute (*right*) errors by amplitude for three different initial separations. Although absolute error is low across all samples and lowest for the smallest separation ($\Delta t = 0.2$ ms), the relative error for the smallest separation is highest because the true measured improvement is near 0; for a portion of the signal range, in fact, the change is sufficiently small that there is no measured change, causing the denominator of the relative error to be identically 0 (this can be seen in the plot up to roughly 2 mA). However, for most amplitudes and stimulation lengths, both the relative and absolute errors are insignificant and near 0 as stimulation amplitude increases.

3.3 Dynamic Programming Desynchronization

The use of multiple inputs to optimally desynchronize a pair of neurons over a single cycle was carried out by leveraging dynamic programming. In the interest of computational efficiency, we opted for a two-level approach to dynamic programming: on a cycle-to-cycle level, the control was greedy, attempting to maximize the desynchronization, while individual inputs within the cycle were found recursively by working backward from the final state, here $\Delta t_{total} = \frac{T_0}{2}$.

The process within a given cycle proceeded as follows: potential choices of I_0 and t_{delay} were gridded while t_{width} was held constant. Additionally, a minimum time of 3 ms between input stimuli was imposed. For a selected (I_0, t_{delay}) pair, a binary search function was used to find the spike time difference that, when the pair of neurons is subjected to the given stimulus, would result in the desired final spike time difference.

This process was then iteratively repeated for every input stimulus whose event horizon was less than $t_{delay} - 3$ (with the difference accounting for the enforced minimum time between firing).

The goodness of a given sequence of control stimuli was determined via a value function:

$$P = \Delta t_{final} - \Delta t_{initial} - \beta I_0^2, \tag{3.13}$$

where β represents a weighting of the input stimulus power relative to the time improvement and $\Delta t_{initial}$ represents the separation at the start of the given cycle; in general, a higher value of β should produce lower-energy desynchronizations requiring more cycles to complete, while a lower value of β should produce higher-energy desynchronizations requiring fewer cycles to complete.

The input stimulus sequence that maximized P determined the endpoint for the previous cycle's optimization problem. For each cycle the same process was applied until the start point for a given cycle terminated at a spike time difference of less than 1 ms. The process for a given cycle is presented in Fig 3.6.

The success of our dynamic programming approach shows that the perturbations to the limit cycle introduced by the applied control were sufficiently weak to allow the neuron to return to its limit cycle in a short timeframe, thereby extending the functionality of the regression model to input sequences. The value of an input sequence was calculated using the neural network according to (3.13). The end separation Δt_{final} was taken to be half the neuron's natural period, or roughly 15 ms. Two different values of β were used to verify the existence of distinct control schemes depending on the relative weighting of energy cost. The faster-responding system used a value of $\beta = 0.01$, while the slower-responding system used a value of $\beta = 0.1$.

After the input sequence was calculated from dynamic programming applied to the

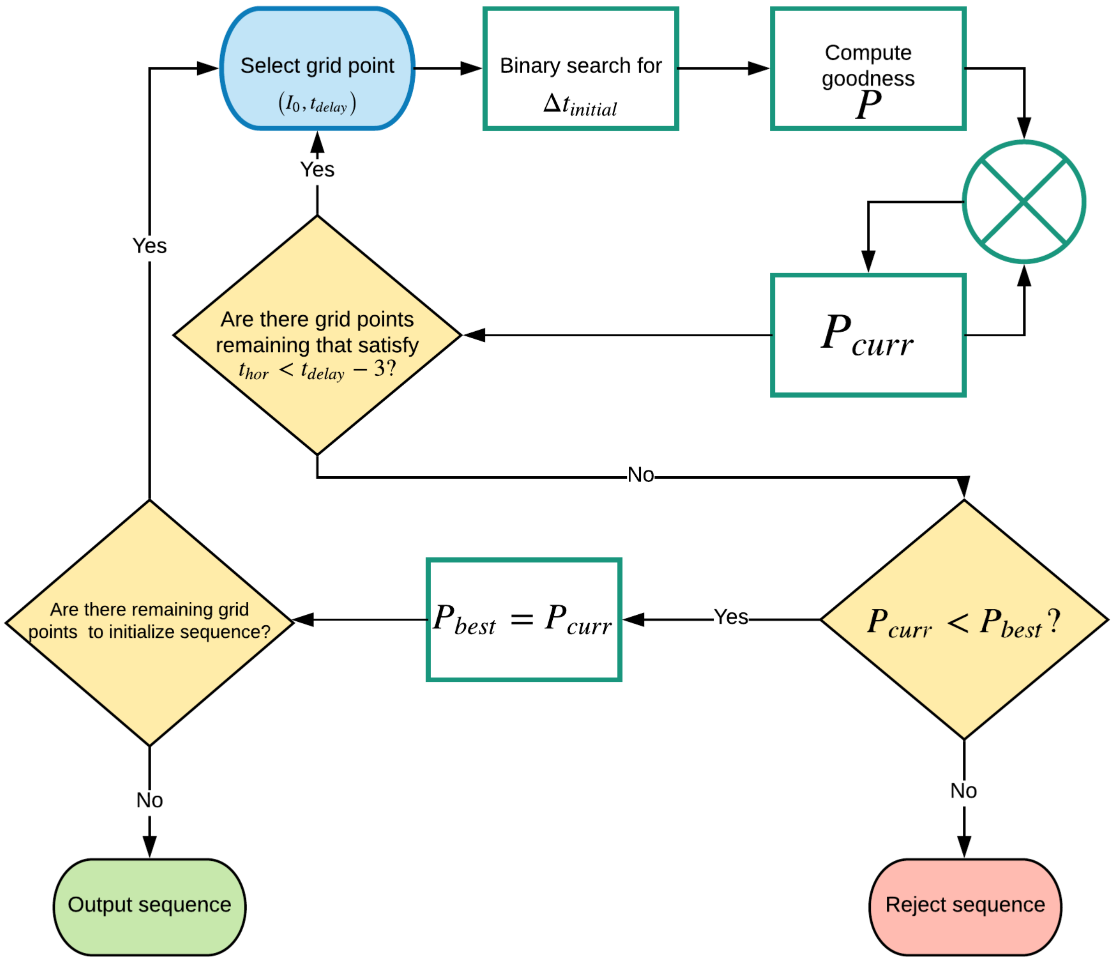

Select grid point (I_0, t_{delay})
Binary search for $\Delta t_{initial}$
Compute goodness P
P_{curr}
Yes
Are there grid points remaining that satisfy $t_{hor} < t_{delay} - 3$?
No
Yes
Are there remaining grid points to initialize sequence?
$P_{best} = P_{curr}$
Yes
$P_{curr} < P_{best}$?
No
No
Output sequence
Reject sequence

Figure 3.6: Process flowchart for dynamic programming application. This process was carried out independently from cycle to cycle until the terminating condition was reached.

ANN, it was applied to the full ODE to verify the result. Rather than using the input sequence as an open-loop stimulus, the firing time of the reference neuron was used as the basis for setting the next cycle's input sequence (that is, t_{delay} was computed relative to the recorded spike time, not the neural network's estimate of the spike time. Because of the observed accuracy limitations at small input magnitudes and small separations seen in Fig 3.5, inputs for the dynamic programming analysis were constrained to the range of $5 \leq I_0 \leq 10$. Results can be seen in Fig 3.7. These results are consistent with expectations: as in the case of the efficient stimulus calculation, the dynamic programming yields a highly accurate final result, and the smaller β value returns a more-responsive input sequence offset by a higher energy cost. We note that, again consistent with prior results, performance does degrade if I_0 is allowed to take on smaller values or if the starting condition requires the neurons to be significantly closer (on the order of $\Delta t = 0.1$ instead of $\Delta t \approx 1$).

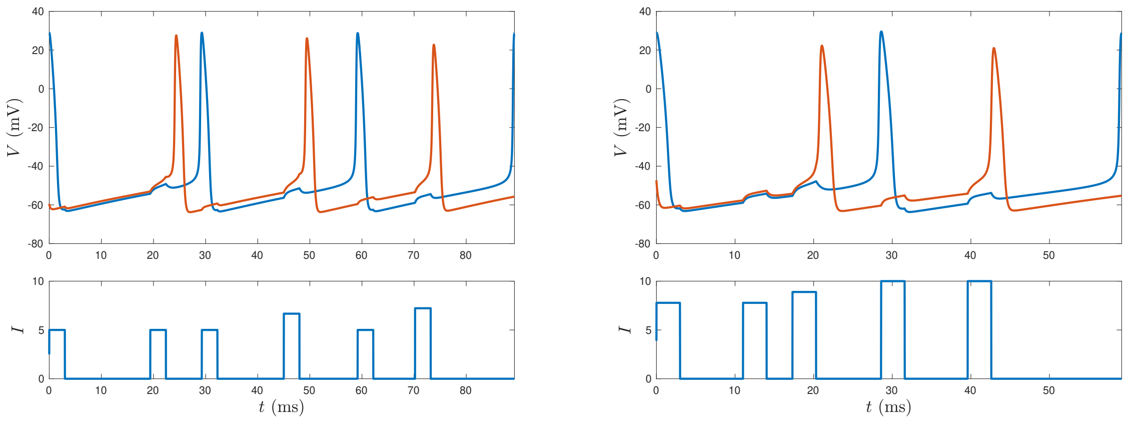


Figure 3.7: Computed stimuli and output for dynamic programming approach to desynchronization, $\beta = 0.05$ (*left*) and $\beta = 0.001$ (*right*). For both trials, $t_{width} = 3$. Note that both accurately yield a final target separation of ≈ 15 ms, but $\beta = 0.01$ results in an input sequence that is higher in energy and faster in response.

Larger inputs can also present challenges for multiple-input control applications. This is a byproduct of the underlying neural network, which only predicts the *next* firing time; larger stimuli will yield a correct prediction of the firing time, but the firing of the *subsequent* spike will not occur exactly a full period later even if no additional stimulus is applied. Although beyond the scope of this paper, future work may be warranted on predicting these aftereffects of stimulation in addition to the immediate effect.

3.4 Clustering of Neurons

3.4.1 Control Policy

Clustering, in which oscillators form discrete, finite groups with similar characteristic behaviors, was accomplished by defining a cost function related to the input stimulus and the change in a state function. This state function was adapted from the standard order parameter $R_k\left(\vec{\theta}\right)$ where R_k represents the k^{th} order parameter and is defined as [66]:

$$\left|R_k\left(\vec{\theta}\right)\right| = \frac{1}{N}\left|\sum_{l=1}^{N} e^{ik\theta_l}\right|. \tag{3.14}$$

Because we are solely interested in the magnitude of R_k, for simplicity of notation in this chapter we will henceforth understand R_k to be the real-valued magnitude $\left|R_k\left(\vec{\theta}\right)\right|$ as defined in (3.14). Here, θ is interpolated from when the neuron previously fired relative to the natural period of the neuron. Assuming the l^{th} neuron fired at $t = 0$, then, θ_l is represented as a linear transformation:

$$\theta_l\left(t\right) = \frac{2\pi t}{T}. \tag{3.15}$$

Order parameters are frequently used for analysis in clustering problems, cf. [67]. In the order parameter context, $R_1\left(\vec{\theta}\right) = 1$ implies the oscillators are perfectly clustered in a single-cluster configuration, while $R_1\left(\vec{\theta}\right) = 0$ suggests desynchrony or a larger number of clusters instead. It should be noted, however, that one drawback of the order parameter is that, for $k \geq 2$, the distributions that generate $R_k = 1$ are non-unique. For example, the one-cluster case of $R_1 = 1$ also yields $R_2 = 1$, $R_3 = 1$, etc. In contrast, an optimally-distributed two-cluster system, with an equal number of neurons in each cluster and the clusters π radians out of phase with each other, would still have $R_2 = 1$, but R_1 would instead equal 0. R_4, however, would also equal 1.

This ambiguity provides a challenge for utilizing order parameters as a basis for control rather than as simply a diagnostic tool for the effectiveness of a control. Controlling to a two-cluster state, for example, requires not only mandating $R_2 = 1$ but also that $R_1 = 0$; more generally, a k-cluster state is better defined according to:

$$R_l = \begin{cases} 0, & l < k \\ 1, & l = k \end{cases}. \tag{3.16}$$

In general, we seek a cost function $C_k\left(\vec{\theta}\right)$ that satisfies:

$$C_k\left(\vec{\theta}\right) = \sum_{l=1}^{k} \alpha_l R_l\left(\vec{\theta}\right), \tag{3.17}$$

such that that C_k is minimized when (3.16) is satisfied. A second constraint results from the existence of undesirable local minima resulting from the definition of the order parameters. Suppose, for example, $k = 2$ and $R_1 \approx 1$. In this case, $R_2 \approx 1$ as well. If $\alpha_1 = -\alpha_2 = 1$, then $C_k \approx 0$ and the global minimum is $C_k = -1$. However, $R_1 = R_2 = 1$ is a local minimum of the cost function, as desynchronizing from the single-cluster state causes the value of R_2 to change more rapidly than that as R_1 (this can be shown by differentiation; for an example, see Fig 3.8). Therefore, some care must be used in selecting α_l to derive a valid cost function.

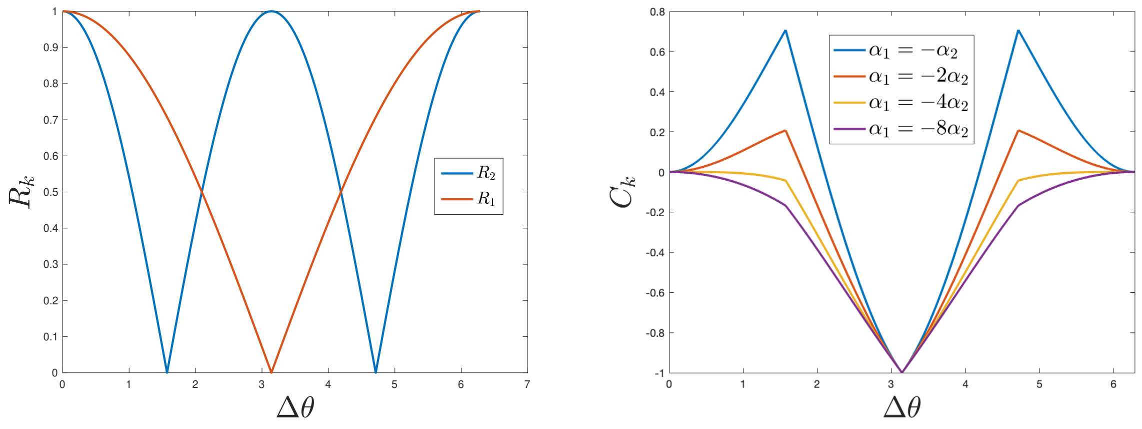


Figure 3.8: Effect of α_l selection on existence of local minima in C_k. To illustrate the potential existence of local minima related to coefficient selection, here we examine the $k = 2$, $N = 2$ case. As can be seen in the *left* panel, for $\Delta\theta < \frac{\pi}{2}$, R_2 decreases more rapidly than R_1, creating a local minimum at $\Delta\theta = 0$ when $\alpha_1 = -\alpha_2$. The *right* panel shows C_k (normalized such that $C_k\,(\Delta\theta = 0) = 0$ and $\min C_k = -1$) for different choices of α_1 relative to α_2. At $\alpha_1 \approx -4\alpha_2$, the fixed point at $\Delta\theta = 0$ loses its stability.

We derive here an approximate sufficient condition for the $k = 2$ case, noting that a similar analysis can be iteratively carried out for $k > 2$ as well. At the undesirable, one-cluster fixed point, the derivative with respect to the jth neuron's phase θ_j may be written as:

$$\frac{\partial C_k}{\partial \theta_j} = \sum_{l=1}^{N} \left[2\frac{\alpha_1}{R_1}\left(\sin\theta_l \cos\theta_j - \cos\theta_l \sin\theta_j\right) + 4\frac{\alpha_2}{R_2}\left(\sin 2\theta_l \cos 2\theta_j - \cos 2\theta_l \sin 2\theta_j\right)\right]. \tag{3.18}$$

Additionally, at this undesired fixed point, $\frac{\partial C_k}{\partial \theta_j} = 0$. A necessary and sufficient condition for this point to be unstable is that, for $|\epsilon| \ll 1$, $\left.\frac{\partial C_k}{\partial \theta_j}\right|_{\theta_j+\epsilon} < 0$. We make use of the Taylor approximations:

$$\begin{aligned}\sin\left(x+\epsilon\right) &= \sin x + \epsilon\cos x + \mathcal{O}\left(\epsilon^2\right)\\ \sin\left(2x+2\epsilon\right) &= \sin 2x + 2\epsilon\cos 2x + \mathcal{O}\left(\epsilon^2\right)\\ \cos\left(x+\epsilon\right) &= \cos x - \epsilon\sin x + \mathcal{O}\left(\epsilon^2\right)\\ \cos\left(2x+2\epsilon\right) &= \cos 2x - 2\epsilon\sin 2x + \mathcal{O}\left(\epsilon^2\right)\end{aligned}. \tag{3.19}$$

Additionally, we note that at the undesirable, one-cluster fixed point:

$$\begin{gathered}\sin\theta_j \approx \sin\theta_l \quad \forall l \in N\\ \text{and}\\ \cos\theta_j \approx \cos\theta_l \quad \forall l \in N.\end{gathered} \tag{3.20}$$

These approximations allow us to rewrite the derivative as:

$$\left.\frac{\partial C_k}{\partial\theta_j}\right|_{\theta_j+\epsilon} = -2\epsilon\frac{\alpha_1}{R_1}\sum_{l\neq j}\left(\cos^2\theta_l + \sin^2\theta_l\right) - 8\epsilon\frac{\alpha_2}{R_2}\sum_{l\neq j}\left(\cos^2 2\theta_l + \sin^2 2\theta_l\right), \tag{3.21}$$

where we have additionally used the fact that (3.18) is equal to 0 at the fixed point. Further simplifying, we find:

$$-2\epsilon\left(N-1\right)\frac{\alpha_1}{R_1} - 8\epsilon\left(N-1\right)\frac{\alpha_2}{R_2} < 0; \tag{3.22}$$

rearranging and solving the inequality with $R_1 = R_2 = 1$ yields the condition:

$$\alpha_1 > -4\alpha_2. \tag{3.23}$$

To aid performance, for our analysis we empirically found a simpler and more conservative bound was useful; for the $N = 50$ system, the coefficients α_l for the k-cluster problem were written as:

$$\alpha_l = \begin{cases} N^{k-l+1}, & l < k \\ -N, & l = k \end{cases}. \tag{3.24}$$

We note that we have made no claims or statements about reachability or controllability with this analysis, only that the condition provided is sufficient to overcome an empirically observed pitfall of order parameter-based control.

3.4.2 Population Clustering Simulation Results

A population of 50 neurons was simulated using the neural network. For each cycle, an input was selected by finding the minimal value on a three-dimensional grid of values for $(I_0, t_{width}, t_{delay})$ of the cost of the input according to the overall cost:

$$Q = C_k + \alpha I_0^2 t_{width}, \tag{3.25}$$

where α was set to 0.01 and C_k was of the form defined in (3.17).

Fifty different trials were run with initial firing times for each neuron randomly selected from a normal distribution ($\mu = 0, \sigma = \frac{T}{8}$). Once optimal input stimuli were generated via the neural network, the inputs were applied to the full ODE model. Phase values at the end of the simulation were calculated as the time that had elapsed since the neuron had last fired divided by the natural period T_0 and multiplied by 2π. Boxplots of these simulation results for two-cluster and three-cluster control objectives are shown in Fig 3.9. For both control objectives, the median values for the desired order parameter (R_2 for two clusters and R_3 for three clusters) were statistically significantly higher than the other two order parameters. The largest difference between the neural network's predicted results and the full simulation's output is that lower order parameters, notably R_1,

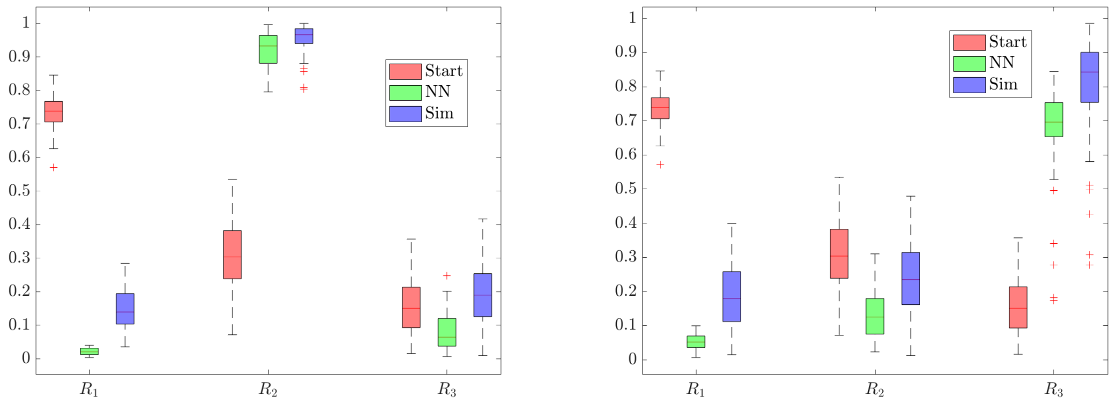


Figure 3.9: Boxplots of order parameters for 50 trials of two-cluster (*left*) and three--cluster (*right*) control objectives. Although there are some errors when comparing the predictions from the neural network (in green) to the actual output from the ODE system (in red), we can see that the neural network performs well and successfully maximizes the correct order parameter in both control objectives.

are not as effectively eliminated in the simulation when compared to the neural network's expectation. This is again consistent with the previous analysis of the regression; these errors are largely due to the inability to accurately distinguish the effects of input stimuli when two neurons are very close (< 1 ms). As a result, while in the neural network the correct number of neurons may end up in each cluster (ideally, balanced equally) which in turn leads the lower order parameters to approach zero, the network simply does not have the precision in practice to correctly cleave the initial distribution when the stimulus is applied to the simulation, resulting in slightly imbalanced (but still strong) clusters. A characteristic resulting output for the 3-cluster case (with values of R_1, R_2, and R_3 close to the median values for the ODE output) is additionally shown in Fig 3.10. We note that despite the values of R_1 and R_2 both hovering around 0.2 for this realization, a distinct 3-cluster state does emerge by the end of the simulation time.

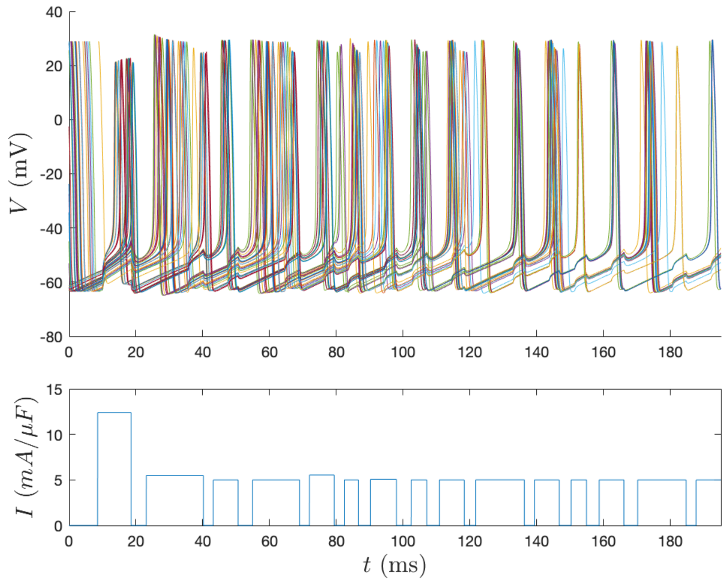


Figure 3.10: ODE simulation output with final order parameter values $R_1 = 0.1747$, $R_2 = 0.2686$, and $R_3 = 0.8392$. This particular realization was selected for its closeness to the median results, suggesting it is representative of an average clustering sequence. The *top* panel shows voltage traces for the 50 simulated neurons; we note that though they start normally distributed, they end in a clear 3-cluster configuration. The *bottom* panel shows the computed control input to generate the 3-cluster configuration.

3.4.3 Extension to Noisy Oscillators

An important measure of robustness when designing control algorithms is sensitivity to noise, so we applied the neural network's generated stimulation patterns to a noisy version of the ODE to evaluate its response. The same initial conditions and generated stimulation patterns from the previous section were again used, but an additive Gaussian white noise term was included in the equation for $\dot{V}$:

$$\dot{V} = \cdots + \sigma\eta\left(t\right), \tag{3.26}$$

where σ is the standard deviation of the noise and $\eta\left(t\right)$ is the Gaussian distribution with mean 0 and standard deviation 1, and the sampling of the distribution is independent and identically distributed such that $\langle\eta\left(t\right)\eta\left(s\right)\rangle = \delta\left(t-s\right)$. Numerical simulation for the noisy oscillators was carried out using a 4th order Runge-Kutta solver adapted for noise. The 2nd order solver was explicitly derived in [68], and the derivation there can, as noted in the original paper, be extended to higher order Runge-Kutta solvers; a 4th order adaptation is presented in [69]. Differing magnitudes of noise were added to the simulation; as can be seen in Fig. 3.11, even for highly noisy systems, the control sequence developed under the no-noise condition retained success in achieving the appropriate clustering.

3.4.4 Control in the Presence of Coupling

The other important measure of robustness is how well the control maintains validity when we consider interactions between the neurons. To this end, we carried out the same protocol as previously but included all-to-all electrotonic coupling in addition to noise. For all trials, the standard deviation σ of the voltage noise was set equal to 5 mV. For

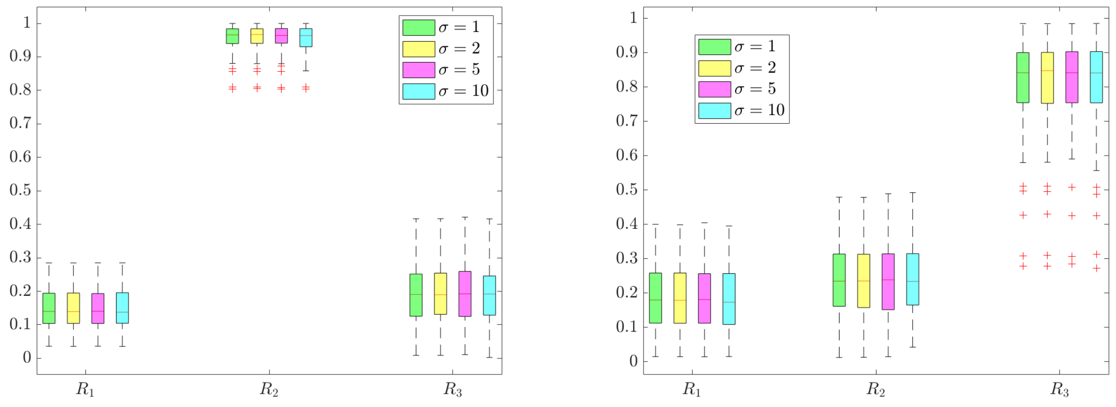
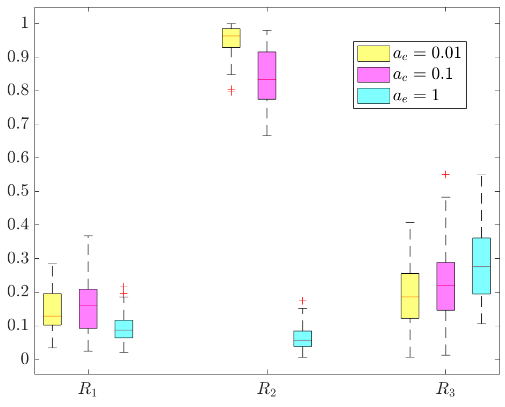


Figure 3.11: Boxplots for 50 trials with additive Gaussian noise of differing magnitudes for two clusters (*left*) and three clusters (*right*). Even for the high-variance case $\sigma = 10$, the clustering control sequence is still successful in generating clusters, indicating the robustness of the control algorithm.

the electrotonic coupling, we again use the equation presented in (2.24) and considered different values of the coupling strength a_e:

$$\Delta \dot{V}_i = a_e \sum_{j=1}^{N} (V_j - V_i)$$

Otherwise, as in the case of considering noisy oscillators, the neurons were stimulated using the same stimulation parameters developed for the noise-free, uncoupled scenario. Three levels of coupling were considered: $a_e = 0.01$, $a_e = 0.1$, and $a_e = 1.0$, which correspond to weak, moderate, or strong coupling, respectively. As can be seen in Fig. 3.12, weak coupling has little impact on the results, while the control sequence still generally performs well with moderate coupling (though performance is degraded when compared to the uncoupled baseline). We additionally note that the 2-cluster control objective outperforms the 3-cluster objective, demonstrating it is easier to achieve fewer clusters. This is particularly pronounced in the case of moderate coupling, where there is far less of a dropoff in performance in the case of two clusters than for three, while strong



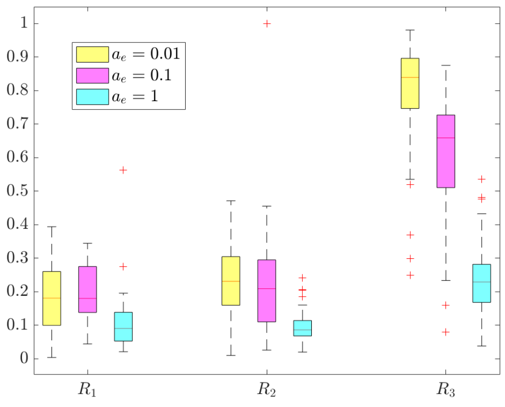


Figure 3.12: Boxplots for weak, moderate and strong coupling for two clusters (*left*) and three clusters (*right*). The results behave as expected, with the predictions of the neural network performing worse as coupling strength increases. Despite this, the control strategy shows significant resilience in the presence of weak to moderate coupling, with clear elevations in the appropriate order parameter compared to the other order parameters.

coupling is impossible for the control sequences for either objective to overcome without modification. This indicates that when only moderate or weak coupling is expected, we need not modify our approach, but in applications where the coupling strength is expected to be much larger, it may be necessary to supplement the base model with some additional estimation of the effect of coupling. This is in contrast to the influence of noise, which showed far less effect on the outcome even for large values of the noise. Most likely, this is because noise does not introduce new stable configurations in the same way coupling does and its average effect is 0 in the case of white noise, independent of the magnitude of the standard deviation. Therefore, noise is roughly as likely to cause a neuron to fire earlier than anticipated as it is to cause the neuron to fire later than anticipated, so in the aggregate, its effect on the accuracy of the control is less pronounced. We would likely see larger effects if the neural network used the voltage traces of the neurons directly in making predictions, but since only the firing times are recorded, individual fluctuations have little effect on the overall system.

3.5 Conclusion

We have successfully trained an artificial neural network to make accurate predictions about the change in spike time of a neuron subject to a square wave stimulus using minimal input data. The network displayed a high degree of accuracy across both its training set and a cross-validation set, with average error amounting to less than 1% of the total natural period. Using this model we then predicted the optimally efficient input for desynchronizing a pair of neurons given a cost function to be minimized and an initial separation. Though the network's model struggled with low-amplitude inputs and low-separation conditions, it was nonetheless accurate over a large range of parameter values. This relative error for low-amplitude inputs, which approaches unity as $I_0 \to 0$, can be understood as the result of the limits in the system's resolution; the neural network predicts a small shift in the next spike time, but in simulation no shift is observed. Increasing sampling of low-amplitude signals would likely improve these predictions, as would bagging with a secondary model specifically trained on low-amplitude signals.

We then applied our neural network to solving two traditional control techniques in neuroengineering: desynchronization utilizing dynamic programming and clustering with a single underactuated input. Using the performance benchmarks from the prior efficiency analysis, we were able to select parameter spaces wherein the predictions from the neural network maintained a high degree of fidelity when transferred to the ODE model. The application to these two stylistically and mechanically distinct problems highlights what we perceive as another strength of our approach: a high degree of flexibility in application. Although an ANN may take some time to train to the point where it may be considered reliable, the computational cost of applying the model once trained is substantially lower than alternative approaches, and the broad range of applications serves to increase the benefits of using the neural network's regression model relative to the initial cost of training it. Additionally, the model demonstrated significant robustness in the presence of both noise and weak-to-moderate coupling without requiring any adjustments in the protocol, a level of generalizability that is not present in the work in Chapter 2.

Chapter 4

Machine Learning and Parkinson's: Practical Analysis and Control Design

While the network developed in Chapter 3 was able to successfully desynchronize the population of oscillators even when noise and coupling effects were added, the type of data it used for analysis is more akin to the types of readings that are possible *in vitro* rather than *in vivo*. Additionally, even relatively small regions of the brain, such as the subthalamic nucleus, have far more than 50 neurons, so getting accurate readings from individual neurons – even something as limited as when a spike occurs – is not generally feasible except in highly controlled environments. More commonly, the types of readings we can expect to get from contact electrodes are known as *local field potentials* (LFPs). We can view this as "zooming out" our view of the brain when we consider the question of scope: the LFP can be viewed as, in some ways, an aggregation of information about the activity of neurons across the region of the brain, so when those neurons fire in concert, we see elevations in the power in the LFP. We often consider the LFP not in the time domain but rather the frequency domain, converting it into a representative power spectral density (PSD); if neurons are oscillating together, they will fire with a consistent period and we should see an elevation in the appropriate frequency interval of the PSD. As mentioned previously, in Parkinsonian patients, this is the "beta band," corresponding to the interval of approximately 15 to 35 Hz.

LFP data defies the simplfied analysis of a phase model reduction or straightforward input-output relationships that we explored in Chapters 2 and 3. Instead, we will develop throughout this chapter a predictor that estimates the likely behavior of the LFP's PSD through the use of a feature extraction technique in machine learning. Using this feature extraction, we will first analyze patient data provided by a research group at UC San Francisco, then see how we can use LFP signal data in conjunction with our feature-extracting autoencoder to create an adaptive control scheme to modulate the behavior of the LFP, even as a patient's physiology changes.

4.1 Parkinsonian Patient Data, Neural Network Design, and Simulated Data Generation

4.1.1 Patient Cohort Description

Data was collected from basal ganglia electrodes implanted in 6 patients. Measurements were recorded over the span of, on average, 21 sessions per patient totaling an average of 26 minutes and 50 seconds per patient. All measurements were taken at rest. Data was collected using the Activa PC+S (*Medtronic, Inc.*); the capabilities of the sensor have been thoroughly described in previous work, cf. [70, 71]. All data were sampled from one bipolar contact pair in the STN with a sampling frequency of 800 Hz in the time domain; a second recording bipolar contact pair in the primary motor cortex was simultaneously sampled in the time domain but is omitted from this analysis. More information on the surgical procedure and collection protocol is available in [72].

Patients ranged in age from 45 to 62 years and in duration of disease from 4 to 15 years.

4.1.2 Neural Network Design

To carry out our analysis of the LFP signal and its correspondence to pathological activity, we first constructed an autoencoder to encode the key features of the power spectral density (PSD) of the incoming LFP signal. Because the amount of data available for developing a predictive model is relatively small compared to traditional ML problems and there may be significant physiological differences from patient to patient, we wished to attempt to mitigate the possibility of overfitting by projecting onto a small number of variables in a large-dimensional space. In this application, the autoencoder is similar to principal component analysis (PCA) [73], or dynamic mode decomposition (DMD) [74], with the advantage of allowing for nonlinear projections.

A generic diagram of an autoencoder network is shown in Fig. 4.1. The nonlinear projection to the encoded variables is found by minimizing the mean-squared error reconstruction error of this network; the remainder of our analysis will utilize these encoded variables rather than the raw PSD signal. In particular, we implemented a 1-dimensional convolutional autoencoder to preserve the correlational relationships between adjacent frequency readings. A summary of the autoencoder is provided in Table 4.1. The LFP data was first normalized to a mean of 0 and standard deviation of 1, then passed into the autoencoder for training. The autoencoder was validated on a separate dataset. The reconstruction error for the validation set is shown in Fig. 4.2. Once the autoencoder was trained, the encoder was extracted to use as the basis for our analysis. The encoded

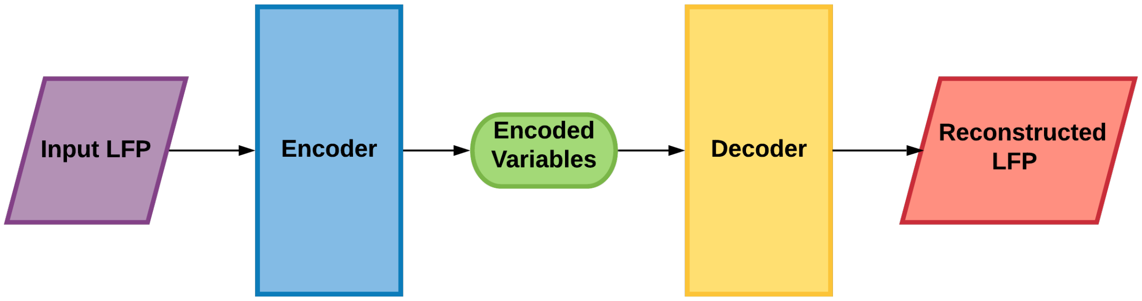


Figure 4.1: Generic autoencoder structure.

Item	Value
Convolutional Layers-Encoder	3
Trainable Parameters-Encoder	2099
Trainable Parameters-Decoder	8099
Encoded Variables	3

Table 4.1: **Autoencoder model summary.**

system consisted of three variables, reduced from an original frequency range of 20 Hz with 1-Hz intervals (15-34 Hz).

We pause here to briefly describe the correspondence of features of the incoming signal to values of the encoded variables. The encoding of a representative sampling of the data is shown in Fig. 4.3, with the value of the extracted feature depicted by the color for each of the sampled input beta-band PSD signals. While the precise relationship between the variables and the input data is complex and nonlinear, we generally assert that high values of Feature 1 correspond to high power in the middle of the beta frequency range, high values of Feature 2 correspond to high power in low end of the beta frequency range, and high values of Feature 3 correspond to high power at the high end of the beta frequency range. We can confirm this by considering the regression of linear mappings of the variables to frequency power; the correlation between features and this power is shown in Fig. 4.4. This reveals the high linear correlation in different portions of the

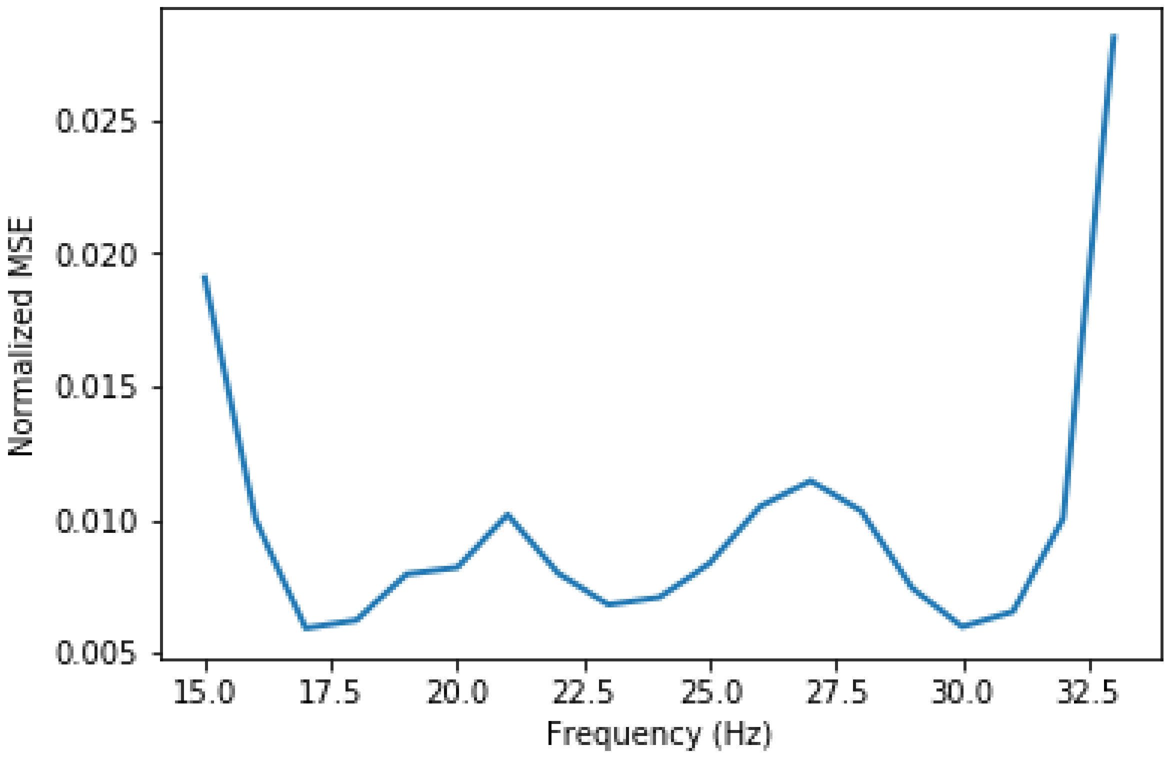


Figure 4.2: Autoencoder mean-squared error reconstruction error for normalized input ($\mu = 0$, $\sigma = 1$). The autoencoder faithfully extracts key features of the LFP's power spectral density and successfully reconstructs the original PSD.

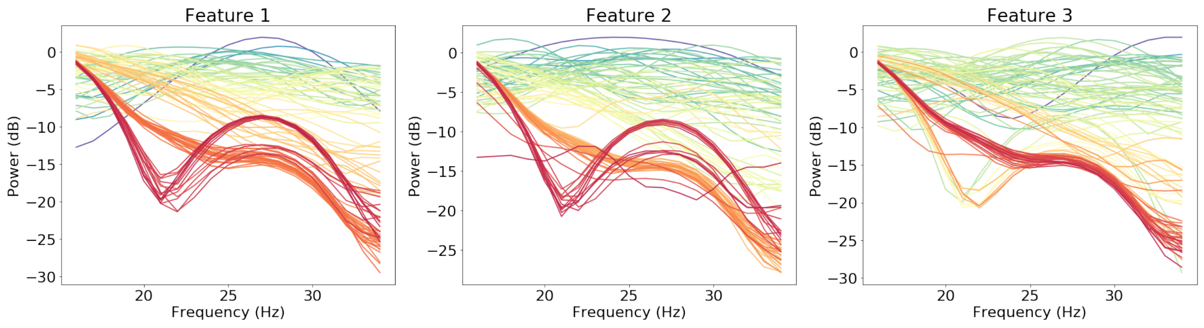


Figure 4.3: Relative values of encoded variables for a sampling of PSD signals. Red denotes the lower end of the value range, while blue indicates the higher end.

frequency range for each of the encoded variables. We can additionally see that the first and third variables may additionally encode information about the overall shape of the PSD; we see this by examining the average first differences of the PSD and noting the strong linear correlation to the first and third encoded variables (see Fig. 4.5).

Our implementations with both patient data and simulated data then used this encoder as the basis for making a prediction about the state of the system at some future time. In working with both the simulation data and patient data, this corresponded to a regression problem. Each individual time snapshot was labeled as possessing a maximum amplitude in the beta band above or below a critical threshold (for the patients, this threshold was the upper quartile value of the local maximum across all snapshots); all of these individual snapshots within a given time interval were then used as the basis for the regression. An example of this labeling is shown in Fig. 4.6. In all cases, the average proportion of slices above the threshold was used as the target for prediction. We considered a slight relaxation of the burst classification in prior literature – rather than considering solely continuous periods above or below the threshold to classify as burst or non-burst, we carried out regression on the maximum moving average proportion above the threshold. For example, we slid a 0.25-second window across each 1-second labeled sample (with each time step labeled with either a 1 if above the threshold or 0 if below)

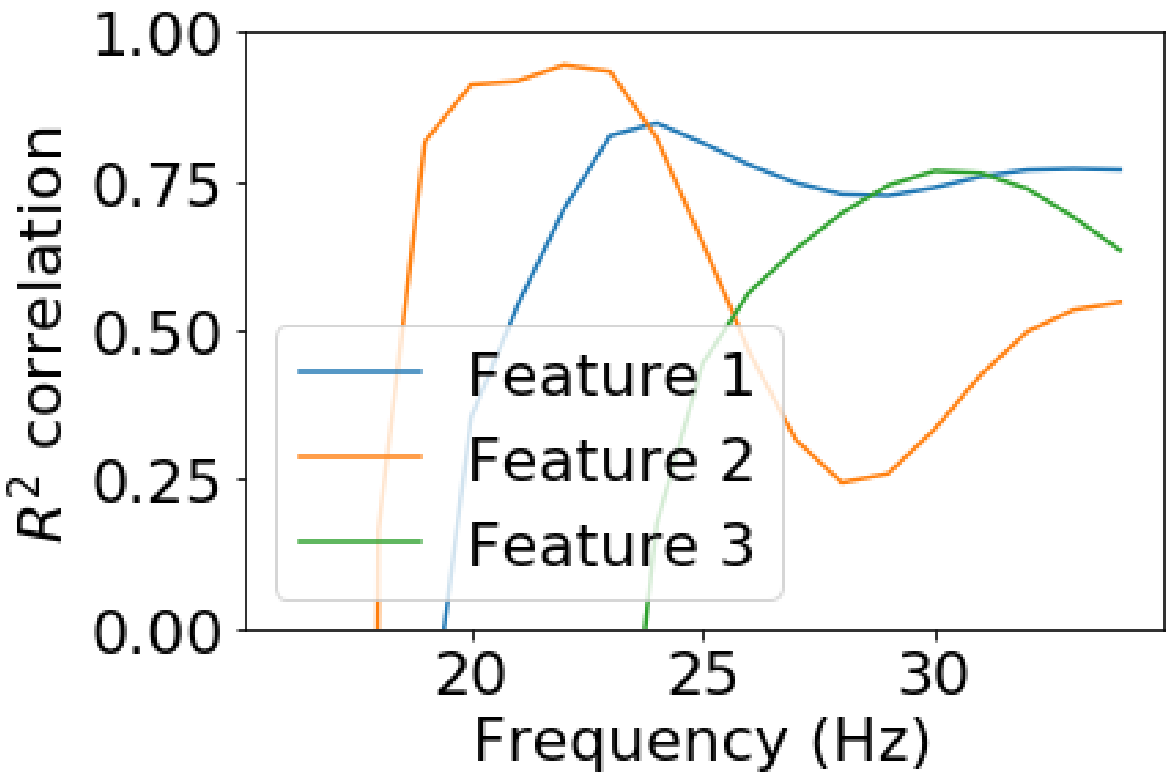


Figure 4.4: R^2 values for a linear regression of each feature at each frequency. Where the curves are not visible, their values are below 0, meaning the linear regression is outperformed by a constant-value hyperplane. As can be seen, in different subsets of the frequency range, the encoded variables show strong linear correlation, with Feature 1 in the medium-high frequency range, Feature 2 in the low-medium frequency range, and Feature 3 at high frequencies.

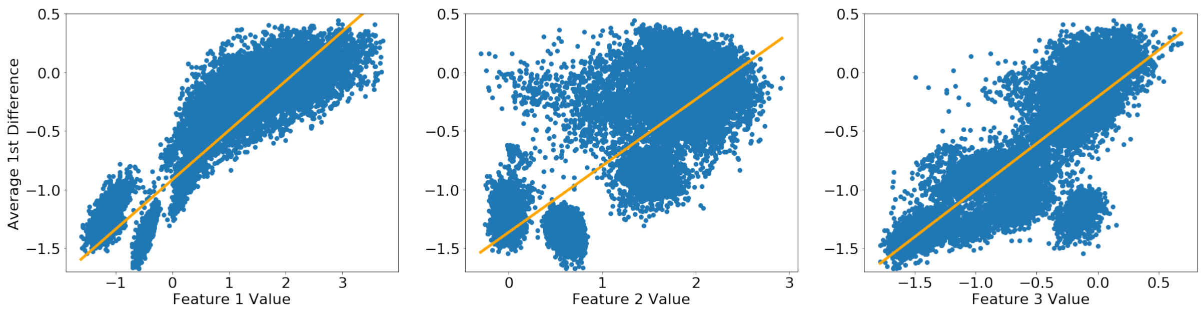


Figure 4.5: Average (mean) first differences versus value of each feature across samples. The orange line denotes the line of best fit for each linear fit. Both the first and third features show a clear correlation between encoded value and first difference, while the trend for the second encoded variable is distinctly less clear.

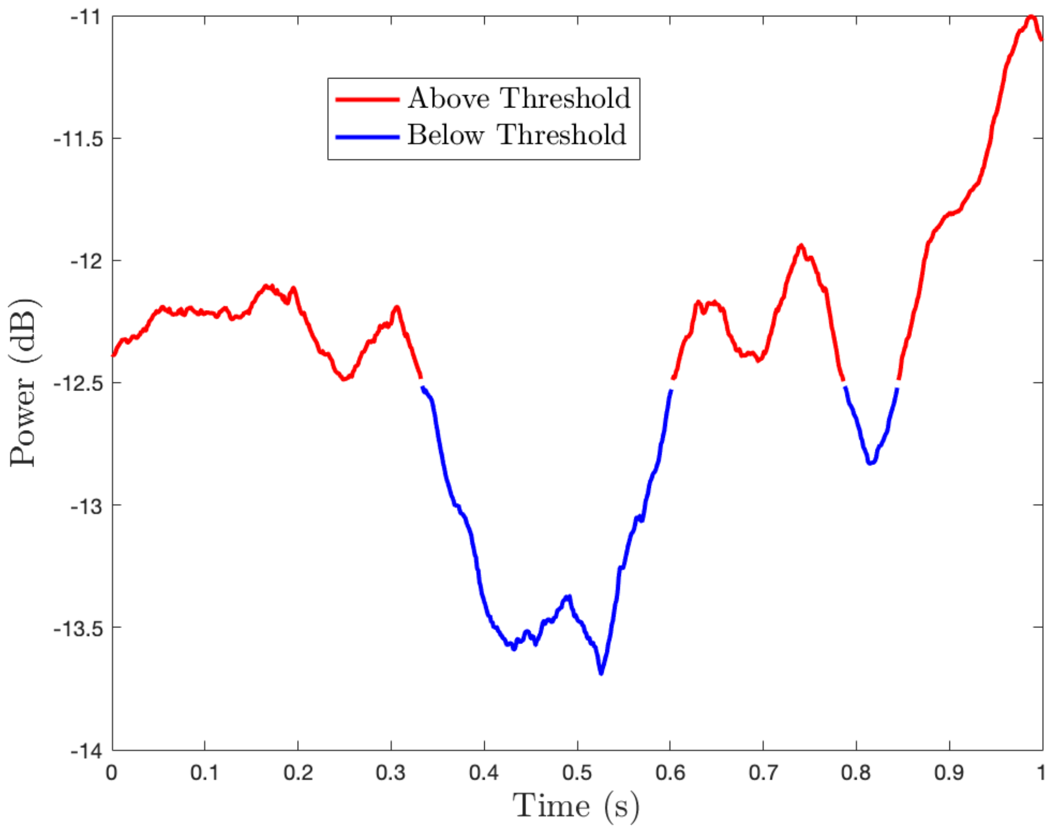


Figure 4.6: Labeling of the peak for snapshots in a 1-second sample, with red denoting above the target threshold and blue below. Prolonged periods above the target threshold indicate bursting activity.

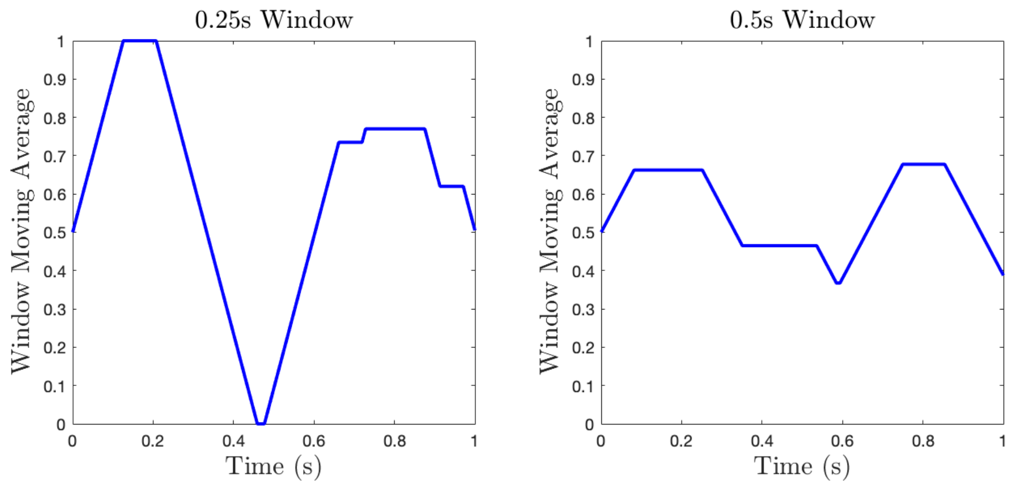


Figure 4.7: Moving average for two different sliding window sizes: 0.25 seconds (*left*) and 0.5 seconds (*right*). The sliding window was applied to the labeled data for the interval depicted in Fig. 4.6. As can be seen, there is a clear short burst at the beginning of the interval, and two short bursts near the end of the interval with a slight dip below the threshold. As a result, the shorter, 0.25-second window is maximized at the beginning of the interval at a value of 1.0. In contrast, the dip ensures the lack of a second maximum in the 0.25-second window, but the elevation in the 0.5-second window indicates a sustained period above the threshold, even if the continuous time above the threshold is shorter. For purposes of analysis, the first window would return a value of 1.0, while the second would return a value of 0.7.

and calculated the mean value within that window. We then considered the entirety of the 1-second sample and selected the maximum value of the moving average within that window. We considered four different-length windows, ranging from 0.25 seconds to 1 second in increments of 0.25 seconds. These differing windows can be viewed as corresponding to identifiers for bursts of varying length – a proportion near 1 in the short window may mean a short burst, while a proportion near 1 in a longer window corresponds to a longer burst. For example, the differences in the processing of the same data for two different window sizes is shown in Fig. 4.7, demonstrating how using multiple-sized windows simultaneously can allow for discrimination between short, medium, and long bursts. We envision these windows as classifiers that may be used either in conjunction or isolation, depending on the desired objective.

4.1.3 Simulated Data Generation

Simulated PSDs were generated in Python based on simulation of the local field potential readings for a dopamine-depleted individual, using parameters from the basal ganglia-thalamocortical LFP model presented in [75, 76]. The code used was adapted from [77], with modifications to allow for additional data collection and parameter control, as well as to integrate the neural networks for encoding and supervised learning.

4.2 Burst Prediction from Parkinsonian Patient Data

4.2.1 Prediction Model Single-Patient Training and Validation

Trials were run for the different averaging windows described above and with different "relative gaps" between the starting points of the input data and output prediction data. That is, for the window used in calculating the power spectral density, the relative gap is the increment between starting points relative to the length of the PSD calculation window, which was equal to 1 second, or 796 snapshots, for each trial here. In this context, a relative gap of 0.5 corresponds to 50% of the time-series data being shared between the input and output, while a relative gap of 1.0 is the minimum gap such that no data is shared between the input and output time-series windows. Results are presented here for relative gaps ranging from 1.0 to 2.0 for each averaging window. Since the interval was 1 second here, a relative gap of 1.0 indicates a delay of 1.0 second, and so on.

As a baseline for comparison, we selected a "no-change" hypothesis – that is, that the proportion of snapshots tagged as containing a burst signal in the previous window is the same as in the current window. This was selected for two primary reasons. First, the no-change hypothesis can be calculated explicitly from the input data provided to the network, and as such operates from a certain "level playing field" with our neural network. Second, the no-change hypothesis performs quite well because starting points are randomly selected: below a relative gap of 1.0 the no-change hypothesis approaches perfect accuracy as the relative gap approaches 0, and as we found, a significant proportion of the data (greater than 25%) is still well-explained by this hypothesis even with a larger relative gap.

We carried out initial training and validation on a single patient, randomly sampling from 22 separate recording sessions for that patient, for a total of 28 minutes and 30 seconds of measurement time. The measurements were conducted with the patient at rest, and the sessions took place over the span of 2 years. Samples contained instances of the patient both on and off medication, and with stimulation either on or off as well. All data were collected via an electrode inserted at the basal ganglia. These trials were subdivided into four conditions, consisting of the differing combinations of on-off medication and on-off stimulation. Each condition was sampled from equally, with 5,000 random samples pulled from each subdivision for a total of 20,000 random samples for training. To minimize computation time, training was accomplished via minibatch gradient descent in a 3-hidden-layer neural network for only 5 epochs; we note that accuracy continued to improve incrementally beyond 5 epochs, but 5 was sufficient to demonstrate appreciable improvements over our baseline. The neural network was provided with 6 inputs: the three encoded variables representing the power spectral density of the LFP at the end of the current time window, the max burst proportion in the current time window, the medication state (1 or 0), and the stimulation state (1 or 0).

The mean absolute error in predictions on the validation set, as well as the first and third quartile errors, are shown in Fig. 4.8. As was previously alluded to, across all trials, the lower quartile of the data (that is, the most accurate 25% in terms of prediction error) possessed an error near 0 for the no-change hypothesis, and was functionally unchanged by the prediction model. However, appreciable gains are seen in the mean error and the upper quartile. Despite the high accuracy of the no-change hypothesis on a significant portion of the dataset, the model still saw reductions of between 10% and 20% in the mean absolute error across trials, with the gains increasing as the averaging window size and relative gap increased. Reductions in the upper quartile error bound were even more dramatic, with reductions ranging from 10-20% in panels (a)-(c) while reaching as high as 35% in the 1.0 second averaging window (panel (d)). These gains in absolute error are shown in Fig. 4.9.

The prediction model improves accuracy primarily by mitigating the largest sources of errors in the no-change hypothesis. Across all four averaging windows, the most likely measurements were values near either 0 or 1, with a smaller peak around 0.25. This can be seen in the histograms presented in Fig. 4.10, which shows the composition of a representative trial for each averaging window (because initial indices are randomly selected, the relative gap does not influence the population statistics). Naturally, this indicates that the highest concentration of solutions should be at either extreme; less immediately apparent is that, for any given burst proportion observed in the current window, the most likely outcome in the subsequent window is to reach a value of either 0 or 1. This is evident by looking at the histogram of errors for the no-change hypothesis presented in Fig. 4.11, with the characteristic "X" persisting across trials. As can be seen in the same figure, our prediction model successfully eliminates these high-error terms, with a corresponding increase in the success of predicting when a transition to 0 or 1 will occur and, more importantly, in which direction it will occur.

We assert here that the prediction model is, in fact, using the encoded LFP signal in producing its prediction and not simply replicating a Markov decision process strategy utilizing the other input data it receives (current burst proportion, stimulation state, and medication state). To demonstrate this, we observe the model's predictions when

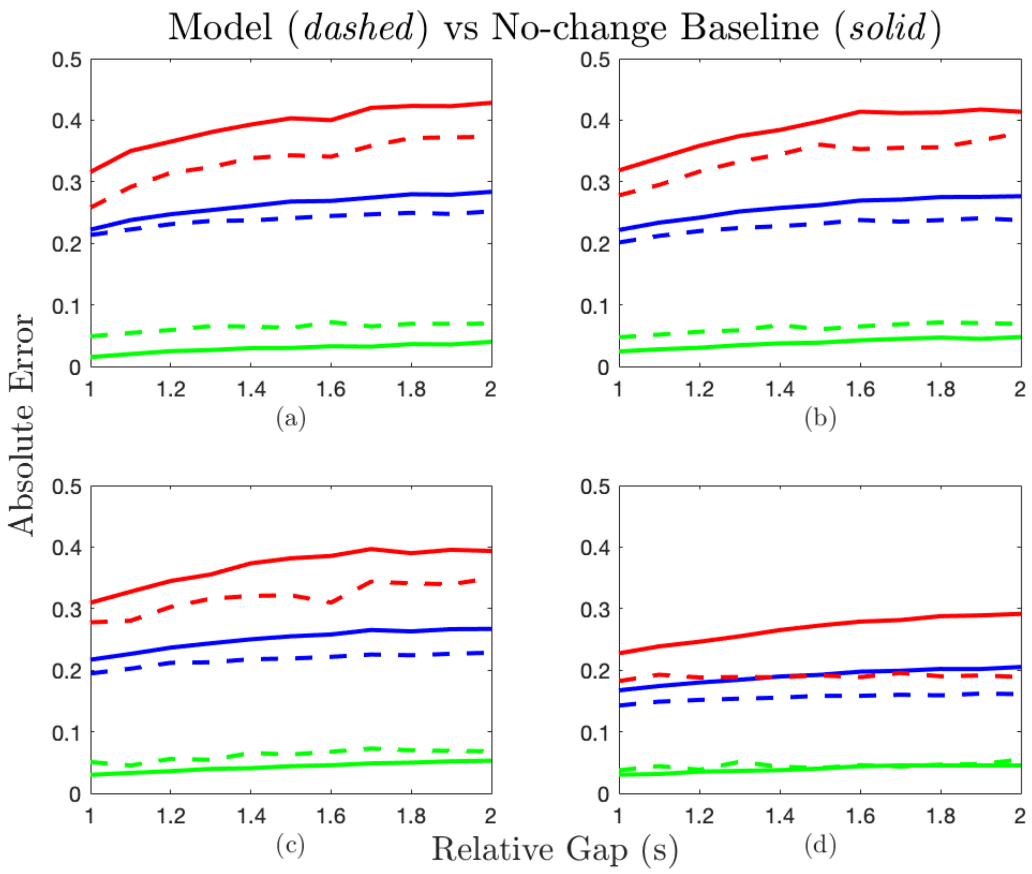


Figure 4.8: Model (*dashed*) versus no-change hypothesis baseline (*solid*) for averaging windows of length 0.25, 0.5, 0.75, and 1.0 seconds (panels (a)-(d), respectively). Three measurements of the error are shown: the mean absolute error (MAE, *blue*), the bound of the lower quartile of error values (*green*), and the bound of the upper quartile of error values (*red*).

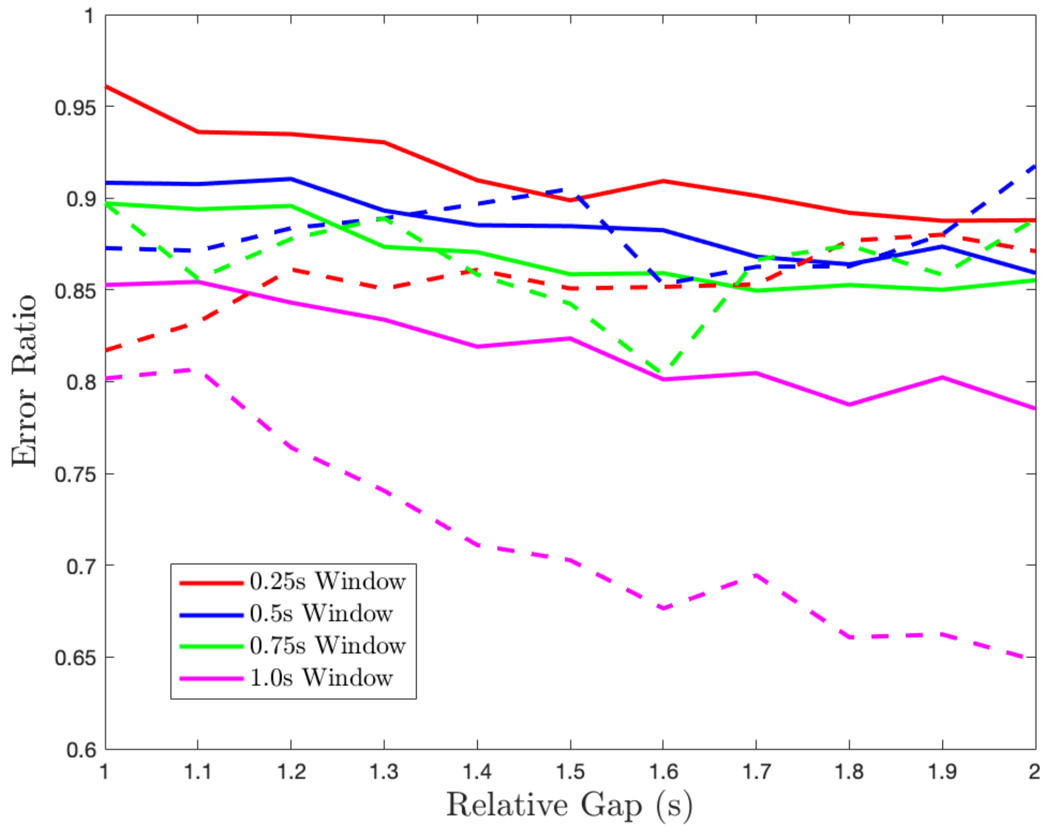


Figure 4.9: Ratio of model error to baseline error for different averaging windows. Both mean absolute error ratios (*solid*) and upper quartile error ratios (*dashed*) are shown.

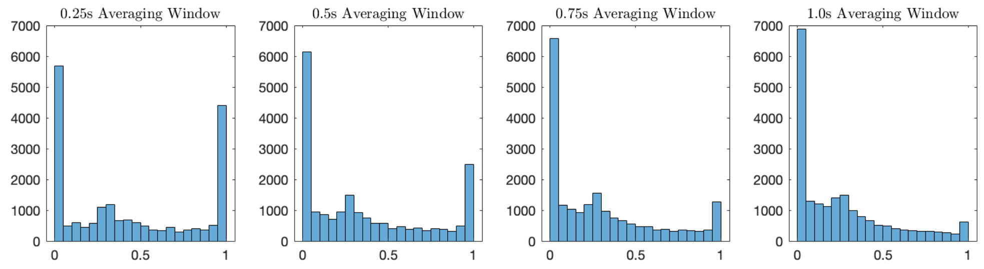


Figure 4.10: Histograms of burst proportions for each averaging window. The total number of samples in each histogram is 2,000. As can be seen, across all windows the single most likely value is 0, with the likelihood increasing as the averaging window length increases. Conversely, the likelihood of a value near 1 decreases as the averaging window length increases, eventually ceasing to represent the second-most likely outcome in the 1.0s averaging window. A peak is observed in all 4 histograms at 0.25 because the threshold for classification of a burst is a maximum above the third quartile threshold, so in an "average" window, 25% of snapshots should be above the threshold.

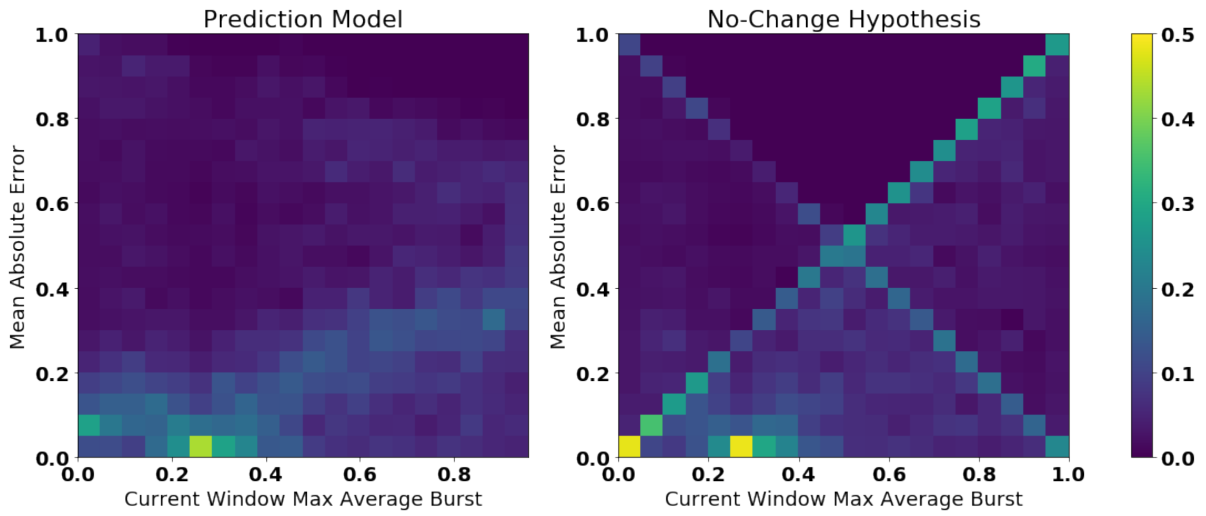


Figure 4.11: Normalized histogram of errors from neural network model and no-change hypothesis (averaging window 0.5s, relative gap 1.5). Because of the uneven distribution of burst proportions seen in Fig. 4.10, for enhanced readability these histograms are normalized such that the sum of values in each column is 1. In other words, if a particular grid space in the 0.2 column has a value of 0.3, that means that 30% of samples that have a value of 0.2 in the current window possess that corresponding error in the prediction. The "X" shape indicates most intermediate states transition to either a 0 or 1; if the no-change hypothesis has an error equal to the current burst proportion, the new burst proportion must be 0, whereas if the error is equal to its complement, the new burst proportion must be 1.

all variables aside from the encoded variables are held constant. Fig. 4.12 illustrates this with the probability density function for one particular instance of this; there is a clear distribution of outcomes dependent on the encoded variable values that could not be replicated without these values. Furthermore, while the stochastic nature of the system inhibits perfect accuracy, we see in Fig. 4.13 that on average, these predictions are consistent with the results.

We can also look more specifically at what features map to what outcomes; while the encoded variables will not directly align with outcomes (since the other variables – burst proportion, medication state, and stimulation state – also influence the result), we can see some clear trends in the data. Most notably, there is a clear "low-probability zone" across window sizes shown in Fig. 4.14 for roughly average values of the first feature and low values of features 2 and 3. If we consider our previous analysis of the feature extraction in Figs. 4.3 and 4.4, this corresponds to a negative average 1st difference and low amplitudes in the middle of the frequency range.

4.2.2 Generalization of Model to Other Patients

The primary goal of implementing the autoencoder in our model was to reduce the risk of overfitting by extracting the key features of the data, rather than using the raw power spectral densities. When passing data through an encoder, some loss of information is to be expected; indeed, if we are only considering data the network is explicitly training on, we would be better suited by using the PSDs as-is rather than adding the step of the encoder. But our hope was that including the encoder would improve the transference of the learning of the model to novel patients after training on a single patient, and this hope was confirmed by our application of the model to a larger cohort of patients. To assess the benefits of the autoencoder, we considered two additional models beyond our

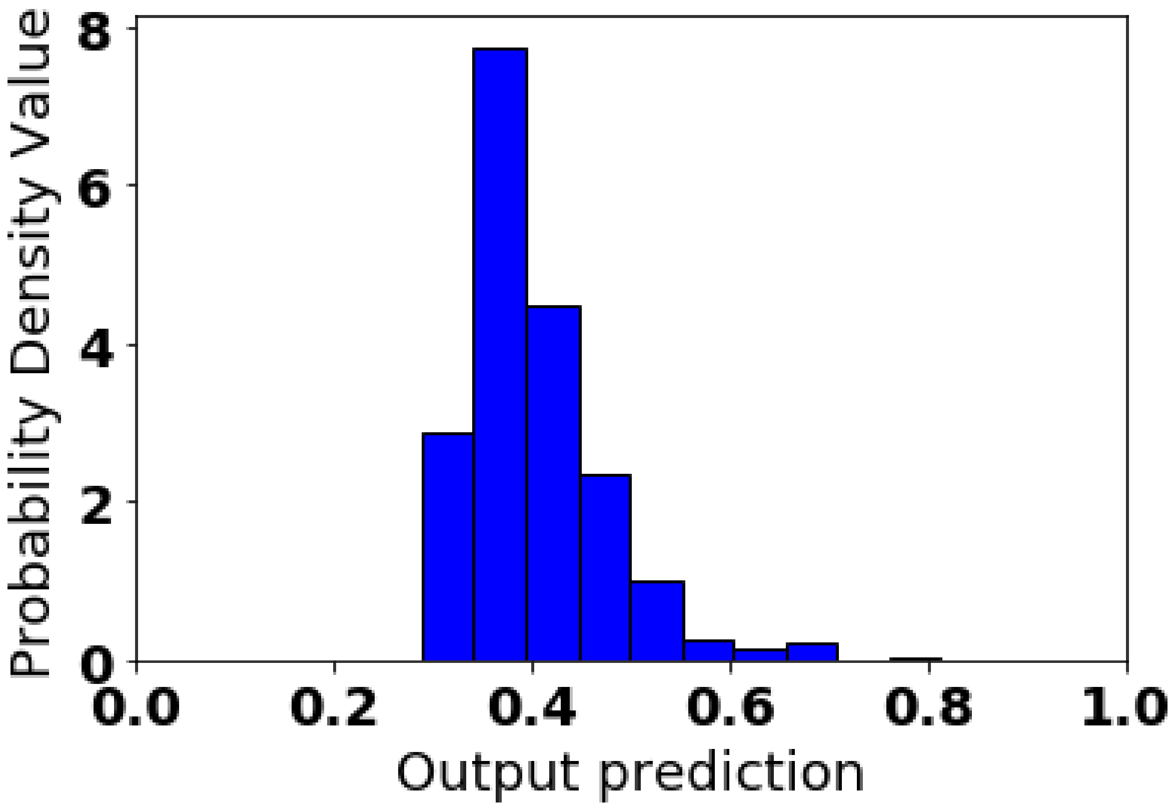


Figure 4.12: Histogram representing a probability density function. The histogram represents a subset of the outcomes for the neural network model trained on an averaging window length of 0.5s and a relative gap of 1.5, and it is normalized such that the area represented by the bars is equal to 1. This histogram specifically represents predictions from the portion of the data for which medication was off (0), stimulation was off (0), and the previous burst proportion was greater than 0.9. All variation in these outcomes is directly attributable to the encoded variables, rather than the other three input variables.

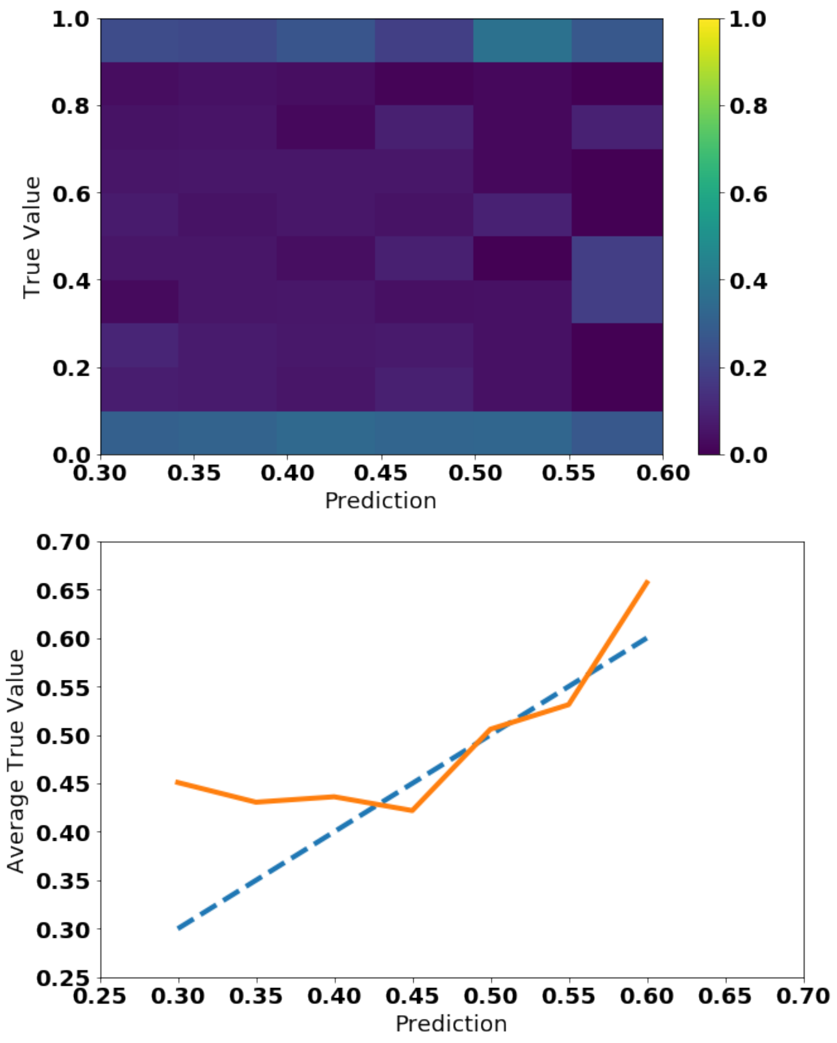


Figure 4.13: Normalized histogram of predictions and true values (*top*) and average true value by model prediction (*bottom*) for the prediction model for a specific subset of values. The same data is used as in Fig. 4.12, with medication off, stimulation off, and previous burst proportion greater than 0.9. The relative gap was 1.5 and the averaging window had a length of 0.5s. The upper panel shows that the majority of true values are either close to 0.0 or 1.0, but the apportionment between the two extremes varies across the prediction value. For the average true values, the orange curve represents the mean for the data, while the dashed blue line indicates perfect averaging, where the predicted burst value exactly matches the average true burst value. The close adherence to this dashed blue line shows that, although the inherent stochasticity of the system makes it difficult to predict either extreme with certainty, the averaged prediction hews closely to the expected value.

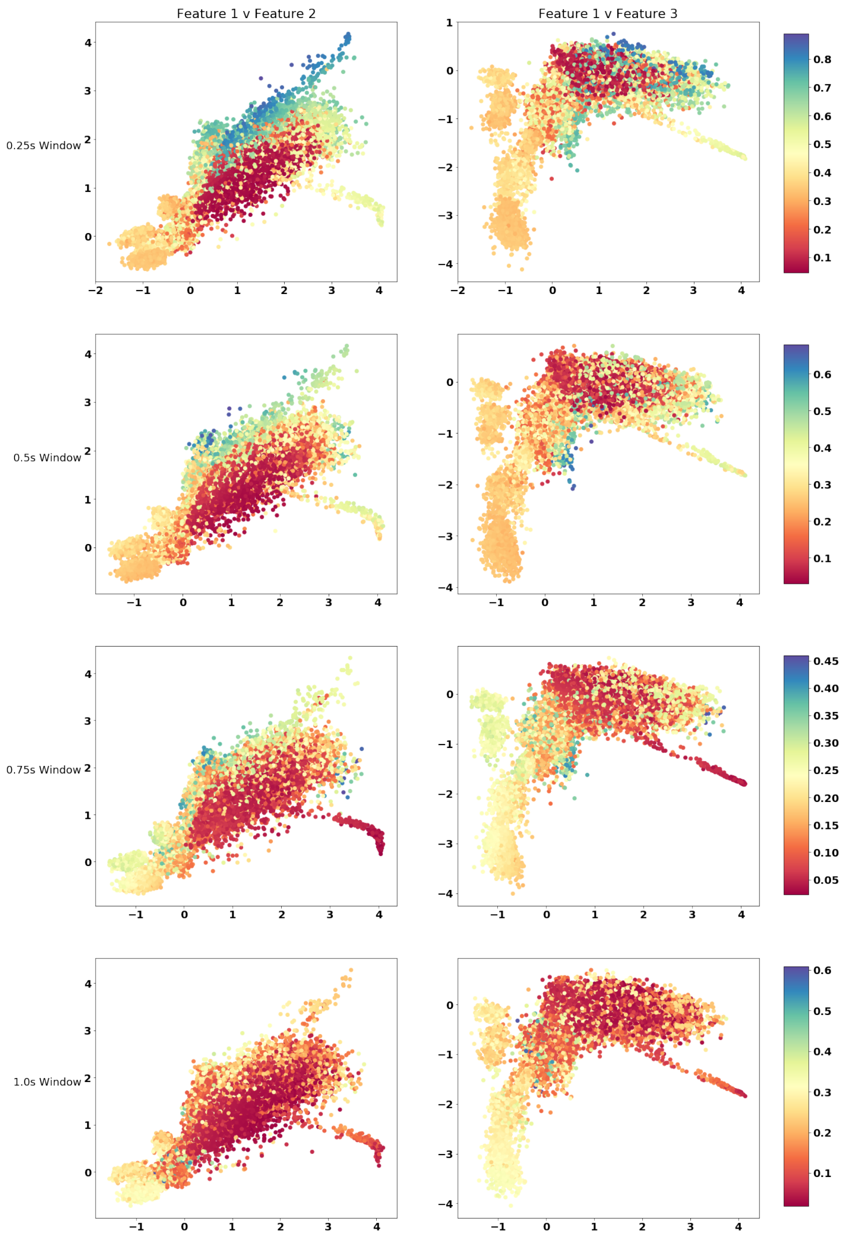


Figure 4.14: Scatter plots of feature values and predictions for proportion of the time above threshold (indicated by color). For all panels, the x-axis denotes the value of feature 1, while in the left panels the y-axis is feature 2 and in the right panels the y-axis is feature 3, as denoted at the top of the figure. The relative gap for all four rows is 1.5s. All four gaps show clear regions of low probability, such as midrange values of Feature 1 with low values of Feature 2, and high probability, such as high values of Feature 1 and Feature 2 for the 0.25s window.

neural network in comparison: a support vector machine that used the raw PSD signals in addition to the other inputs, and an alternative support vector machine (SVM) [78] that utilized the encoded variables instead. We again compared all three models to our no-change hypothesis, and both SVMs were implemented via Python's scikit package and trained on the same single-patient data as the neural network. As can be seen in Figs. 4.15 and 4.16, not only does the prediction model outperform all other models and continue to maintain an advantage over the no-change hypothesis, but the SVM using the extracted features also outperforms the SVM using raw PSD signals across most trials. From this, we assert that utilization of an autoencoder for feature extraction may be useful in novel ways as further research is conducted on predicting burst behavior and should be considered a valuable area of further investigation.

Lastly, we make two observations about the overall accuracy of the network when applied to the unseen patient data. First, while the mean absolute error is higher than for the original, same-patient validation data, so too is that of the no-change hypothesis, suggesting the initial patient possessed fundamental physiological differences from the "average" patient in the cohort. Secondly, the relative improvement of the neural network (particularly with relative gaps nearer to 2.0) over this baseline (see Fig. 4.17 compared to Fig. 4.9) suggests the feature extraction is robust, however, and these physiological differences do not fully undermine the bases for prediction resulting from the prediction model. We believe this robustness is a promising indicator of the potential for a generalized burst predictor not requiring re-training on a patient-to-patient level, which is likely infeasible.

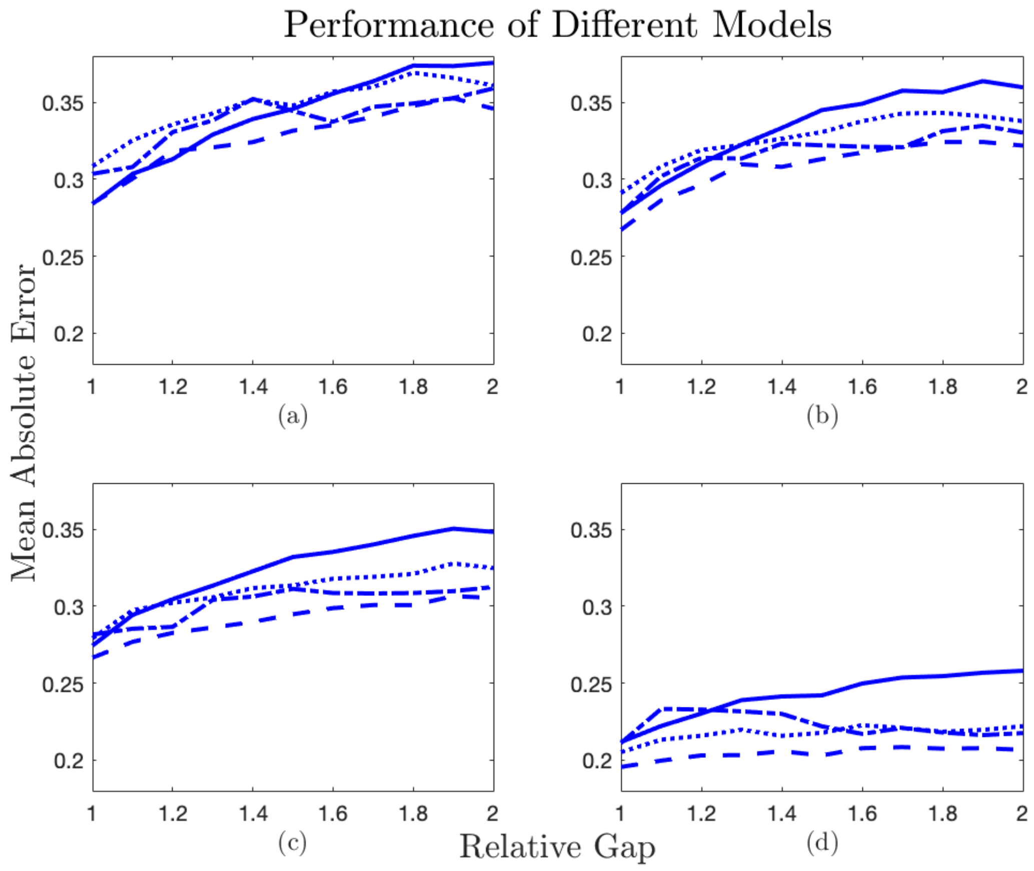


Figure 4.15: Mean absolute error (MAE) for averaging windows of 0.25-1.0s ((a)-(d), respectively). The no-change hypothesis is again denoted by the *solid* line, while the prediction model is represented once more by the *dashed* line. The SVM without autoencoding is given by the *dotted* line, while the SVM with autoencoding is represented by the *dash-dot* line.

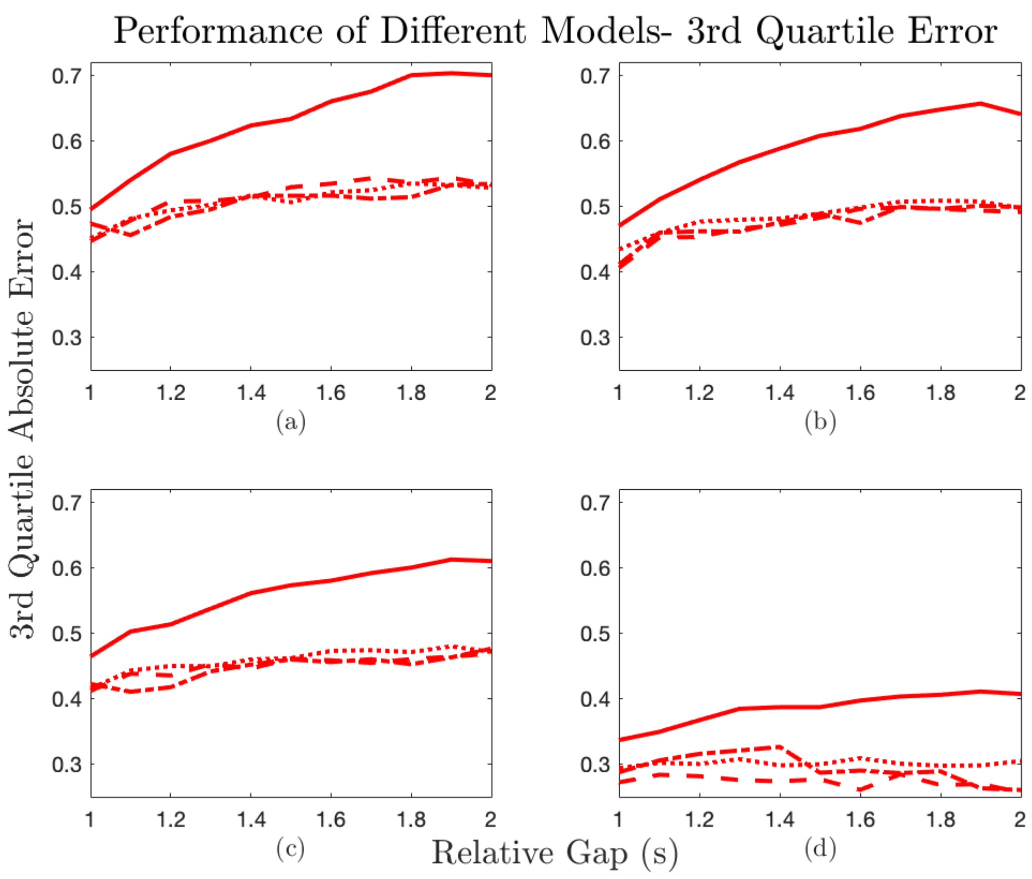


Figure 4.16: Upper quartile bound for averaging windows of 0.25-1.0s ((a)-(d), respectively). As in Fig. 4.15, the different lines represent no-change (*solid*), prediction model (*dashed*), SVM without autoencoding (*dotted*), and SVM with autoencoding (*dash-dot*).

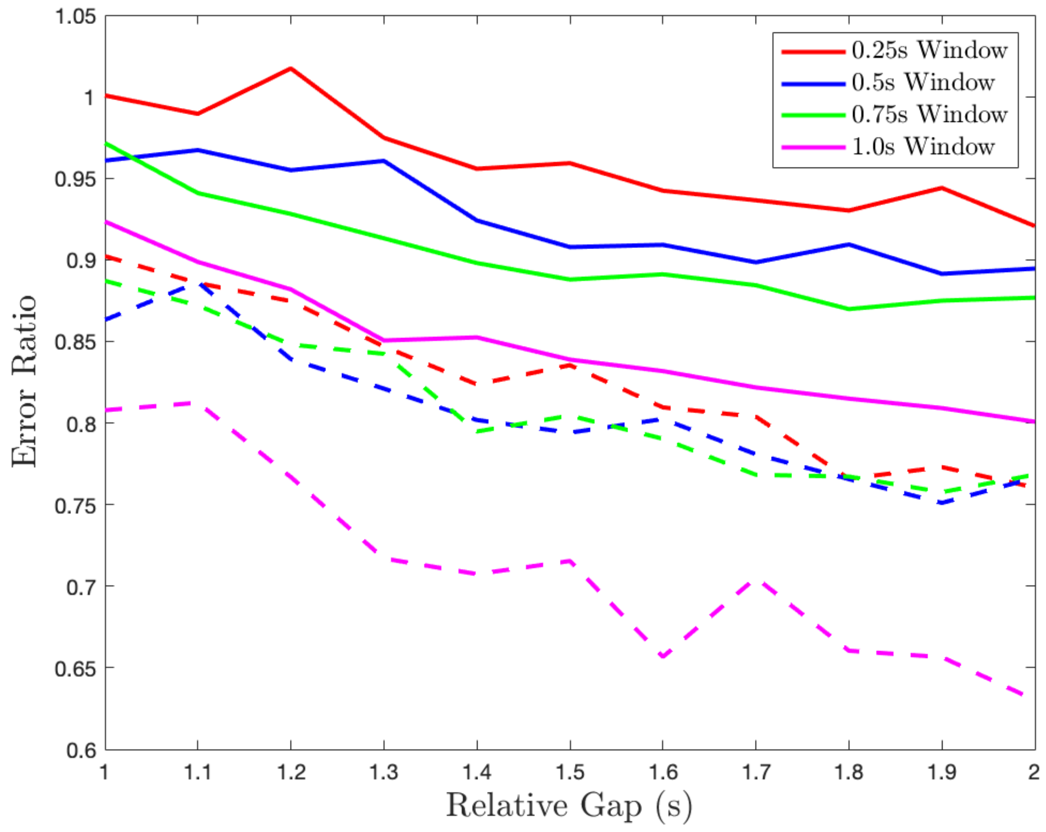


Figure 4.17: Ratio of prediction model error to baseline error for different averaging windows when applied to unseen, multi-patient data. Both mean absolute error ratios (*solid*) and upper quartile error ratios (*dashed*) are shown.

4.3 Supervised Learning of Adaptive DBS

Having demonstrated success with the autoencoded PSD signals, we wished to see if we could use the information proactively, rather than simply diagnostically. As an initial step, we turned to computational modeling of the basal ganglia to attempt to design an adaptive controller to modulate activity in the local field potential's power spectral density. A modified, adaptive version of the prediction model was implemented for the computational model of the basal ganglia from [76, 75]. A greedy algorithm was used to select an input driving stimulus frequency ν that would drive the system such that the maximum power in the beta band was below a target threshold T with a specified probability q during the next window of consideration; this corresponded to finding the lowest frequency for which the predicted probability of a burst not occurring exceeds the target probability. We note that this probability is equivalent to the complement of the average for the 1.0-second window in the previous section; we have flipped the convention here to improve the clarity in the visible trends in the subsequent results. More formally, we define our control policy as:

$$\Pi\left(P, \nu_0\right) = \min_{\nu \in [0,200]} \nu \Big| p\left(\max_{f \in [15,35)} P\left(f\right) < T | \nu, \nu_0\right) = q, \tag{4.1}$$

where P is the power spectral density of the current window and ν_0 is the average frequency in the current window. For all the simulations presented in this paper, a target threshold of -15.781 dB was used, as this corresponded to the mean magnitude across 10 trials for the PSD when stimulated with a constant frequency of 135 Hz. 135 Hz was used as the benchmark control frequency because it is within conventional frequency stimulation ranges for deep brain stimulation.

The stimulation frequency was capped at 200 Hz, and the stimulation frequency was refreshed every 2 seconds. The adaptive model was trained on a batch of signal-PSD correlations every 0.5 seconds with a learning rate of $\alpha = 0.0001$. Training was accomplished using a weighted random sampling of previous windows, with a bias toward recent samples. The probability weight, ρ_k, for the PSD at index k, P_k, was calculated as:

$$\rho_k = \frac{\mu\left(P_k\right)}{\sum_{i=1}^{N} \mu\left(P_i\right)}, \tag{4.2}$$

where:

$$\mu\left(P_k\right) = \exp \frac{k - N}{20L} \tag{4.3}$$

and L is a user-defined constant that may be adjusted as desired (for all trials here, it was set to the length of one window of data, or 2048). Because of the stochasticity of the system, multiple trials were run at each condition for comparison. Trials using conventional DBS were also run for comparison at a range of frequencies.

Examples of two such trials are shown in Fig. 4.18. We see that, although the stimulation frequencies vary differently between the two trials, they both maintain a tight bound around the target probability of $q = 0.5$, indicating these fluctuations are responsive to the stochasticity of the system rather than error in the training. Unless otherwise noted, all simulations were carried out for 1,000 seconds of simulation.

The control algorithm demonstrated the ability to target probabilities across a broad spectrum. With the same target PSD threshold of -15.781 dB, the algorithm was able to effectively and consistently learn appropriate controls to maintain probabilities below the threshold ranging from $q = 0.25$ to $q = 0.75$ (see Fig. 4.19). In all trials, accuracy in the first half of the simulation was outperformed by accuracy in the second half of simulation, demonstrating the algorithm's ability to effectively learn and improve its predictions.

We make two observations about these results. Perhaps most importantly, the control strategy is successful in maintaining consistent results despite the presence of noise in

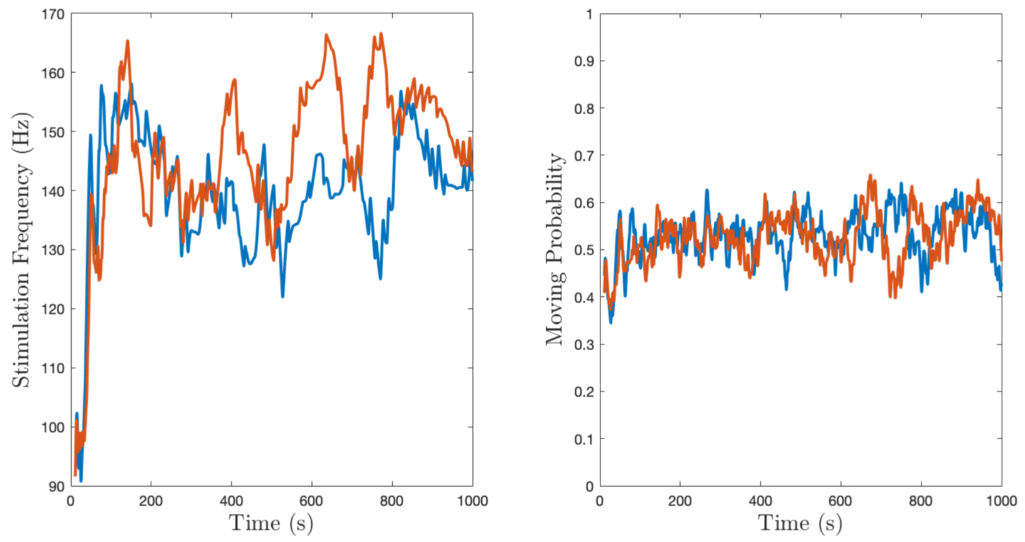


Figure 4.18: Examples of stimulation frequencies (*left*) and probabilities (*right*) for two trials (one in red, one in blue) with target probability $q = 0.5$. The adaptations in the frequency effectively keep the moving-average probability near 0.5

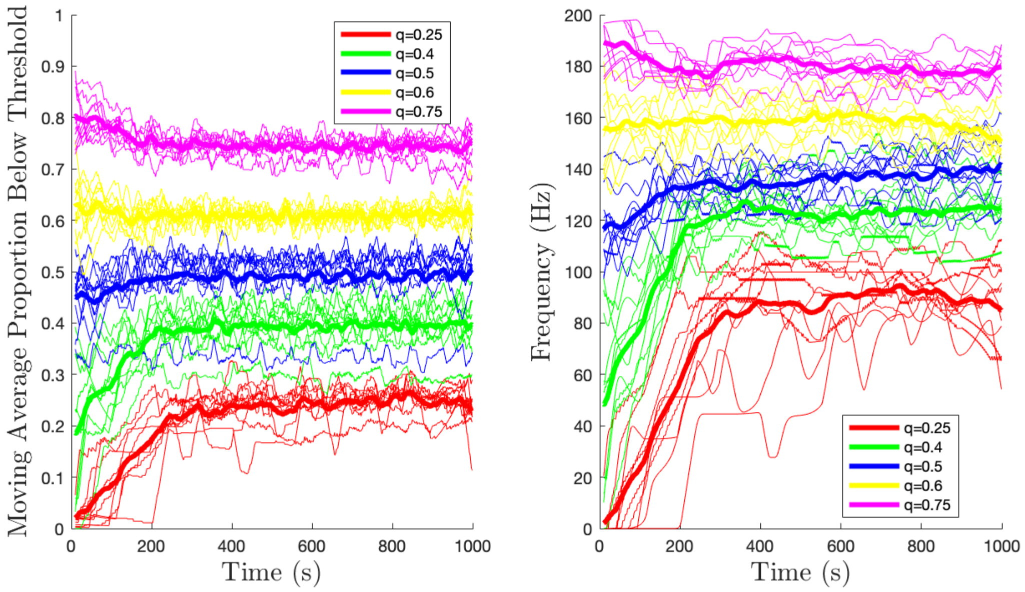


Figure 4.19: Results of trials for different probabilities with the same threshold. All trials attempted to control the maximum magnitude of the PSD in the beta band to stay below the threshold (-15.781 dB) for a proportion of the time defined by q. The mean across trials for the running average proportion is bolded, while individual trials are shown as unweighted lines.

the model. While the presence of noise ensures that it is impossible to maintain perfect tracking of the target probability of all times, the results, when coupled with the trials of conventional DBS, demonstrate that informing stimulation frequency selection using prior signal information can effectively allow us to operate closer to this theoretical limit.

Second, we observe that the model is successful in rapidly learning to accurately control the LFP signal. Referring again to Fig. 4.19, we can see that the rise time for the cumulative probability is less than 200 seconds across all trials.

4.4 Adaptation to Slow Parameter Variations

In practice, DBS is typically combined with pharmacological treatment of Parkinson's disease, typically with L-Dopa. Traditional DBS requires tuning of both off-medication and on-medication parameters with the intent of minimizing symptoms while off-medication and reducing negative side effects while on-medication, such as dyskinesia. Progression of the PD condition in patients may also require adjustment of the parameters associated with effective symptom control. One shortcoming of traditional adaptive DBS strategies is that they lack a mechanism for addressing these changes in the underlying physiology in adaptively updating stimulation parameters; rather, parameters are designed based on a specific state or model which may lack the ability to extend to a slowly-changing system.

We explored the ability of our control strategy to learn these slow adaptations by modifying our model to introduce a time variation in the parameters that differed between the dopamine-depleted and healthy states. The value of a given parameter X at time t was treated as varying sinusoidally, given by:

$$X\left(t\right) = X_H + 0.5\left(1 + \cos\frac{2\pi t}{\tau}\right)\left(X_{DD} - X_H\right), \tag{4.4}$$

where τ is the period of the slowly-varying parameter changes and X_H and X_{DD} represent the "healthy" and "dopamine-depleted" parameter states, respectively. In this formulation, a properly-functioning control algorithm that is effectively learning the slow variation of parameters should reduce stimulation when $X \approx X_H$ and operate at or near the stimulation frequencies shown previously when $X \approx X_{DD}$.

The algorithm was able to learn the desired behavior for a range of variation frequencies. Simulations were extended to 10,000 seconds to allow for observation of the effects of parameter variation over the course of, at a minimum, two cycles; 9 trials were conducted with periods incremented from τ =1,000 seconds up to τ =5,000; the moving average of the probability was then calculated. In each trial, the target probability was set to $q = 0.6$.

We summarize the results of these trials in Fig. 4.20. We found our network was able to consistently track and adapt to the changing system without a significant loss in tracking performance. The average stimulation frequency showed a reliable variation that was consistent with that of the parameter variation. This is evidenced in Fig. 4.21, which compares the oscillation in stimulation frequency with that of the parameter variation. We note that, in general, stimulation frequency closely matches the underlying parameter variation in both stimulation frequency and phase. We additionally note that for several samples the peak stimulation frequency actually led the parameter variation rather than lagging it; the cause of this was unclear in our trials.

We emphasize here that the adaptive model was given no information about the variation in parameters of the underlying basal ganglia model. Instead, changes in the relationship between stimulation and LFP measurements allowed the network to learn the variation and develop characteristic fluctuations in stimulation frequency that mimicked the variation rate of the underlying parameter variation. We believe this agnosticism to model information is a particularly attractive feature of our proposed adaptive model

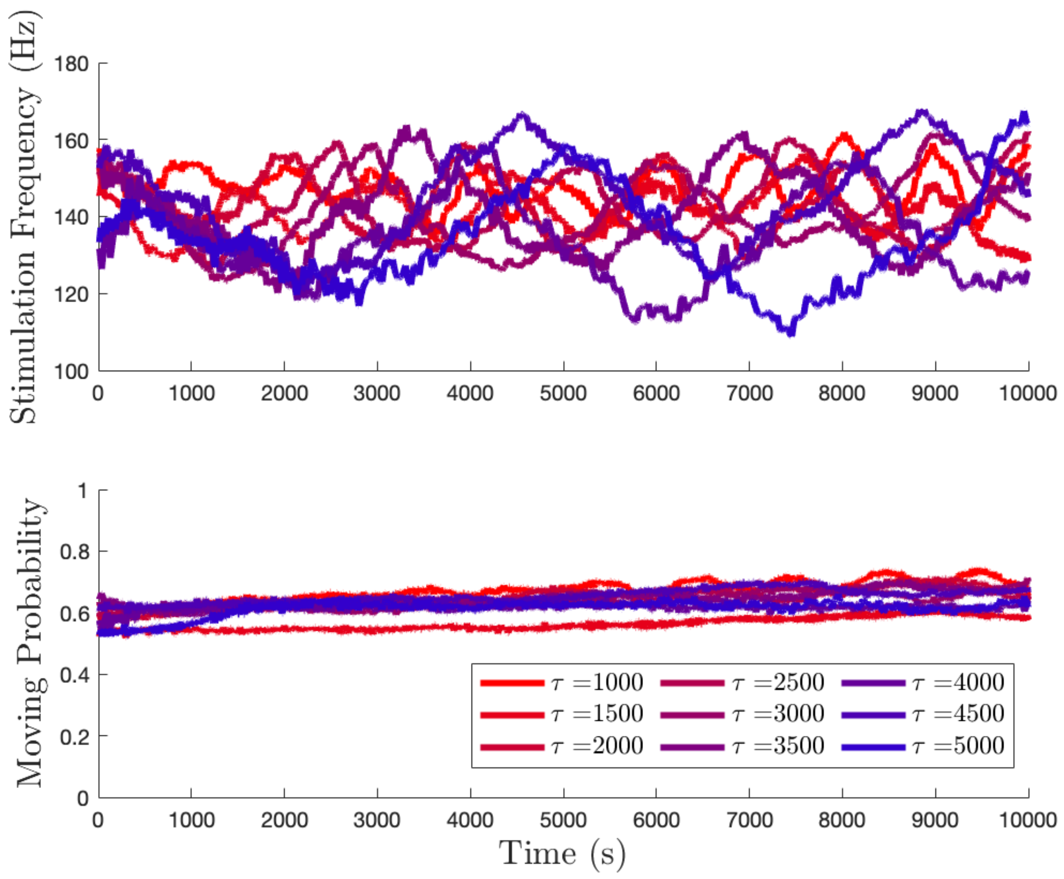


Figure 4.20: Smoothed moving averages for frequency (*top*) and probability (*bottom*) for each trial. As can be seen, tracking error is relatively low, and clear oscillations with the same period as the parameter variation can be seen in the stimulation frequency, demonstrating the network's ability to effectively adapt to the changing parameters.

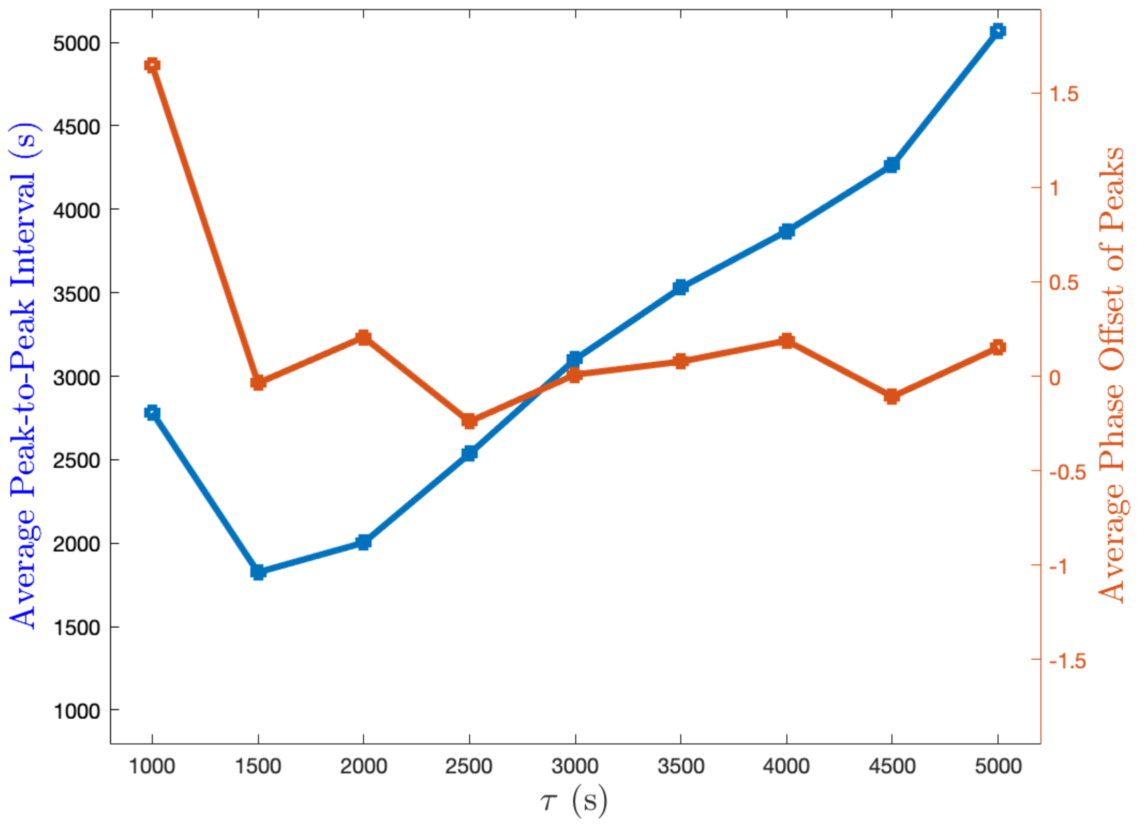


Figure 4.21: Average peak-to-peak interval (blue) and phase offset (red). The phase offset was measured as the time difference between the peak of the parameter variation and that of the stimulation frequency normalized to radians relative to the parameter variation's period.

when compared to alternative forms of adaptive DBS. As patient physiology changes, either in the short term because of medication or longer term because of progression of the disease, appropriate stimulation parameters and targets must be adjusted as a result in most methods; here, the adaptive model is able to make these adjustments on-line without additional input. With this in mind, we have presented an artificial neural network that is trained to predict the response of the the basal ganglia to varying stimulation, allowing for a more robust basis for control design without sacrificing the capacity to adapt to changes in the underlying model parameters.

4.5 Conclusion

We have developed a robust, flexible neural network-based algorithm that is capable of accurately achieving a range of control objectives while simultaneously learning and refining itself. From a common starting point of a trained autoencoder for feature extraction, we have demonstrated robust accuracy in both prediction tasks on patient data and control tasks on simulation data and demonstrated the general utility of an autoencoder in addressing prediction problems in neuroscience. In its current formulation, the control algorithm already has significant flexibility, allowing for the arbitrary choice of both threshold values and probabilities and maintaining fidelity across these choices. Furthermore, the neural network successfully adapts to variations in the underlying biological model provided it is trained with an appropriate learning rate. Beyond this, however, the core architecture of this model allows for further exploration of different control objectives, including non-greedy control strategies with discounted future rewards or subjecting the control to additional constraints. There are also a number of exciting directions for modifying the existing architecture to increase the functionality of the algorithm, including the introduction of new state parameters, such as stimulation amplitude, and adaptation to other frequency ranges, e.g. the gamma range, to attempt to ameliorate symptoms of Parkinson's disease such as dyskinesia. Additionally, we believe the use of a common autoencoder for both components of our study suggests the core architecture of our control algorithm may be carried over into in-patient adaptive control.

Chapter 5

Continuing Projects: Filling in the Blanks with Machine Learning

Throughout this dissertation, we have seen examples of how machine learning can allow us to develop robust, accurate, and precise predictions when information may be at a premium. These predictions, in turn, allow us to develop and implement control protocols. Whereas the work in Chapters 3 and 4 focused on how to generate control strategies when the right-hand side of the governing ODEs is unknown or we cannot measure the appropriate states of the system, here we shift our attention to using machine learning techniques to learn precisely those pieces of information we assumed unknowable. Specifically, we will examine two ongoing projects. The first of these utilizes artificial neural networks to learn an analytical expression for the right-hand side of the governing ODEs, while the second attempts to extract global phase and isostable response curves from a limited dataset.

5.1 Symbolic Regression via Artificial Neural Networks

We have begun exploring a novel strategy for identifying the analytic expressions for systems of equations based on an analysis of their time-series data. The goal is to develop a strategy that is as general as possible, including minimal presuppositions about any characteristics of the model. This is in contrast to previous methods, such as those proposed in [79], where system identification proceeds from a dictionary of constituent terms. We accomplish this by using a multilayered, operation-based approach, with the capacity to learn compound operations through training. For ease of training, all operations are designed to be continuous and differentiable. This can be viewed as a form of symbolic regression, but with a structure that is amenable to the use of artificial neural networks.

Our network can learn four different types of operations, each of which will be subsequently explained: 1) Linear combinations; 2) Polynomial combinations; 3) Simple products of variables; and 4) Common operators. The network can possess a two-tier hierarchical architecture with $K \geq 1$ stacked layers, each of which carries out each of these operations $L \geq 1$ times. The network takes initially as input the n system states and an additional constant value of 2 to aid with scaling, for $n+1$ total states. If desired, time could be included as well, allowing for $n+2$ initial states. If $K > 1$, then the kth layer takes as input not only the initial $n+1$ states but also the $3L\,(k-1)$ states generated by operations in the previous layers, which allows for highly complex analytic expressions with minimal training parameters. Each of these processes and operations is outlined below.

5.1.1 Operational Layers

Throughout this section, the totality of states, including any observables generated within the network, will be referred to simply as x, with x_m denoting the m^{th} value in the vector x. Weight vectors for trainable layers will be referred to as W, with W_m denoting the corresponding weight.

Linear combinations

As suggested by the name, these are just linear combinations of the pre-existing states:

$$y = \sum W_m x_m. \tag{5.1}$$

This is helpful in deeper stacked implementations, where sums of individual components of x may undergo significant nonlinearities.

Polynomial combinations

The output of a polynomial combination layer is given by:

$$y = \prod |x_m|^{W_m}. \tag{5.2}$$

The absolute value of each state is used to ensure that each output value y remains in the domain of real numbers; sign considerations are handled separately. To enforce parsimony (also known as Occam's Razor - choosing the simplest explanation for a phenomenon), regularization is applied to the model; for all non-constant values of x, the L_1 norm is considered, and this value is multiplied by the product of $1.1^{W_{last}}$, where W_{last} is the weight corresponding to the constant value of 2. This is designed to drive the value of this state toward 0 rather than 1, which in turn effectively weighs the entire term to equal 0, thereby removing it from the final expression.

For example, suppose we wish to represent the equation $\dot{x_1} = {x_2}^2$; if our system is $[x_1, x_2, 2]$, then the corresponding learned weight matrix should yield $W = [0.0, 2.0, 0.0]$ after training.

Simple products

In many instances, the absolute value of a state is insufficient; the simple product operation allows us to combine states raised only to the first power. Similar to the polynomial combinations, these operations are dictated by a product rule:

$$y = \prod v_m, \tag{5.3}$$

where v_m is given by:

$$v_m = \sigma\left(W_m\right) x_m + \left(1 - \sigma\left(W_m\right)\right). \tag{5.4}$$

$\sigma\left(W_m\right)$ represents the sigmoidal activation function so that for $W_m \ll 0$, $v_m = 1$ and for $W_m \gg 0$, $v_m = x_m$. No regularization is applied to this layer since responses are already bounded.

For example, suppose $\dot{x_1} = x_1 x_2$. Then the resulting weight matrix after training may be similar to $W = [1.5, 2.1, -0.99]$; $\sigma\left(1.5\right) \approx \sigma\left(2.1\right) \approx 1$ while $\sigma\left(-0.99\right) \approx 0$, so plugging into (5.4) for each term and multiplying according to (5.3) yields:

$$\left(1x_1 + \left(1-1\right)\right)\left(1x_2 + \left(1-1\right)\right)\left(0\left(2\right) + \left(1-0\right)\right) = x_1 x_2.$$

Common operators

This set of operational layers is reliant on a set of common symbolic operations - such as exponentiation, sine, sign, and the sigmoid operator - which are selected between. The layer consists of two consecutive implementations of dense layers; the first dense layer returns a linear combination of the input states:

$$y_a = \sum x_m W_m, \tag{5.5}$$

while the second is a linear combination of the different operations g applied to the output of the first:

$$y = \sum g_m \left(y_a\right) w_m. \tag{5.6}$$

This layer uses a square-root norm regularization, heavily penalizing initial deviations from 0 with a decreasing subsequent penalty as the distance from 0 increases. This regularization is designed to enforce sparsity while not heavily penalizing correct terms. Note that, although we need to choose the set of common symbolic operations that can be used in this operational layer, by putting this together with the other operational layers we are able to obtain a much richer set of possible terms than one would expect from methods that require full pre-specification of possible terms.

5.1.2 Complex Representations via Stacking Layers

A core strength of this methodology is the capacity for learning complex representations of dynamics without knowing those representations *a priori*. For example, suppose we wish to model an exponentially decaying derivative, such as:

$$\dot{x} = \exp\left(-\frac{\left(x - x_0\right)^2}{\tau}\right) - ax, \tag{5.7}$$

which is similar in form to conductance-based modeling in computational neuroscience. The exponential behavior may be dificult to identify without prior knowledge of the system, especially if there are a number of other terms and the exponential amounts only to a transient behavior. With just two stacked layers, however, the network we have proposed can learn this behavior. Initially, each input state is simply $[x, 2]$. The first of the stacked layers generates $3L$ new observables, among which is 2^0x^2, which was generated by a polynomial combination operating layer. The second layer can then generate the right hand side of (5.7) by passing the observables through a common operators operational layer, selecting for the exponential operator and appropriate weights on x^2, x, and 2. Finally, all the generated states and observables are passed through a dense layer, where weights of 1 and $-a$ are selected for the exponential and x terms, respectively; all other output observables/states will possess weights nearly identical to 0.

5.1.3 Application: Hodgkin-Huxley Equations

The reduced Hodgkin-Huxley equations are provided in Appendix A.1; the full equations are of a similar format with two additional gating variables. Traditional system identification procedures would be unlikely to successfully generate such a complex set of equations, but the methods proposed here are, in principle, able to reconstruct these equations. The dynamics of V are comparatively straightforward to reconstruct; we can do so by grouping polynomials and rewriting, finding the general structure of $\frac{dV}{dt}$:

$$\frac{dV}{dt} = C_1 + C_2n^4 + C_3n^4V + C_5m^3h + C_6m^3hV + C_7V. \tag{5.8}$$

The first and final terms in this equation can be generated solely from initial inputs (the constant value and voltages, respectively). All other terms represent polynomial combinations, so the dynamics of the voltage can be fully described by the model. Because V can be negative or positive and the polynomial generator only allows for absolute values, the monomials n^4 and m^3 can be generated in the first stack, allowing the simple product operators to generate the terms in a second stack.

Each of the exponentials in $\alpha(V)$ and $\beta(V)$ can be generated by observing that the arguments are just a linear combination of a constant value and V; if we have 6 common operator layers in our first layer, we can generate all the exponents required. To generate the denominators in the $\alpha(V)$ terms requires a second stacked set of operations, requiring 3 linear combinations to get the terms in the denominators. We may also generate the terms in the numerators using additional linear combinations, in either the first or second stack.

Therefore, we have introduced sufficient architecture with two stacks to model:

$$1.\ \exp\left(\frac{-V}{\tau_{x,2}}\right); \quad 2.\ \exp\left(\frac{V-\theta_x}{\tau_x}\right)-\sigma_x; \quad 3.\ V-\theta_x; \text{and} \quad 4.\ \text{every term in } \frac{dV}{dt}.$$

A third stack allows for us to fully synthesize the components for implementing the α functions: we arrive at the expression using a polynomial combination of $V-\theta_x$ and $\exp\left(\frac{V-\theta_x}{\tau_x}\right)-\sigma_x$ with exponents of 1 and -1 respectively, and we simultaneously implement the sign operator on $\exp\left(\frac{V-\theta_x}{\tau_x}\right)-\sigma_x$ and $V-\theta_x$ to account for the loss of sign when taking the absolute value. Finally, in the fourth stack of operations, we can use simple product rules to generate all the desired terms for $\frac{dn}{dt}$, $\frac{dm}{dt}$, and $\frac{dn}{dt}$.

A summary of this process is presented in Fig. 5.1. Despite the apparent complexity of this process, we note that our formulation accomplishes something no other methods can similarly accomplish: the capacity to reconstruct highly complicated dynamics with *no prior insight into the system's dynamics.* The form of the Hodgkin-Huxley equations would almost certainly elude approaches with a pre-specified dictionary because of its specificity and uncommon dynamics; unless the user knew ahead of time that these dynamics were the correct dynamics, they would not be included in a dictionary-based approach.

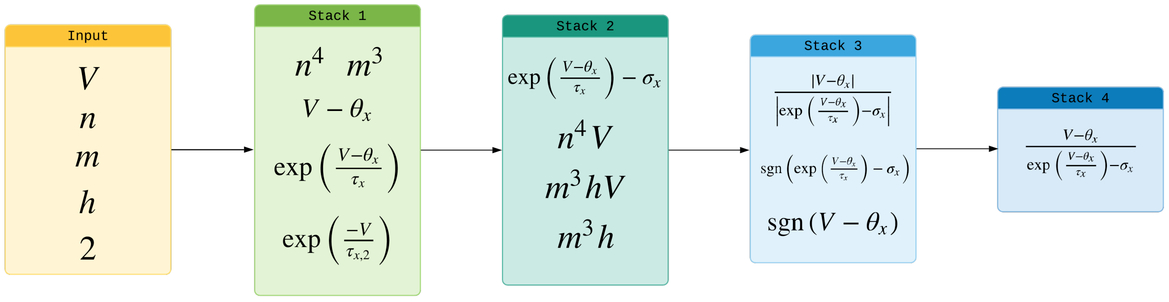

Input
V
n
m
h
2
Stack 1
n^4 m^3
$V-\theta_x$
$\exp\left(\frac{V-\theta_x}{\tau_x}\right)$
$\exp\left(\frac{-V}{\tau_{x,2}}\right)$
Stack 2
$\exp\left(\frac{V-\theta_x}{\tau_x}\right)-\sigma_x$
n^4V
m^3hV
m^3h
Stack 3
$\frac{|V-\theta_x|}{\left|\exp\left(\frac{V-\theta_x}{\tau_x}\right)-\sigma_x\right|}$
$\mathrm{sgn}\left(\exp\left(\frac{V-\theta_x}{\tau_x}\right)-\sigma_x\right)$
$\mathrm{sgn}(V-\theta_x)$
Stack 4
$\frac{V-\theta_x}{\exp\left(\frac{V-\theta_x}{\tau_x}\right)-\sigma_x}$

Figure 5.1: Abstraction of model generation for the Hodgkin-Huxley equations.

5.1.4 Network Training and Model Selection

The ANN that generates the model is trained as a regression problem on the derivatives of the states, with the error calculated as the mean absolute error between the true time derivatives at each state/time and the network's estimates using the current parameters. While the model was known in the preliminary work we have done (and therefore exact derivatives could be calculated when generating the dataset), this could in principle be done using methods such as numerical approximation or automatic differentiation [80].

Because all of the component layers in our network are continuous and differentiable almost everywhere, we can employ standard machine learning techniques to train the network. This includes any traditional optimizers or methods for feed-forward neural networks, such as Adam, Nesterov momentum, RMSprop, and stochastic gradient descent.

We enforce parsimony in the resulting estimate of the dynamical system by utilizing the specialized regularizers described previously. The goal of these regularizers is to drive as many of the output variables in the network as possible to 0 or, barring that, a constant value. In reality, however, the weights of the network will not reach values yielding identically constant or identically 0 values. To compensate for this, we then "prune" the output, identifying terms with minimal contribution to the overall dynamical system and manually setting their weighting in the output to 0. For example, a polynomial combination layer might yield a term resembling:

$$y = {x_1}^{4e-4} {x_2}^{2.01} {x_3}^{-0.01} 2^{0.003}. \tag{5.9}$$

All of the terms have negligible impact on the overall value when compared to x_2, and are nearly equal to 1 for most states. Therefore, in the pruning process this equation would instead be recorded as $y = {x_2}^{2.01}$, which could be approximated as $y = x_2^2$ if desired.

One area of further research is in automation of this pruning process. Observation of the impact of each quantity on the output is one possible strategy for pruning; if, for example, the magnitude of the output of a set of operations is small for all tested data points (below some predetermined threshold), it may be a good candidate for pruning. Similarly, terms that have a disproportionately large effect on outliers but little effect on accurate data points may be suitable for elimination as well.

Parsimony is also generated by specifying the network structure. By enforcing a particular maximum number of stacks and operations in each stack, we limit the complexity of resulting expressions. This in turn limits the number of trainable parameters, reducing the risk of overfitting.

5.1.5 Preliminary Results

As a proof of concept, we have implemented our method with two different canonical dynamical systems that were selected to test the ability for our method to respond to common dynamical models. The two models tested so far are a realization of a Bogdanov-Takens system with parameters selected to generate a stable fixed point, and a system of three Kuramoto oscillators. In both cases, training was done based on the results of a single simulated trajectory. Different architectures, both single-stack and double-stack, were attempted for both systems. We note here that overfitting is less of a concern in our method than in most deep learning applications as the number of trainable parameters in our network remains quite small (on the order of 100 for a single-stack network). Still, the trajectory simulation data was separated into five separate folds, and K-fold validation was used to select the best model for the dynamical system.

Bogdanov-Takens System

The Bogdanov-Takens normal form is given by [81]:

$$\dot{y}_1 = y_2, \tag{5.10}$$

$$\dot{y}_2 = \beta_1 + \beta_2 y_1 + {y_1}^2 + y_1 y_2. \tag{5.11}$$

For our test simulation, we selected $\beta_1 = \beta_2 = -1$, which yields a system of equations with a stable fixed point at $\vec{y} = \left[\frac{1-\sqrt{5}}{2}, 0\right]^T$ and a saddle at $\vec{y} = \left[\frac{1+\sqrt{5}}{2}, 0\right]^T$. The network was initially trained as a single stack with 10 copies of each operational layer. The original quiver plot for the simulated trajectory as well as the reconstructed quiver plot based on the network's predictions are shown in Fig. 5.2. As can be seen, the network was able to faithfully learn the dynamics of this trajectory.

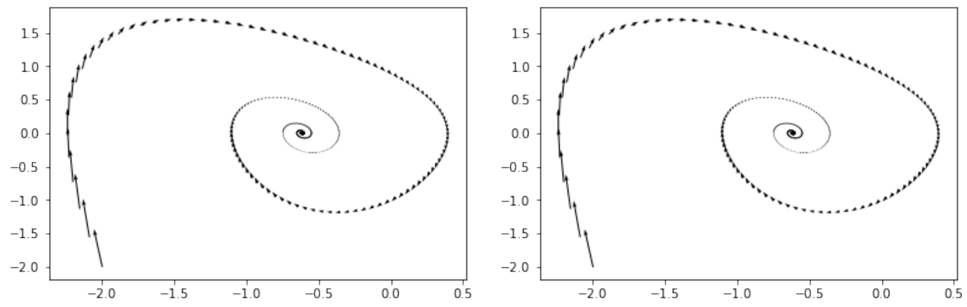


Figure 5.2: Quiver plots of the Bogdanov-Takens system's true dynamics (*left*) and the artificial neural network's estimates of these dynamics (*right*). Here, the x-axis corresponds to y_1 while the y-axis corresponds to y_2.

After training, the weights were output and manually pruned, discarding terms with small effects on the overall output of the system. Because of the regularization strategies used, this was a relatively straightforward task, as the majority of output values could be assumed to be almost identically 0 or 1. By discarding the bulk of the terms and keeping only those with significant weights in the output equations, a simplified version of the method's estimate was acquired:

$$\hat{\dot{y_1}} = 0.9997 y_2, \tag{5.10a}$$

$$\hat{\dot{y_2}} = -1.012 - 1.012 y_1 + 0.99528 {y_1}^{2.00} + 1.02 y_1 y_2. \tag{5.11a}$$

From this we see that not only was our network able to successfully reconstruct the dynamics of the trajectory it was trained on, it was able to accurately reconstruct the global dynamics of the system. The estimated dynamical system implies the existence of the broader dynamics of the Bogdanov-Takens normal form, including the existence of a saddle point and possible parameters that may be varied to adjust the behavior of the system.

Kuramoto Oscillators

The Kuramoto model is a common simplification of coupled oscillatory systems, such as those found in neuroscience applications [4, 82]:

$$\dot{\theta}_j = \omega_j + \frac{K}{N} \sum_{j \neq k}^{N} \sin\left(\theta_k - \theta_j\right), \qquad j = 1, \cdots, N, \tag{5.12}$$

where θ_j is the phase of the j^{th} oscillator, N is the total number of oscillators, and K represents a coupling strength between the oscillators. We simulated a system of three such identical ($\omega_1 = \omega_2 = \omega_3$) oscillators with $K = 0.5$ and $\omega = 1$. For these parameters, the three oscillators eventually converge to an in-phase state, continuing to oscillate such that their phases θ approach each other ($\theta_1 = \theta_2 = \theta_3$).

The dynamics were modeled first considering the modulo 2π values of θ. The results of these simulations are shown in Fig. 5.3; as can be seen, the dynamics are largely recovered, though some artifacts exist because of the discontinuity at 2π. The size of these artifacts is relatively minimal compared to the broader dynamics, and could be filtered out by manually pruning the output.

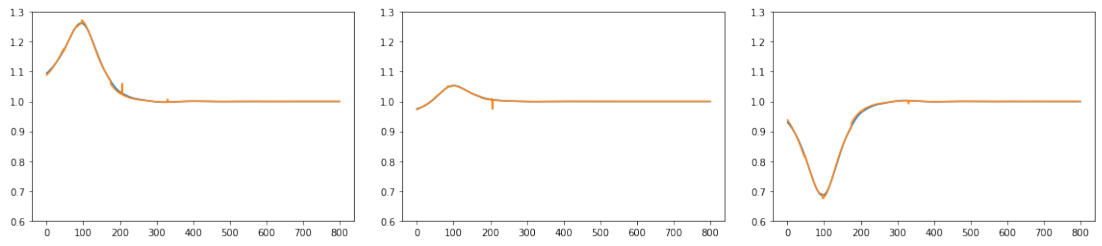


Figure 5.3: Derivatives as functions of time (*left* to *right*, $\dot{\theta}_1$, $\dot{\theta}_2$, and $\dot{\theta}_3$) for the three simulated oscillators. Time (x-axis) is in arbitrary units. The blue curves represent the true derivatives, while the orange curves are the network's predictions.

5.1.6 Generating Lyapunov Functions

The generation of novel Lyapunov functions is an area of great interest in the dynamics and control communities. Typically, Lyapunov functions are generated in an *ad hoc* fashion for dynamical systems, requiring some combination of insight and luck on the part of the researcher. A first guess is often to take a linear combination of the squares of all state variables, but this is not guaranteed to work. For example, consider the Lotka-Volterra system [83, 84]:

$$\dot{v} = v(a - ev - bp), \tag{5.13}$$

$$\dot{p} = p(-c + dv), \tag{5.14}$$

with parameters a, b, c, d, e all positive and real, and assume that $ad - ce > 0$. The function $V_1(v,p) = \frac{1}{2}(p^2 + v^2)$ is not a Lyapunov function, but the following function is:

$$V(v,p) = d(v - \tilde{v}\ln v) + b(p - \tilde{p}\ln p) - d(\tilde{v} - \tilde{v}\ln\tilde{v}) - b(\tilde{p} - \tilde{p}\ln\tilde{p}), \tag{5.15}$$

where $(c/d, (ad - ec)/(bd)) \equiv (\tilde{v}, \tilde{p})$ is a fixed point.

Machine learning presents an attractive strategy for automating the process of finding Lyapunov functions. A general strategy, for example, may consist of finding a mapping of a set of variables such that the output is positive definite for $\vec{x} \neq 0$ and its time derivative is negative semidefinite; computation of this derivative is straightforward for a neural network provided the dynamics of the system are known, and such a network is readily trainable.

The chief hurdles in using neural networks to learn Lyapunov functions are:

1. Neural networks are typically excellent at interpolation and poor at extrapolation; and

2. Neural networks are black boxes that evaluate a finite number of points and are therefore difficult to generalize conclusively to a continuous region in $\mathbb{R}^n$ about the origin.

Put simply, a neural network generally can only tell us conclusively that a finite number of test points satisfy the conditions for a Lyapunov function, and they provide little in the way of mechanisms to generalize these discrete results to a continuous space.

We believe our method could automate the process of generating candidate Lyapunov functions for a given dynamical system. By providing an analytic expression satisfying the constraints of a Lyapunov function at a set of finite points, one can then verify or reject the candidate function by evaluating this function with the techniques one would traditionally use for testing analytic functions when developing them *ad hoc*. Randomized initialization of the network, as well as using different architectures, may provide multiple alternative candidate Lyapunov functions.

There are two primary components to our strategy for developing candidate functions. The first is intelligent selection of the operational layers in the method; if the method consists of creating a vocabulary, this may be seen as instituting a "grammar" for functions. The second is the selection of the objective function to be minimized.

We require a function that is identically zero at the origin and positive definite away from the origin; we can accomplish this by selecting our operations such that they all individually satisfy this requirement. If this is the case, then any composition or combination of the functions will also satisfy the required positive semidefiniteness.

Suppose $f\left(\vec{x} \neq 0\right) > 0$ and $g\left(\vec{x} \neq 0\right) > 0$, and further suppose $f(0) = g(0) = 0$. Then the following are all true:

$$g \circ f(0) = 0, \qquad f \circ g(0) = 0,$$

$$g \circ f\left(\vec{x} \neq 0\right) > 0, \qquad f \circ g\left(\vec{x} \neq 0\right) > 0,$$

$$\left(f + g\right)(0) = 0, \qquad \left(f + g\right)\left(\vec{x} \neq 0\right) > 0.$$

All we require, then, is that every individual operation in our method, when applied to a vector $\vec{x}$, returns 0 when $\vec{x} = 0$ and a positive value otherwise. This is easily accomplished by editing the core architecture with specific constraints. Considering the different layer types described above, simple examples of valid constraints are presented in Table 5.1.6.

The function we then choose to minimize is the time derivative of the Lyapunov function output. Specifically, we are interested in finding a Lyapunov function V such that $\dot{V} < 0 \;\; \forall \vec{x}$. We only have a valid Lyapunov candidate, then, if at every test point the

Layer Type	**Constraint**
Polynomial Combination	All weights $w_i \geq 0$
Linear Combination	No direct output to Lyapunov function (weight in output layer is identically 0)
Simple Products	Constant value should be subtracted; absolute values used for states.
Common Operators	Appropriate constants/modifications made to satisfy requirements (e.g., instead of $\exp(x)$, $\exp(x^2) - 1$)

Table 5.1: **Example constraints for Lyapunov function generation.**

time derivative is less than 0. The time derivative itself is straightforward to compute, as all of our terms have analytic derivatives and can therefore be found with successive applications of the chain rule. If a Lyapunov function exists subject to the imposed grammar, then our objective function L to minimize is:

$$L = \sup_{x \in \mathbb{D}} \dot{V}, \tag{5.16}$$

where $\mathbb{D}$ is the set of all points used for training and the minimized loss should be less than 0.

5.2 Estimation of Global Phase and Isostable Response Curves

As demonstrated in Chapter 2, knowing the global phase and isostable response curves for a dynamical system is immensely powerful, opening up the ability to achieve a wide array of control objectives, such as optimizing firing time or controlling to a point away from the limit cycle, which require significantly more computation if considered in the original state space. However, as also noted in Chapter 2, it is not necessarily the case that a simple, analytic representation of the global coordinates exists, and our ability to

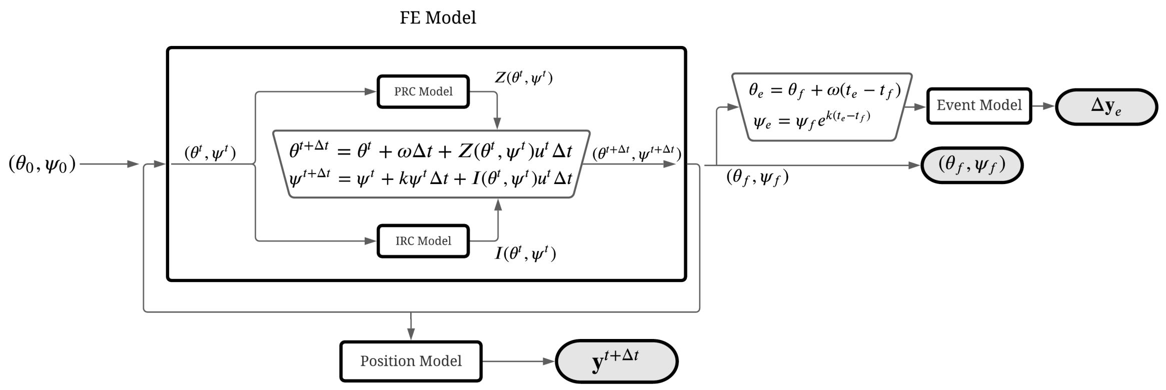

FE Model
PRC Model
$Z(\theta^t,\psi^t)$
$\theta^{t+\Delta t} = \theta^t + \omega\Delta t + Z(\theta^t,\psi^t)u^t\Delta t$
$\psi^{t+\Delta t} = \psi^t + k\psi^t\Delta t + I(\theta^t,\psi^t)u^t\Delta t$
IRC Model
$I(\theta^t,\psi^t)$
(θ_0,ψ_0)
(θ^t,ψ^t)
$(\theta^{t+\Delta t},\psi^{t+\Delta t})$
$\theta_e = \theta_f + \omega(t_e - t_f)$
$\psi_e = \psi_f e^{k(t_e - t_f)}$
Event Model
$\Delta \mathbf{y}_e$
(θ_f,ψ_f)
Position Model
$\mathbf{y}^{t+\Delta t}$

Figure 5.4: Neural network architecture for global phase reduction modeling (figure from Yates et al. 2021, *in preparation*).

identify where in phase space we are can be limited by our inability to fully observe the dynamical system of interest. In collaboration with our undergraduate researcher Andrew Yates, we have worked to develop a neural network architecture that allows us to extract global isostable and isochron response curves from datasets composed of repeated trials with constant stimulation of varying amplitudes, durations, and start times. In addition, the model does not require every state of the system to be fully observable.

This represents a potentially significant step forward in our ability to globalize phase model reduction techniques, which frequently requires explicit knowledge of the right-hand side of the system of ODEs or can only be solved numerically for specific trajectories, cf. [46]. Our proposed model, developed by Yates, subdivides the problem into a series of inerconnected neural networks, each tasked with generating an estimate of a particular component of the overall solution, and makes use of an event-based estimation strategy to overcome the lack of full observability in models such as conductance-based neuron models. This neural network model is shown in Fig. 5.4. The model was first verified on systems with known global response curves, including the supercritical Hopf bifurcation analyzed in Chapter 2; the results for that analysis are shown in Fig. 5.5. With the

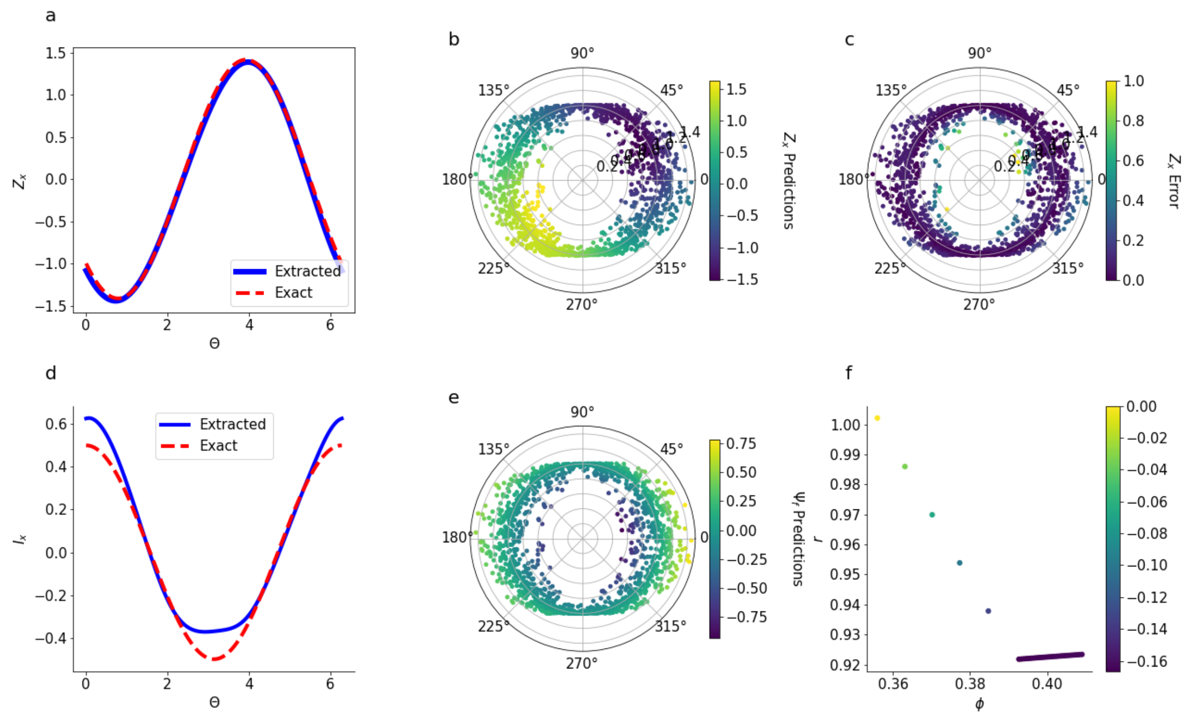


Figure 5.5: Estimated PRC/IRC on the limit cycle and globally for the supercritical Hopf bifurcation, as well as prediction error. Panels (a) and (d) show the learned limit cycle PRC and IRC, respectively, compared to the exact values. Panel (b) shows an estimate of the phase response curve at different points in the state-space, while panel (c) shows the associated error in these predictions. Panel (e) shows predictions of the isostable coordinate in state-space, and panel (f) shows an example trajectory, with the color corresponding to the isostable coordinate (figure from Yates et al. 2021, *in preparation*).

validity of the modeling verified, we applied the model to a system without a previously known global PRC/IRC, specifically, the STN model utilized in Chapter 3 (equations in Appendix A.3). The corresponding results for this are shown in Fig. 5.6; using these estimates, we have been able to generate optimal control signals for modulating firing time that substantially outperform alternative methods in terms of accuracy, and we believe this research opens the door to significant development in the area of phase model reduction moving forward by untethering it from the limit cycle in a robust, consistent, and model-free manner.

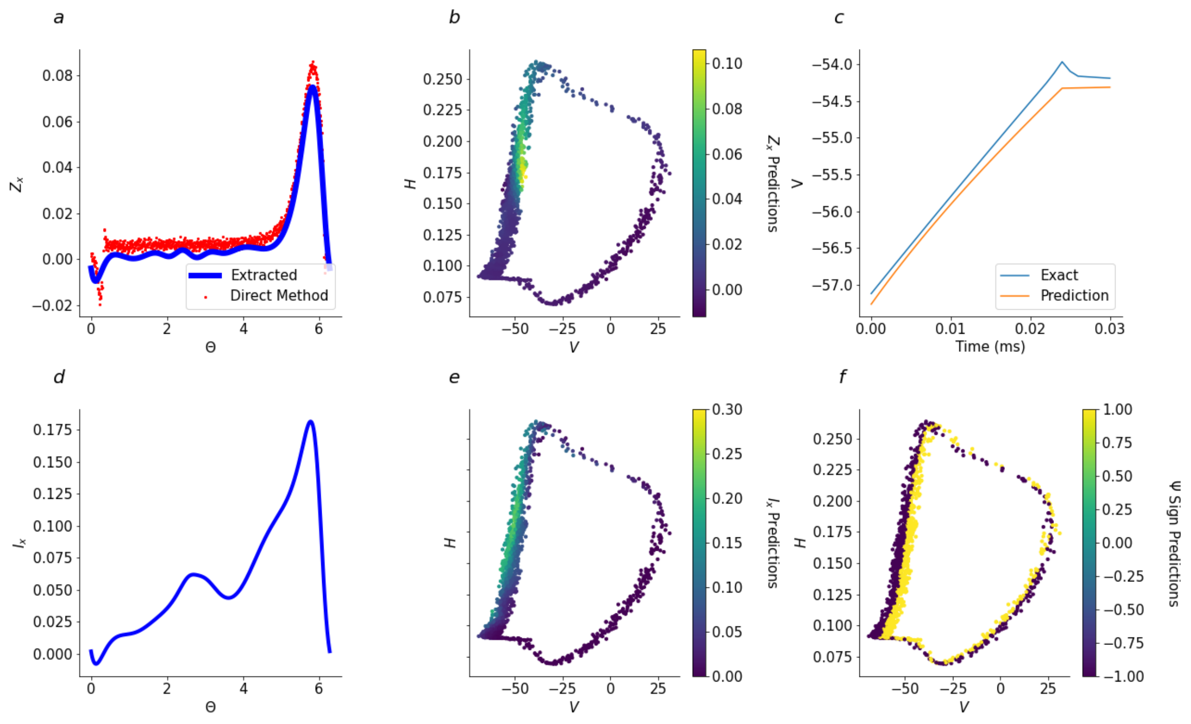


Figure 5.6: Limit cycle PRC/IRC and global coordinates, as well as error, for neural network model applied to the STN neuron model. Panels (a) and (d) show the learned limit cycle PRC and IRC, respectively. Panel (b) shows an estimate of the phase response curve at different points in the state-space, while panel (e) shows the global estimate of the isostable response curve. Panel (c) shows a predicted trajectory, demonstrating the general ability of the network to accurately reconstruct the voltage as a function of time. Lastly, panel (f) shows the sign of ψ at different points in the state space, revealing an error near the spiking of the neuron where the convention for the isostable coordinate flips direction. This is most likely because of the significant lack of data in this region (and proportionately little time spent in the vicinity) when compared to left side of the state-space, below the firing threshold (figure from Yates et al. 2021, *in preparation*).

Chapter 6

Concluding Remarks

In Chapter 2, we saw how we can develop straightforward, intuitive solutions to problems if we can successfully simplify the dynamics to a more approachable system. The challenges of doing so inspired the work of Chapter 3, which considered what we can do if that simplification isn't possible and showed that machine learning can be an effective tool in providing critical information for control design. Chapter 4 demonstrated not only the strength but the flexibility of machine learning techniques, allowing us to generalize from a single Parkinsonian patient's LFP readings to make meaningful predictions on a larger cohort and design adaptive control strategies for combating PD. And in Chapter 5, we looked forward, briefly describing some ongoing projects that seek to further bridge the gap between the worlds of dynamics and control and machine learning.

Throughout this dissertation, it has hopefully become clear that there are distinct strengths and weaknesses to not only the techniques we typically bring to bear on problems in dynamics and control as engineers, but also with the potential solutions provided by machine learning. There is no one-size-fits-all solution for analyzing dynamical systems; sometimes, simpler methods are sufficient, and the addition of techniques such as neural network regression are overkill. Other times, a solution may be difficult to find or even intractable without the use of powerful machine learning techniques, but when processed by a neural network, that solution may lack any context or provide little insight to the researcher.

Whenever we make use of machine learning to tackle a problem, we should not lose sight of the central question of "why." ML may provide us often with easy answers, but an answer is frequently less valuable than a solution. To this end, we have attempted to contextualize the results of our neural networks wherever possible, analyzing not only the predictions they make, but why it sometimes errs or what it is using to make those predictions. Machine learning is most effective when it is a tool in an arsenal of techniques and not treated as a simple panacea for difficult problems. We have attempted to demonstrate this perspective throughout the work presented here, and as our ongoing projects continue to develop, we will strive to maintain and promote this mindset.

Appendix A

ODE Equations

A.1 Hodgkin-Huxley Equations (Reduced)

The Hodgkin-Huxley model is the canonical example of a conductance-based neuron model, developed in the 1950s by Hodgkin and Huxley based on experiments on giant squid axons. It models the response to stimulation of a single neuron and is referred to as a "conductance-based" model because its voltage changes are associated with changes in the permeability of the axon cell membrane to various ions, which in turn affects the magnitude of the associated current. While the original H-H model used 4 variables, the version used here is a reduced model with only two ODEs; the other two variables are assumed to reach steady-state values arbitrarily quickly compared to the voltage and the other gating variable, and are taken as such. The two remaining equations are:

$$\dot{V} = I_0 + I\left(t\right) - \bar{g}_{Na} m_\infty^3 h \left(V - V_{Na}\right) - \bar{g}_K n^4 \left(V - V_K\right) - \bar{g}_\ell \left(V - V_\ell\right); \tag{A.1}$$

$$\dot{n} = \alpha_n - \left(\alpha_n + \beta_n\right) n. \tag{A.2}$$

Variable	**Value**	**Description**
V_{Na}	115 mV	Sodium current reversal potential
V_K	-12 mV	Potassium current reversal potential
V_ℓ	10.5989 mV	Leak current reversal potential
$\bar{g}_{Na}$	120 nS/μm^2	Maximum sodium current conductance
$\bar{g}_K$	36 nS/μm^2	Maximum potassium current conductance
$\bar{g}_\ell$	0.3 nS/μm^2	Maximum leak current conductance
I_0	20 mA	Base current
h_0	0.8	Constant for $h-n$ heuristic

Table A.1: **Table of constants for Hodgkin-Huxley equations.**

These ODEs make use of the following auxiliary equations:

$$h = h_0 - n \tag{A.3}$$

$$m_\infty = \frac{\alpha_m}{\alpha_m + \beta_m} \tag{A.4}$$

$$\alpha_m = \phi \frac{0.1\,(25 - V)}{e^{0.1(25-V)} - 1} \tag{A.5}$$

$$\beta_m = 0.4\phi e^{-\frac{V}{18}}. \tag{A.6}$$

The associated constants are listed in Table A.1.

A.2 Thalamic Neuron Model

Like the Hodgkin-Huxley model, the thalamic neuron model is a conductance-based model, and as such shares the same basic structure as the Hodgkin-Huxley model, but is a three-dimensional model instead:

$$\begin{aligned}\dot{V} &= I_0 + I\left(t\right) - \bar{g}_{Na} m_\infty^3 h \left(V - V_{Na}\right) - \bar{g}_K \left(0.75^4\right) \left(1-h\right)^4 \left(V - V_K\right) \\ &\quad - \bar{g}_t p_\infty^2 r \left(V - V_t\right) - \bar{g}_\ell \left(V - V_\ell\right) \qquad \text{(A.7)} \\ \dot{h} &= \frac{h_\infty - h}{\tau_h} \qquad \text{(A.8)} \\ \dot{r} &= \frac{r_\infty - r}{\tau_r}. \qquad \text{(A.9)}\end{aligned}$$

The associated auxiliary equations are:

$$\begin{aligned}h_\infty &= \frac{1}{1 + e^{\frac{V+41}{4}}} \qquad \text{(A.10)} \\ r_\infty &= \frac{1}{1 + e^{\frac{V+84}{4}}} \qquad \text{(A.11)} \\ \alpha_h &= 0.128 e^{-\frac{V+46}{18}} \qquad \text{(A.12)} \\ \beta_h &= \frac{4}{1 + e^{-\frac{V+23}{5}}} \qquad \text{(A.13)} \\ \tau_h &= \frac{1}{\alpha_h + \beta_h} \qquad \text{(A.14)} \\ \tau_r &= 28 + e^{-\frac{V+25}{10.5}} \qquad \text{(A.15)} \\ m_\infty &= \frac{1}{1 + e^{-V+377}} \qquad \text{(A.16)} \\ p_\infty &= \frac{1}{1 + e^{-V+606.2}}. \qquad \text{(A.17)}\end{aligned}$$

Associated constants are listed in Table A.2.

A.3 STN Neuron Model

Our third conductance-based model, the STN neuron is adapted from the interconnected model of the basal ganglia presented in [33, 57], in turn adapted from [85]. With

Variable	**Value**	**Description**
V_{Na}	50 mV	Sodium current reversal potential
V_K	-90 mV	Potassium current reversal potential
V_ℓ	-70 mV	Leak current reversal potential
V_t	0 mV	Calcium current reversal potential
$\bar{g}_{Na}$	3 nS/μm^2	Maximum sodium current conductance
$\bar{g}_K$	5 nS/μm^2	Maximum potassium current conductance
$\bar{g}_\ell$	0.05 nS/μm^2	Maximum leak current conductance
$\bar{g}_t$	5 nS/μm^2	Maximum calcium current conductance
I_0	5 mA	Base Current

Table A.2: **Table of constants for thalamic neuron model.**

five ODEs to evaluate, it is the most computationally complex of the conductance-based models we considered in this dissertation. For legibility, we will write the voltage as a summation of the variable-dependent currents and subsequently provide equations for said currents. The governing ODEs are:

$$\dot{V} = -\left(I_\ell + I_{Na} + I_K + I_{AHP} + I_{Ca} + I_t\right) + I_0 + I\left(t\right) \tag{A.18}$$

$$\dot{h} = \phi \frac{h_\infty - h}{\tau_h} \tag{A.19}$$

$$\dot{n} = \phi \frac{n_\infty - n}{\tau_n} \tag{A.20}$$

$$\dot{r} = \phi_r \frac{r_\infty - r}{\tau_r} \tag{A.21}$$

$$\dot{[\text{Ca}]} = \epsilon \phi \left(-I_{Ca} - I_t - k_{Ca}\left[\text{Ca}\right]\right), \tag{A.22}$$

with $\phi = 0.75$, $\phi_r = 0.5$, and $\epsilon = 5E - 5$. We note here that whereas in the prior two conductance-based models, the variables consisted of the voltage V and then a set of gating variables, here [Ca] refers to the concentration of calcium ions within the neuron.

The auxiliary equations are:

$$s_\infty = \frac{1}{1+e^{-\frac{V+39}{8}}} \tag{A.23}$$

$$m_\infty = \frac{1}{1+e^{-\frac{V-30}{15}}} \tag{A.24}$$

$$h_\infty = \frac{1}{1+e^{\frac{v+39}{3.1}}} \tag{A.25}$$

$$n_\infty = \frac{1}{1+e^{\frac{v+32}{-8}}} \tag{A.26}$$

$$\tau_h = 1 + \frac{500}{1+e^{\frac{V+57}{3}}} \tag{A.27}$$

$$\tau_n = 1 + \frac{100}{1+e^{\frac{v+80}{26}}} \tag{A.28}$$

$$\tau_r = 7.1 + \frac{17.5}{1+e^{\frac{V-68}{2.2}}} \tag{A.29}$$

$$T_\infty = \frac{1}{1+e^{\frac{V+63}{-7.8}}} \tag{A.30}$$

$$r_{new} = \frac{1}{1+e^{\frac{r-0.25}{-0.07}}} - \frac{1}{1+e^{-\frac{0.25}{-0.07}}}. \tag{A.31}$$

Using these, we can calculate the various currents as:

$$I_\ell = \bar{g}_\ell \left(V - V_\ell\right) \tag{A.32}$$

$$I_{Na} = \bar{g}_{Na} {m_\infty}^3 h \left(V - V_{Na}\right) \tag{A.33}$$

$$I_K = \bar{g}_K n^4 \left(V - V_K\right) \tag{A.34}$$

$$I_{AHP} = \bar{g}_{AHP} \left(V - V_K\right) \frac{[\mathrm{Ca}]}{[Ca] + k_1} \tag{A.35}$$

$$I_{Ca} = \bar{g}_{Ca} {s_\infty}^2 \left(V - V_{Ca}\right) \tag{A.36}$$

$$I_T = \bar{g}_T {T_\infty}^3 {r_{new}}^2 \left(V - V_{Ca}\right). \tag{A.37}$$

The associated constants are listed in Table A.3.

Variable	**Value**	**Description**
V_{Na}	55 mV	Sodium current reversal potential
V_K	-80 mV	Potassium current reversal potential
V_ℓ	-60 mV	Leak current reversal potential
V_{Ca}	140 mV	Calcium current reversal potential
$\bar{g}_{Na}$	37.5 nS/μm^2	Maximum sodium current conductance
$\bar{g}_K$	45 nS/μm^2	Maximum potassium current conductance
$\bar{g}_\ell$	2.25 nS/μm^2	Maximum leak current conductance
$\bar{g}_T$	0.5 nS/μm^2	Maximum T-type calcium current conductance
$\bar{g}_{AHP}$	9 nS/μm^2	Maximum afterhyperpolarization potassium current conductance
$\bar{g}_{Ca}$	0.5 nS/μm^2	Maximum high-threshold calcium current conductance
I_0	25 mA	Base Current

Table A.3: **Table of constants for STN neuron model.**

Appendix B

Stabilizability – Proof of Zero-State Observability

We begin by noting there are several trivial counterexample systems that are not stabilizable, but whose phase response curves and conditions cannot be considered physically relevant. First, if the phase response curve is a constant value (so that stimulation anywhere in the period has the same effect), it is impossible to effect changes in the relative positioning of oscillators. Beyond that, one may devise highly artificial examples, such as the case of two oscillators separated by π radians and a phase response curve given by $\sin 2\theta$, that do not allow for the relative positioning of oscillators to be adjusted. We set these aside and consider, then, the problem subject to some light assumptions.

We will specifically consider here the summation of observables:

$$\sum_{i=1}^{l} \beta_i \frac{\partial v_i}{\partial \Delta\theta_i} \left(Z\left(\theta_j\right) - Z\left(\theta_k\right) \right), \tag{B.1}$$

and we will show it is zero-state observable. We will make several initial observations and assumptions:

1. Since the summation value is 0 regardless, we may set u^* to 0 as well; this has the effect of making $\dot{\theta}_i = \omega$ for each neuron.

2. We assume the phase response curve is continuous and differentiable.

3. Since the PRC is continuous, differentiable, and, by definition, 2π-periodic, we may approximate it with arbitrary accuracy with an infinite Fourier series of the form $a_0 + \sum_{n=1}^{\infty} (a_n \cos n\theta + b_n \sin n\theta)$.

Taken together, these observations require that, for a system to fail to be zero-state observable, advancing all oscillators by a constant phase (since they oscillate with the same natural frequency) should remain within the invariant set of solutions where the observable is equal to 0 but where at least one of the coefficients $\frac{\partial v_i}{\partial \Delta\theta_i} \neq 0$ (since this coefficient is, by construction equal to 0 only at the target separation). Without loss of generality, we may rewrite the observable at a state where it is equal to 0 as a function of the k oscillators whose PRCs are utilized in the $k-1$ phase difference equations necessary to fully constrain the system to a specified state:

$$\sum_{i=1}^{k} c_i Z\left(\theta_i\right) = 0, \tag{B.2}$$

again noting at least one value c_i should be nonzero. Because the coefficients c_i depend only on the separation between neurons and not their absolute locations in phase space, we additionally note that they do not vary as the system oscillates if all oscillators advance with the same rate. We now substitute in our Fourier series and advance all the oscillators by Δx such that our new observable value is equal to:

$$\sum_{i=1}^{k} c_i \sum_{n=1}^{\infty} \left[a_n \cos n\left(\theta_i + \Delta x\right) + b_n \sin n\left(\theta_i + \Delta x\right)\right]. \tag{B.3}$$

We then expand and rewrite the sines and cosines, yielding:

$$\sum_{i=1}^{k} c_i \sum_{n=1}^{\infty} \left[a_n \left(\cos n\theta_i \cos n\Delta x - \sin n\theta_i \sin n\Delta x\right) + b_n \left(\sin n\theta_i \cos n\Delta x + \cos n\theta_i \sin n\Delta x\right)\right]. \tag{B.4}$$

We may now rewrite this as a Fourier series for Δx, grouping sine and cosine terms appropriately:

$$\sum_{n=1}^{\infty} \sum_{i=1}^{k} \left[c_i \left(a_n \cos n\theta_i + b_n \sin n\theta_i\right) \cos n\Delta x + c_i \left(b_n \cos n\theta_i - a_n \sin n\theta_i\right) \sin n\Delta x\right]. \tag{B.5}$$

For the value of this equation to be identically 0 for all values of Δx, the Fourier coefficients for each Δx must be equal to 0. This gives us two equations for each value of n:

$$a_n \sum_{i=1}^{k} c_i \cos n\theta_i + b_n \sum_{i=1}^{k} c_i \sin n\theta_i = 0, \tag{B.6}$$

$$-a_n \sum_{i=1}^{k} c_i \sin n\theta_i + b_n \sum_{i=1}^{k} c_i \cos n\theta_i = 0. \tag{B.7}$$

This system of equations is readily solved by noting that the coefficients are identical but flipped between a_n and b_n, with the addition of a negative:

$$C_1 a_n + C_2 b_n = 0, \tag{B.8}$$

$$-C_2 a_n + C_1 b_n = 0. \tag{B.9}$$

Solving this system of equations yields $C_1 = 0$ and $C_2 = 0$, so for our system to be zero-state observable:

$$\sum_{i=1}^{k} c_i \sin n\theta_i = 0 \tag{B.10}$$

and

$$\sum_{i=1}^{k} c_i \cos n\theta_i = 0. \tag{B.11}$$

We will demonstrate this is impossible by contradiction. If both equal 0, then their sums equal 0 as well, as do the sums of their squares:

$$\left(\sum_{i=1}^{k} c_i \cos n\theta_i\right)^2 + \left(\sum_{i=1}^{k} c_i \sin n\theta_i\right)^2 = 0. \tag{B.12}$$

Carrying out this multiplication and combining terms yields:

$$\sum_{i=1}^{k} \left(c_i^2 \cos^2 n\theta_i + c_i^2 \sin^2 n\theta_i + c_i \cos n\theta_i \sum_{j\neq i}^{k} c_j \cos n\theta_j + c_i \sin n\theta_i \sum_{j\neq i}^{k} c_j \sin n\theta_j\right) = 0. \tag{B.13}$$

Simplifying further by combining the squared sines and cosines and noting $\cos u \cos j + \sin u \sin j = \cos u - j$ allows us to rewrite this as:

$$\sum_{i=1}^{k} \left[c_i^2 + c_i \sum_{j\neq i}^{k} c_j \cos n\left(\theta_i - \theta_j\right)\right] = 0. \tag{B.14}$$

The challenge, however, is that the first term in (B.14) is independent of n, whereas the second term is not. This implies that, for every n such that $a_n \neq$ or $b_n \neq 0$, the value of the summation must remain constant:

$$\sum_{i=1}^{k} c_i \sum_{j\neq i}^{k} c_j \cos\left(\theta_i - \theta_j\right) = \sum_{i=1}^{k} c_i \sum_{j\neq i}^{k} c_j \cos 2\left(\theta_i - \theta_j\right) = \cdots = \sum_{i=1}^{k} c_i \sum_{j\neq i}^{k} c_j \cos n\left(\theta_i - \theta_j\right). \tag{B.15}$$

If the solution set for the phase differences is finite, it is impossible for this equality to hold if the Fourier series is infinite. Suppose $\mathbf{x} \in \mathcal{D}$ is the set of all points that are the solution for $n = 1$; if a solution exists, this means that there is some subset of points on which the transformation $\mathbf{x} \to 2\mathbf{x}$ is invariant on $\mathcal{D}$. However, the transformation $\mathbf{x} \to 3\mathbf{x}$

must also be contained in $\mathcal{D}$, as must $2\mathbf{x} \to 3\mathbf{x}$. This in turn means the transformation $\mathbf{x} \to 1.5\mathbf{x}$ is invariant on $\mathcal{D}$. Indeed, it follows that:

$$\frac{p}{q}\mathbf{x} \in \mathcal{D}, \qquad p, q \in \mathbb{Z}^+. \tag{B.16}$$

As there are infinite such ratios of p and q, we must have either infinite solutions or exactly one. Since $\mathbf{x} = \vec{0}$ is a solution but is disallowed by assumption, the only alternative is that we must have infinite solutions.

As a result, we need only show the solution set is finite. We can demonstrate this is the case. If we allow $c_1^2 = \cdots = c_k^2$, then:

$$\min_{c_i = \pm c_1} \sum_{i=1}^{k} c_i \sum_{j \neq i}^{k} c_j \cos\left(\theta_i - \theta_j\right) = -kc_i^2. \tag{B.17}$$

Since this is exactly what is required for (B.14) to be equal to 0, we know that the $\mathbb{R}^{k-1}$ hypersurface representing values of the double sum on the set of all possible phase differences $\Delta\theta$ intersects the hyperplane representing the required solution not at an infinite set of points in the modulo 2π set, but rather a finite number (since the value represents a global minimum). Therefore, the set containing all solutions to:

$$\sum_{i=1}^{k} c_i \sum_{j \neq i}^{k} c_j \cos\left(\theta_i - \theta_j\right) = -kc_i^2 \tag{B.18}$$

cannot simultaneously contain all the requisite solutions to the infinite Fourier series.

Any oscillator with phase response curve with an infinite Fourier series representation, then, is zero-state observable; it is only in so-called "toy" problems with trivial phase response curves (such as a Kuramoto oscillator) where the splay state represents an invariant set. This analysis, however, precluded the application of an input stimulus.

The presence of an input stimulus immediately removes a system of oscillators from the splay state, at which point we have left the invariant set, even if the value of the observable in the new state is still identically 0. Therefore, there exists no solution outside of trivial or disallowed conditions where the system is not stabilizable.

Bibliography

[1] S. DeWeerdt, *How to map the brain*, *Nature* **571** (2019) S6–S8.

[2] R. A. Poldrack and M. J. Farah, *Progress and challenges in probing the human brain*, *Nature* **526** (2015) 371–379.

[3] M. Rosenblum and A. Pikovsky, *Delayed feedback control of collective synchrony: An approach to suppression of pathological brain rhythms*, *Physical Review E* **70** (2004) 041904.

[4] Y. Kuramoto, *Chemical Oscillations, Waves, and Turbulence*, vol. 19 of *Springer Series in Synergetics*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1984.

[5] E. Brown, J. Moehlis, and P. Holmes, *On the phase reduction and response dynamics of neural oscillator populations*, *Neural Computation* **16** (2004) 673–715.

[6] P. Sacré and R. Sepulchre, *Sensitivity analysis of oscillator models in the space of phase-response curves: Oscillators as open systems*, *IEEE Control Systems* **34** (2014) 50–74.

[7] J.-S. Li, I. Dasanayake, and J. Ruths, *Control and synchronization of neuron ensembles*, *IEEE Transactions on Automatic Control* **58** (2013) 1919–1930.

[8] J. Moehlis, E. Shea-Brown, and H. Rabitz, *Optimal Inputs for Phase Models of Spiking Neurons*, *Journal of Computational and Nonlinear Dynamics* **1** (2006) 358.

[9] B. Monga, G. Froyland, and J. Moehlis, *Synchronizing and Desynchronizing Neural Populations through Phase Distribution Control*, *American Controls Conference* (2018) 2808–2813.

[10] A. Zlotnik and J.-S. Li, *Optimal subharmonic entrainment of weakly forced nonlinear oscillators*, *SIAM Journal on Applied Dynamical Systems* **13** (2014) 1654–1693.

[11] A. Zlotnik, R. Nagao, I. Z. Kiss, and J.-S. Li, *Phase-selective entrainment of nonlinear oscillator ensembles*, *Nature Communications* **7** (2016) 1–7.

[12] D. Wilson and J. Moehlis, *Isostable reduction of periodic orbits*, *Physical Review E* **94** (2016) 1–7.

[13] W. Govaerts and B. Sautois, *Computation of the Phase Response Curve: A Direct Numerical Approach*, *Neural Computation* **18** (2006) 817–847.

[14] E. Phoka, H. Cuntz, A. Roth, and M. Häusser, *A new approach for determining phase response curves reveals that purkinje cells can act as perfect integrators*, *PLoS Computational Biology* **6** (2010).

[15] A. Mauroy, I. Mezić, and J. Moehlis, *Isostables, isochrons, and Koopman spectrum for the action-angle representation of stable fixed point dynamics*, *Physica D: Nonlinear Phenomena* **261** (2013) 19–30.

[16] A. Mauroy, B. Rhoads, J. Moehlis, and I. Mezić, *Global Isochrons and Phase Sensitivity of Bursting Neurons*, *SIAM Journal on Applied Dynamical Systems* **13** (2014) 306–338.

[17] H. Yuan, Z. W. Zhang, L. W. Liang, Q. Shen, X. D. Wang, S. M. Ren, H. J. Ma, S. J. Jiao, and P. Liu, *Treatment strategies for Parkinson's disease*, *Neuroscience Bulletin* **26** (2010) 66–76.

[18] H. Russmann, J. Ghika, P. Combrement, J. G. Villemure, J. Bogousslavsky, P. R. Burkhard, and F. J. Vingerhoets, *L-Dopa-induced dyskinesia improvement after STN-DBS depends upon medication reduction*, *Neurology* **63** (2004) 153–155.

[19] S. J. Groiss, L. Wojtecki, M. Sudmeyer, and A. Schnitzler, *Deep brain stimulation in Parkinson's disease*, *Therapeutic Advances in Neurological Disorders* **2** (2009) 379–391.

[20] J. G. Nutt, W. R. Woodward, J. P. Hammerstad, J. H. Carter, and J. L. Anderson, *The "on-off" phenomenon in Parkinson's disease. Relation to levodopa absorption and transport*, *The New England Journal of Medicine* **310** (1984) 483–8.

[21] T. M. Herrington, J. J. Cheng, and E. N. Eskandar, *Mechanisms of deep brain stimulation*, *Journal of Neurophysiology* **115** (2016) 19–38.

[22] The Deep-Brain Stimulation for Parkinson's Disease Study Group, *Deep-brain stimulation of the subthalamic nucleus or the pars interna of the globus pallidus in Parkinson's disease*, *New England Journal of Medicine* **345** (2001) 956–963.

[23] I. Adamchic, C. Hauptmann, U. B. Barnikol, N. Pawelczyk, O. Popovych, T. T. Barnikol, A. Silchenko, J. Volkmann, G. Deuschl, W. G. Meissner, M. Maarouf, V. Sturm, H.-J. Freund, and P. A. Tass, *Coordinated reset neuromodulation for Parkinson's disease: Proof-of-concept study*, *Movement Disorders* **29** (2014) 1679–1684.

[24] B. Lysyansky, O. V. Popovych, and P. A. Tass, *Desynchronizing anti-resonance effect of m:n ON-OFF coordinated reset stimulation*, *Journal of Neural Engineering* **8** (2011) 036019.

[25] B. Lysyansky, O. V. Popovych, and P. A. Tass, *Optimal number of stimulation contacts for coordinated reset neuromodulation*, *Frontiers in Neuroengineering* **6** (2013) 5.

[26] R. Savica, M. Stead, K. J. Mack, K. H. Lee, and B. T. Klassen, *Deep brain stimulation in Tourette syndrome: A description of 3 patients with excellent outcome*, *Mayo Clinic Proceedings* **87** (2012) 59–62.

[27] A. Wagle Shukla, P. Zeilman, H. Fernandez, J. A. Bajwa, and R. Mehanna, *DBS programming: an evolving approach for patients with Parkinson's Disease*, *Parkinson's Disease* **2017** (2017) 1–11.

[28] A. Beric, P. J. Kelly, A. Rezai, D. Sterio, A. Mogilner, M. Zonenshayn, and B. Kopell, *Complications of deep brain stimulation surgery*, *Stereotactic and Functional Neurosurgery* **77** (2002) 73–78.

[29] M. C. Rodriguez-Oroz, J. A. Obeso, A. E. Lang, J. L. Houeto, P. Pollak, S. Rehncrona, J. Kulisevsky, A. Albanese, J. Volkmann, M. I. Hariz, N. P. Quinn, J. D. Speelman, J. Guridi, I. Zamarbide, A. Gironell, J. Molet, B. Pascual-Sedano, B. Pidoux, A. M. Bonnet, Y. Agid, J. Xie, A. L. Benabid, A. M. Lozano, J. Saint-Cyr, L. Romito, M. F. Contarino, M. Scerrati, V. Fraix, and N. Van Blercom, *Bilateral deep brain stimulation in Parkinson's disease: A multicentre study with 4 years follow-up*, *Brain* **128** (2005) 2240–2249.

[30] P. A. Tass, *A model of desynchronizing deep brain stimulation with a demand-controlled coordinated reset of neural subpopulations*, *Biological Cybernetics* **89** (2003) 81–88.

[31] D. Wilson and J. Moehlis, *Clustered desynchronization from high-frequency deep brain stimulation*, *PLoS Computational Biology* **11** (2015) 1–26.

[32] C. J. Wilson, B. Beverlin, and T. Netoff, *Chaotic desynchronization as the therapeutic mechanism of deep brain stimulation*, *Frontiers in Systems Neuroscience* **5** (2011) 50.

[33] J. E. Rubin and D. Terman, *High frequency stimulation of the subthalamic nucleus eliminates pathological thalamic rhythmicity in a computational model*, *Journal of Computational Neuroscience* **16** (2004) 211–235.

[34] L. Lücken, S. Yanchuk, O. V. Popovych, and P. A. Tass, *Desynchronization boost by non-uniform coordinated reset stimulation in ensembles of pulse-coupled neurons*, *Frontiers in Computational Neuroscience* **7** (2013) 63.

[35] C. C. Chen, V. Litvak, T. Gilbertson, A. Kühn, C. S. Lu, S. T. Lee, C. H. Tsai, S. Tisch, P. Limousin, M. Hariz, and P. Brown, *Excessive synchronization of basal ganglia neurons at 20 Hz slows movement in Parkinson's disease*, *Experimental Neurology* **205** (2007) 214–221.

[36] C. Hammond, H. Bergman, and P. Brown, *Pathological synchronization in Parkinson's disease: networks, models and treatments*, *Trends in Neurosciences* **30** (2007) 357–364.

[37] R. Levy, W. Hutchison, A. Lozano, and J. Dostrovsky, *High-frequency synchronization of neuronal activity in the subthalamic nucleus of Parkinsonian patients with limb tremor*, *The Journal of Neuroscience* **20** (2000) 7766–7775.

[38] A. Schnitzler and J. Gross, *Normal and pathological oscillatory communication in the brain*, *Nature Reviews. Neuroscience* **6** (2005) 285–96.

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

[40] S. Little and P. Brown, *What brain signals are suitable for feedback control of deep brain stimulation in Parkinson's disease?*, *Annals of the New York Academy of Sciences* **1265** (2012) 9–24.

[41] S. Little, A. Pogosyan, S. Neal, B. Zavala, L. Zrinzo, M. Hariz, T. Foltynie, P. Limousin, K. Ashkan, J. FitzGerald, A. L. Green, T. Z. Aziz, and P. Brown, *Adaptive deep brain stimulation in advanced Parkinson disease*, *Annals of Neurology* **74** (2013) 449–457.

[42] B. A. Mitchell and L. R. Petzold, *Control of neural systems at multiple scales using model-free, deep reinforcement learning*, *Scientific Reports* **8** (2018) 1–12.

[43] V. Nagaraj, A. Lamperski, and T. I. Netoff, *Seizure control in a computational model using a reinforcement learning stimulation paradigm*, *International Journal of Neural Systems* **27** (2016).

[44] G. Cybenko, *Approximation by superpositions of a sigmoidal function*, *Mathematics of Control, Signals, and Systems* **2** (1989) 303–314.

[45] T. D. Matchen and J. Moehlis, *Phase model-based neuron stabilization into arbitrary clusters*, *Journal of Computational Neuroscience* **44** (2018) 363–378.

[46] B. Monga, D. Wilson, T. Matchen, and J. Moehlis, *Phase reduction and phase-based optimal control for biological systems: a tutorial*, *Biological Cybernetics* **113** (2019).

[47] H. Khalil, *Nonlinear Control.* Pearson, New York, NY, 1 ed., 2015.

[48] J. Keener and J. Sneyd, *Mathematical Physiology.* Springer New York, New York, NY, 2009.

[49] A. Hodgkin and A. Huxley, *A quantitative description of membrane current and its application to conduction and excitation in nerve*, *Journal of Physiology* (1952) 500–544.

[50] G. B. Ermentrout and D. H. Terman, *Mathematical Foundations of Neuroscience*, vol. 35 of *Interdisciplinary Applied Mathematics.* Springer New York, New York, NY, 2010.

[51] R. F. Galan, G. B. Ermentrout, and N. N. Urban, *Efficient estimation of phase-resetting curves in real neurons and its significance for neural-network modeling*, *Physical Review Letters* **94** (2005) 1–4.

[52] D. Johnston and S. M.-S. Wu, *Foundations of Cellular Neurophysiology.* MIT Press, Cambridge, MA, 1 ed., 1995.

[53] G. S. Schmidt, D. Wilson, F. Allgower, and J. Moehlis, *Selective Averaging with Application to Phase Reduction and Neural Control*, *Nonlinear Theory and Its Applications, IEICE* **5** (2014) 424–435.

[54] D. Golomb and D. Hansel, *The number of synaptic inputs and the synchrony of large, sparse neuronal networks.*, *Neural Computation* **12** (2000) 1095–139.

[55] S. E. Hua, F. a. Lenz, T. a. Zirh, S. G. Reich, and P. M. Dougherty, *Thalamic neuronal activity correlated with essential tremor*, *Journal of Neurology, Neurosurgery, and Psychiatry* **64** (1998) 273–276.

[56] H. Cagnan, J. S. Brittain, S. Little, T. Foltynie, P. Limousin, L. Zrinzo, M. Hariz, C. Joint, J. Fitzgerald, A. L. Green, T. Aziz, and P. Brown, *Phase dependent modulation of tremor amplitude in essential tremor through thalamic stimulation*, *Brain* **136** (2013) 3062–3075.

[57] J. E. Rubin and D. Terman, *High Frequency Stimulation of the Subthalamic Nucleus Eliminates Pathological Thalamic Rhythmicity in a Computational Model*, *Journal of Computational Neuroscience* **16** (2004) 211–235.

[58] P. J. Hahn and C. C. McIntyre, *Modeling shifts in the rate and pattern of subthalamopallidal network activity during deep brain stimulation*, *Journal of Computational Neuroscience* **28** (2010) 425–441.

[59] B. Monga and J. Moehlis, *Optimal phase control of biological oscillators using augmented phase reduction*, *Biological Cybernetics* (2018) 1–18.

[60] P. Danzl, J. Hespanha, and J. Moehlis, *Event-based minimum-time control of oscillatory neuron models: Phase randomization, maximal spike rate increase, and desynchronization*, *Biological Cybernetics* **101** (2009) 387–399.

[61] R. Bellman, *Dynamic Programming*, *Science* **153** (1966) 34–37.

[62] A. T. Winfree, *The Geometry of Biological Time*, vol. 12 of *Interdisciplinary Applied Mathematics*. Springer New York, New York, NY, 2001.

[63] O. Castejón, A. Guillamon, and G. Huguet, *Phase-Amplitude response functions for Transient-State stimuli*, *Journal of Mathematical Neuroscience* **3** (2013) 1–26.

[64] A. Nabi, M. Mirzadeh, F. Gibou, and J. Moehlis, *Minimum energy desynchronizing control for coupled neurons*, *Journal of Computational Neuroscience* **34** (2013) 259–271.

[65] X. Glorot and Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks*, in *International Conference on Artificial Intelligence and Statistics*, vol. 9, pp. 249–256, 2010.

[66] H. Daido, *Onset of cooperative entrainment in limit-cycle oscillators with uniform all-to-all interactions: Bifurcation of the order function*, *Physica D: Nonlinear Phenomena* **91** (1996) 24–66.

[67] P. A. Tass, *Desynchronization by means of a coordinated reset of neural sub-populations - A novel technique for demand-controlled deep brain stimulation*, *Progress of Theoretical Physics Supplement* **150** (2003) 281–296.

[68] R. L. Honeycutt, *Stochastic Runge-Kutta algorithms. I. White noise*, *Physical Review A* **45** (1992) 600–603.

[69] J. W. Durham, *Controlling Canards Using Ideas From The Theory of Mixed-Mode Oscillations*. PhD thesis, University of California, Santa Barbara, 2007.

[70] A. T. Connolly, A. Muralidharan, C. Hendrix, L. Johnson, R. Gupta, S. Stanslaski, T. Denison, K. B. Baker, J. L. Vitek, and M. D. Johnson, *Local field potential recordings in a non-human primate model of Parkinsons disease using the Activa PC + S neurostimulator*, *Journal of Neural Engineering* **12** (2015) 066012.

[71] P. Afshar, A. Khambhati, S. Stanslaski, D. Carlson, R. Jensen, S. Dani, M. Lazarewicz, J. Giftakis, P. Stypulkowski, and T. Denison, *A translational platform for prototyping closed-loop neuromodulation systems*, *Frontiers in Neural Circuits* **6** (2013) 1–15.

[72] N. C. Swann, C. de Hemptinne, S. Miocinovic, S. Qasim, J. L. Ostrem, N. B. Galifianakis, M. S. Luciano, S. S. Wang, N. Ziman, R. Taylor, and P. A. Starr, *Chronic multisite brain recordings from a totally implantable bidirectional neural interface: experience in 5 patients with Parkinson's disease*, *Journal of Neurosurgery* **128** (2018) 605–616.

[73] I. Jolliffe, *Principal Component Analysis*, in *International Encyclopedia of Statistical Science*, pp. 1094–1096. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[74] P. J. Schmid, *Dynamic mode decomposition of numerical and experimental data*, *Journal of Fluid Mechanics* **656** (2010) 5–28.

[75] S. van Albada and P. Robinson, *Mean-field modeling of the basal ganglia-thalamocortical system. I*, *Journal of Theoretical Biology* **257** (2009) 642–663.

[76] S. van Albada, R. Gray, P. Drysdale, and P. Robinson, *Mean-field modeling of the basal ganglia-thalamocortical system. II*, *Journal of Theoretical Biology* **257** (2009) 664–688.

[77] L. L. Grado, M. D. Johnson, and T. I. Netoff, *Bayesian adaptive dual control of deep brain stimulation in a computational model of Parkinson's disease*, *PLoS Computational Biology* (2018) 1–23.

[78] B. E. Boser, V. N. Vapnik, and I. M. Guyon, *Training Algorithm Margin for Optimal Classifiers*, *Perception* (1992) 144–152.

[79] S. L. Brunton, J. L. Proctor, J. N. Kutz, and W. Bialek, *Discovering governing equations from data by sparse identification of nonlinear dynamical systems*, *Proceedings of the National Academy of Sciences of the United States of America* **113** (2016) 3932–3937.

[80] A. Güneş Baydin, B. A. Pearlmutter, A. Andreyevich Radul, and J. Mark Siskind, *Automatic differentiation in machine learning: A survey*, *Journal of Machine Learning Research* **18** (2018) 1–43.

[81] J. Guckenheimer and P. Holmes, *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*, vol. 42 of *Applied Mathematical Sciences.* Springer New York, New York, NY, 1983.

[82] S. H. Strogatz, *From Kuramoto to Crawford: Exploring the onset of synchronization in populations of coupled oscillators*, *Physica D: Nonlinear Phenomena* **143** (2000) 1–20.

[83] A. J. Lotka, *Elements of Physical Biology.* Williams & Wilkins, 1925.

[84] V. Volterra, *Variazioni e fluttuazioni del numero d'individui in specie animali conviventi.* Societá anonima tipografica "Leonardo da Vinci", 1927.

[85] D. Terman, J. E. Rubin, A. C. Yew, and C. J. Wilson, *Activity Patterns in a Model for the Subthalamopallidal Network of the Basal Ganglia*, *The Journal of Neuroscience* **22** (2002) 2963–2976.