Provably Optimal Design of a Brain-Computer Interface

Yin Zhang

CMU-RI-TR-18-63
August 2018

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Steven M. Chase, Carnegie Mellon University, Chair
Robert E. Kass, Carnegie Mellon University
J. Andrew Bagnell, Carnegie Mellon University
Patrick J. Loughlin, University of Pittsburgh

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

Keywords: Brain-Computer Interface, Closed-Loop System, Optimal Control

Abstract

Brain-computer interfaces are in the process of moving from the laboratory to the clinic. These devices act by reading neural activity and using it to directly control a device, such as a cursor on a computer screen. Over the past two decades, much attention has been devoted to the decoding problem: how should recorded neural activity be translated into movement of the device in order to achieve the most proficient control? This question is complicated by the fact that learning, especially the long-term skill learning that accompanies weeks of practice, can allow subjects to improve performance over time. Typical approaches to this problem attempt to maximize the biomimetic properties of the device, to limit the need for extensive training. However, it is unclear if this approach would ultimately be superior to performance that might be achieved with a non-biomimetic device, once the subject has engaged in extended practice and learned how to use it. In this thesis, I first recast the decoder design problem from a physical control system perspective, and investigate how various classes of decoders lead to different types of physical systems for the subject to control. This framework leads to new interpretations of why certain types of decoders have been shown to perform better than others. Based on this framework, I present a formal definition of the usability of a device under the assumption that the brain acts as an optimal controller. Using ideas from optimal control theory, it can be shown that the optimal, post-learning mapping can be written as the solution of a constrained optimization problem which maximizes this usability. I then derive the optimal mappings for particular cases common to most brain-computer interfaces. Results suggest that the common approach of creating biomimetic interfaces may not be optimal when learning is taken into account. More broadly, this method provides a blueprint for optimal device design in general control-theoretic contexts.

Given the optimal, post-learning mapping, successful implementation of such brain-computer interface depends critically on the subject's ability to learn how to modulate the neurons controlling the device. However, the subject's learning process is probably the least understood aspect of the control loop. An effective training schedule should manipulate the difficulty of the task to provide enough information to guide improvement without overwhelming the subject. In this thesis, I introduce a Bayesian framework for modeling the closed-loop BCI learning process that treats the subject as a bandwidth-limited communication channel. I then develop an adaptive algorithm to find the optimal difficulty-schedule for performance improvement. Simulation results demonstrate that this algorithm yields faster learning rates than several other heuristic training schedules, and provides insight into the factors that might affect the learning process.

Acknowledgments

My foremost thanks go to my advisor, Dr. Steven M. Chase, for his unfailing support and generous mentorship. I had little knowledge of neuroscience when I joined Carnegie Mellon, Dr. Chase is very patient and helps me to catch up. I am very glad he has not given up on me. I've learned a tremendous amount from him during my journey, and I'm eternally grateful and in debt for his guidance and support. I would not have completed this thesis without his support.

I would like to thank my first year's advisors, Dr. Robert E. Kass and Dr. Andrew B. Schwartz. They led me to the Sci-Fi like technique of brain-computer interface. I also want to thank other members of my thesis committee, Dr. Patrick J. Loughlin and Dr. J. Andrew Bagnell, for providing invaluable feedback throughout this study.

I've benefited greatly from my interactions with many wonderful colleagues, Sam Clanton, Sagi Perel, Matthew Golub, William Bishop, Xiao Zhou, Emily Oby, Scott Kennedy and Jiuguang Wang, for their help over the course of my academic career. I would like to thank all the friends I made at Pittsburgh for all the fun we had together. I would like to thank fellow students, staff, and the faculty of Robotics Institute for creating an inclusive and productive environment.

Last but not least, I want to thank my mother and my wife for years of support and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 What are BCIs?

Brain-computer interfaces (BCIs), also known as brain-machine interfaces, are a powerful class of assistive devices that may one day restore movement ability to paralyzed individuals [Lebedev and Nicolelis, 2017, Schwartz et al., 2006]. According to a study initiated by the Christopher and Dana Reeve Foundation [2009], approximately 1.9% of the U.S. population, or some $5,596,000$ people, reported some form of paralysis that impacts the performance of tasks of daily living. The leading cause of paralysis was stroke (29%), followed by spinal cord injury (23%) and multiple sclerosis (17%). When the neurons in the spinal cord are damaged, they face anatomical, physiological, chemical, and immune system changes that hinder their attempts at regeneration. Therefore, most of the time there is no specific treatment for brain and spinal cord injuries. Brain-computer interfaces, however, may alleviate symptoms of paralysis by providing an alternate way to actuate devices. BCIs read and translate neural activity into motor commands of assistive devices such as a cursor on a computer screen [Aflalo et al., 2015, Carmena et al., 2003, Gilja et al., 2012, 2015, Serruya et al., 2002, Taylor et al., 2002], a robotic arm [Chapin et al., 1999, Collinger et al., 2013, Hochberg et al., 2006, 2012, Schwartz et al., 2006, Velliste et al., 2008], or even paralyzed muscle tissue [Bouton et al., 2016, Ethier et al., 2012, Moritz et al., 2008], bypassing the defective neural transmission and the muscle activation. Fig. 1.1 shows the closed loop BCI system adopted in [Velliste et al., 2008] where a monkey was implanted with a 96 channel intracortical microelectrode "Utah" array in its primary motor cortex. The recording array extracts information from populations of recorded single neurons. Through a BCI decoder translating the recorded neural activity into movement commands, the monkey learned to control a 4 degree-of-freedom robotic arm to feed itself. Ongoing BCI research is focusing on building technological bridges between neural activity and external devices in order to restore motor function for those people with injury or disease, such as brainstem stroke or loss of a limb, which can disconnect the cortex from its effector target. However, much work yet needs to be done to give subjects control over artificial limbs that might rival control of the natural limb [Gilja et al., 2012, Shenoy and Carmena, 2014]. Therefore, the design of BCI control algorithms that might enable stable and robust control is an active area of research.

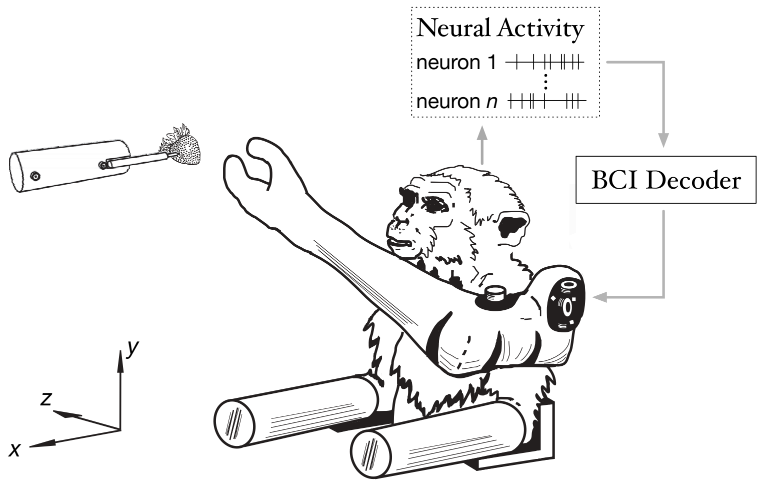


Figure 1.1: **A monkey controlling a robotic arm through the closed loop BCI system to feed itself.** This figure is adapted from [Velliste et al., 2008].

1.1.1 Brief history of BCIs

Depending on different recording devices, BCIs can be divided into two categories: non-invasive BCIs and invasive BCIs. The earliest non-invasive BCIs was designed by Vidal in the 1970's [Vidal, 1973, 1977]. In his experiment, electroencephalography (EEG) signals, which monitor the electrical activity of the brain, were recorded and used to control a "cursor" on a computer screen to solve a maze by moving in four directions [Vidal, 1977]. Later, in the 1980's, Elbert et al. [1980] showed that people can change their EEG activity to control the vertical movements of a rocket image traveling across a television screen. Farwell and Donchin [1988] demonstrated that, by using the P300 event-related potential (a positive voltage deflection of the electrocortical potential 300-500 ms post a rare event that initiates sensory and mental processing), subjects can spell words on a computer screen. Experiments in [Wolpaw et al., 1991] showed that the mu-rhythm recorded from the scalp over the central sulcus of one hemisphere can be translated into cursor movements in 1 dimension. Early clinical trials with EEG BCIs have provided impressive proof-of-concept demonstrations of their clinical potential. Wolpaw and McFarland [2004] demonstrated that by using EEG-based signals and an adaptive algorithm, patients with spinal cord injuries can control the computer cursor to make 2-dimensional point-to-point movements.

One limitation of EEG as a signal source for BCI is the relatively low spatial resolution of the signal, which makes it impossible to accurately localize the source of the neural activity. In contrast, functional magnetic resonance imaging (fMRI) measures the blood oxygen level-dependent (BOLD) signal across the entire brain with relatively high spatial resolution, though with low temporal resolution (in the range of millimeters and seconds) [Goebel et al., 2010, Sitaram et al., 2007]. Despite the fact that BOLD is an indirect measure, there is a strong correlation between the BOLD signal and the electric brain activity [Logothetis et al., 2001]. The first online fMRI-based BCI experiment that allowed for some degree of external control was conducted by Yoo et al. [2004]. In their experiment, participants were asked to perform four different mental tasks that evoked differential brain activation at four distinct brain locations and those mental tasks were interpreted as four predetermined BCI commands (up, down, left and right) in order to allow the subject to virtually navigate through a simple 2D maze. Later, they demonstrated that it is also possible to control 2D movements of a robotic arm by using the same principles [Lee et al., 2009]. Those studies needed the data to be averaged across many trials to provide reliable differentiation. In [Goebel et al., 2004], authors demonstrated that participants could play an analogue of the computer game 'pong' simply by adjusting their single-trial brain activation level. However, due to its high cost and complexity of development and usage, fMRI-based BCI has limited practicality in clinical trials.

To date, non-invasive recording devices which measure brain activity by detecting changes associated with blood flow suffer from low information content within the recorded signals. Thus, without more information-rich signals, most BCI systems based on non-invasive recording devices can only output categorical commands, such as one digit from 0 to 9, or one movement direction from four possible directions (up, down, left and right). In order to get more accurate signals, invasive recoding devices, such as the intracortical microelectrode, can be used. Each intracortical microelectrode is sensitive enough to pick up the action potentials, commonly referred as *spikes*, of a few neurons (typically 1 or 2). Spikes are nearly universally employed by neurons for communication in the brain. Currently, the highest levels of performance to date have been achieved using intracortical spiking recorded from penetrating electrodes.

The first invasive BCI was arguably designed by Fetz in 1969 [Fetz, 1969]. It used an implanted stainless steel microelectrode to collect neural activity from the primary motor cortex of a Rhesus macaque (Macaca mulatta). Spikes from a single neuron (called a 'unit') were isolated from extracellular voltage signals recorded by the microelectrode. In his experiment, Fetz demonstrated that monkeys could volitionally increase the activity of isolated single neurons in order to receive food rewards. In the 2000's, microelectrode arrays started to be widely used to collect the firing patterns of many neurons simultaneously. In 2006, a 96-channel intracortical microelectrode array was implanted in the primary motor cortex (M1) of a tetraplegic human [Hochberg et al., 2006]. Using the recorded neuronal ensemble activity, the patient was able to control a BCI system to open simulated e-mail, operate devices such as a television, open and close a prosthetic hand, and perform rudimentary actions with a multijointed robotic arm. In 2012, they further demonstrated the ability of two people with long-standing tetraplegia to use a BCI system to control a robotic arm to perform 3-dimensional reach and grasp movements [Hochberg et al., 2012]. Collinger et al. [2013] implanted two 96-channel microelectrodes in the motor cortex of a 52-year-old individual with tetraplegia and demonstrated a patient who could use a BCI to control an anthropomorphic prosthetic limb with 7 degrees of freedom (3-dimensional translation, 3-dimensional orientation, 1-dimensional grasping). Later work by this group demonstrated simultaneous control of a 10 degree-of-freedom robotic arm, the highest achieved to date [Wodlinger et al., 2014]. A study published in 2015 [Gilja et al., 2015] demonstrated that two individuals with amyotrophic lateral sclerosis implanted with intracortical microelectrode arrays could control the cursor finishing center-out-and-back and random point-to-point target-acquisition tasks. Their experiments achieved the lowest target acquisition time reported to date.

New BCI capabilities continue to be demonstrated. Researchers have now demonstrated control of external devices by neurons recorded outside of primary motor cortex, including the posterior parietal cortex [Aflalo et al., 2015] and even, in mice, primary visual cortex [Neely et al., 2018]. Researches and clinicians are also starting to reanimate the user's own paralyzed limb. Bouton et al. [2016] showed that intracortically recorded signals can be linked in real-time to muscle activation to restore movement in a paralyzed human, and Ajiboye et al. [2017] used functional electrical stimulation to restore reach and grasp ability.

The reports above are only a snapshot of the current capabilities of BCI systems. In the sections below, I will break down the central components common to BCI systems, and the assumptions they typically make when mapping recorded neural activity to device activation.

1.2 Components of a BCI system

A typical BCI system consists of four components (Fig. 1.1). The first component is a recording device that detects neural activity resulting from the user imagining or attempting a movement. Different recording devices will provide different types of neural signals and thus will affect how we interpret the user's intent. The accuracy of the recording device will also determine whether the system can achieve continuous control or discrete control. In this thesis, I will focus exclusively on invasive recording technologies (like intracortical microelectrode arrays), since these have demonstrated the highest fidelity control to date. Thus, the recorded neural signals I will consider are spikes of a population of neurons from the subject's primary motor cortex.

The second component of a general BCI system is the decoding algorithm (also called the decoder, or mapping) which specifies how recorded signals get translated into movement commands. The movement commands could be the external device's kinematics, such as the positions, velocities, and accelerations [Hochberg et al., 2012, Velliste et al., 2008] or the kinetics, such as the forces and torques [Chhatbar and Francis, 2013]. Depending on different assumptions about how motor intent is encoded in the spiking activity of neurons, various decoding algorithms have been proposed. Widely used decoding algorithms include linear estimators which assume linear encoding [Georgopoulos et al., 1986, Salinas and Abbott, 1994], Wiener filters which enact linear decoding based on firing rate history [Carmena et al., 2003, Ethier et al., 2012, Hatsopoulos et al., 2004, Hochberg et al., 2006, Serruya et al., 2002, Wessberg et al., 2000], Kalman filters which augment linear estimators with a smoothly evolving prior [Koyama et al., 2010a, Li et al., 2009, Wu et al., 2006], point process methods which model spikes as a binary time series [Brown et al., 1998, 2002, Truccolo et al., 2005] and other varieties of decoders [Olson et al., 2005, Sanchez et al., 2004]. The differences in these algorithms will be discussed in detail in chapters 2 and 3.

The third component is an external device that is in turn actuated or controlled by commands decoded from neural activity, or displays the categorical decoding outputs. Such a device could be a cursor on a computer screen [Carmena et al., 2003, Gilja et al., 2012, Serruya et al., 2002, Taylor et al., 2002], a robotic arm [Chapin et al., 1999, Velliste et al., 2008], an electrical stimulator that can animate paralyzed muscle tissue [Bouton et al., 2016, Ethier et al., 2012, Moritz et al., 2008], or a spelling device which the user can select from among potentially dozens of target characters [Chen et al., 2015]. In this thesis, I will consider the case where the decoded output commands control a computer cursor or a robot arm.

The fourth component is the sensory system where the subject can get feedback about the movement of the external device. Although recent research efforts are exploring the benefits of stimulation in somatosensory cortex to restore ongoing proprioceptive or tactile sensation [Flesher et al., 2016, Kim et al., 2015, O'Doherty et al., 2011], the most prominent source of this feedback so far has been visual [Hatsopoulos and Donoghue, 2009, Lebedev and Nicolelis, 2006, Linderman et al., 2008]. Therefore, here I will focus in this work on devices that rely on visual feedback.

1.3 The decoding problem

The decoding problem focuses on inferring the subject's intended movement from recorded neural activity and is a critical roadblock to the clinical deployment of BCI systems. There are several factors that complicate the decoding problem.

It is unclear how volitional movements are represented in neurons.

Volitional movement is the result of cognitive processes which lead to the exertion of some action on the world. The goal of decoding is to estimate this volitional movement and use it to drive an external device. However, the process of movement generation is not clearly defined or understood. Without concrete, rigorous models to describe how volitional movements are represented in neurons, it is difficult to develop a robust decoder to infer volitional movements from neural activity.

There has been a long debate about whether neurons in M1 represent volitional movements by their kinematics (e.g., the time-evolving position and velocity of the arm end-point) or their kinetics (e.g., the forces and torques around the individual joints). Since the pioneering experiments of Evarts [1968], many studies have demonstrated correlations between neuronal activity and the output forces and torques [Georgopoulos et al., 1992, Hepp-Reymond et al., 1978, Humphrey et al., 1970] and several groups have shown that muscle or force related signals can be predicted from the activity of simultaneously recorded neurons [Carmena et al., 2003, Ethier et al., 2012, Fagg et al., 2009, Gupta and Ashe, 2009, Humphrey et al., 1970, Kim et al., 2007]. These results suggest that M1 neurons may represent the movement's kinetics and encode at the muscle level which is near the final stage of the motor control. In contrast, other studies have described relationships between M1 activity and various kinematic parameters of motor output such as the direction and distance of targets relative to the hand, and the direction, speed, and spatial path of hand displacement [Ashe and Georgopoulos, 1994, Caminiti et al., 1990, 1991, Fu et al., 1993, 1995, Georgopoulos et al., 1982, 1988, Kalaska et al., 1989, Moran and Schwartz, 1999a,b, Reina et al., 2001, Schwartz, 1992, 1993, 1994, Schwartz and Moran, 1999, Schwartz et al., 1988, 2004]. These findings suggest that M1 neurons may represent the movement's kinematics and encode motor intent at a earlier stage of the motor control hierarchy. Currently, most clinical BCIs focus on decoding kinematic control signals from the M1 activity, primarily because these are relatively easy to transform into cursor movements on a computer screen or endpoint movements of a robotic arm. Notable exceptions include work to decode desired muscle activity to re-animate a paralyzed limb, as done in [Ethier et al., 2012].

Beside the lack of knowledge about how volitional movements are represented in neurons, ground truth knowledge of the volitional movement itself is hard to obtain. For a healthy user who is capable of making overt movements (as in a laboratory setup with non-human primates [Carmena et al., 2003, Gilja et al., 2012, Serruya et al., 2002, Taylor et al., 2002]), it is possible to observe resulting physical movements and use them as an approximation of volitional movements in the calibration session. However, in many cases of interest the user is not able to make overt movements, and in those cases, volitional movements are usually approximated by movements the user observes or imagines [Hochberg et al., 2006, Kim et al., 2008, Suminski et al., 2010, Velliste et al., 2008, Wahnoun et al., 2006]. In the absence of well-approximated volitional movements, it is difficult to calibrate a decoding model to translate neural activities into desired motor commands.

The representation of intent is task and context specific.

Beside the movement itself, the neural representation can also depend on the task and the context the movement is associated with. It has been observed that even with the same movement, under different tasks or different contexts, the neural representation could be different. For example, researches have demonstrated that motor neuronal activity and static force show a linear relationship over a restricted force range. In an early study conducted by Humphrey et al. [1970], the coefficient value of the linear regression model varies with the range of forces in the task. Hepp-Reymond et al. [1999] later explored this phenomenon in greater detail and demonstrated that motor cortical neurons encoding force during an isometric pinching task undergo dynamic range adaptation with changes in the task context and cued expectation of the task context. For the kinematics case, in the classical model, the firing rates linearly modulate between the minimum and maximum rates with the cosine of the angle between the movement direction and preferred direction [Georgopoulos et al., 1982, Schwartz et al., 1988]. This tuning model has been demonstrated for movements in both 2D and 3D arm reaches. Similar to the kinetics case, Rasmussen et al. [2017] found that this linear model between the neural representation and the movement direction is also context specific. In their experiment, they trained two rhesus macaque monkeys to use their brain activity to move a cursor on a virtual reality screen in either 2D or 3D. Studying this brain activity showed that the modulation depth of the linear model shrinks when the movement context switch from 2D to 3D. That means that neurons are less sensitive to the cursor's direction of movement when the movement is in a 3D space than in a 2D space.

Other aspects of motor encoding have been found to be context specific. In a BCI clinical trial, Downey and colleagues noted that the neural representation of a grasping movement changed depending on whether the subject was attempting to grasp an actual object or empty space [Downey et al., 2017]. Work from the Francis lab has further demonstrated the existence of a reward signal within M1 that can distort the neural representation of a movement depending upon whether that movement is conducted in a rewarding environment or not [Marsh et al., 2015, McNiel et al., 2016]. If unaccounted for, these contextual factors complicate the decoding of motor intent signals.

Learning makes decoding a moving target.

Another aspect which makes the decoding problem difficult is subject learning. Current BCI systems are far from perfect, and thus learning is quite critical for the subject to control the closed-loop BCI system effectively [Orsborn and Pesaran, 2017, Shenoy and Carmena, 2014]. Every time the resulting movement of the artificial limbs deviates from the subject's intended movement, feedback about the error gives the subject a way to estimate the performance and update his/her knowledge about the system. This trial-and-error process often enables a subject to eventually control an imperfect BCI system through learning.

Studies have indicated that the process of learning is associated with changes in the directional tuning properties of neurons [Carmena et al., 2003, Jarosiewicz et al., 2008, Li et al., 2001, Mandelblat-Cerf et al., 2009, Paz and Vaadia, 2004, Taylor et al., 2002]. Although these learning-related changes improve the subject's performance, they impose challenges for the design of BCI decoders because the decoding parameters calibrated at the beginning of learning may no longer be optimal, or even appropriate, after the subject becomes familiar with the system. While some researchers have modeled the closed-loop BCI learning process [Héliot et al., 2010, Legenstein et al., 2010, Zhang et al., 2012], the learning process is still probably the least understood aspect of the sensorimotor control loop.

1.3.1 Optimal BCI decoder design

The goal of this thesis is to study the problem of how to design an optimal BCI decoder. Before we dive deep into this problem, we first need to define what is the "optimality" of a BCI decoder. It is intriguing to define the "optimal" BCI system as the one that allows an individual to control the system as naturally as his or her own arm. Ideally, once this optimal decoder is calibrated, it will never need to be re-calibrated again, and during the entire lifespan of the device, the individual would not feel any difference between controlling the BCI system and controlling his or her own arm. Currently, most BCI decoding algorithms are designed from this *biomimetic*, or *estimation* standpoint, in which neurons are assumed to represent upcoming intended movements and recorded firing rates are noisy observations of that underlying intent. Under this framework, BCI design is properly treated as a signal estimation problem, where the goal is to move the prosthesis to achieve as close a match to the decoded intent as possible. A BCI designed in this fashion will tend to be biomimetic, since it leverages the endogenous tuning to motor intent.

If tuning to motor intent were static, BCIs designed from an estimation standpoint would be statistically optimal provided the assumptions made about the form of intention tuning were correct. In fact, the differences between various decoding algorithms tend to stem from differences in the assumptions they make about neural tuning [Zhang and Chase, 2015]. However, tuning to motor intent is not static: it is well known that tuning curves change during learning in a manner that reflects increased task performance [Chase et al., 2012, Gandolfo et al., 2000, Ganguly et al., 2011, Jarosiewicz et al., 2008, Mandelblat-Cerf et al., 2011, Paz et al., 2005, Richardson et al., 2012, Wise et al., 1998]. Given that tuning curves can change, it is no longer guaranteed that a decoder designed for one set of neural tunings would be better than a decoder designed for a different set that could be learned.

The second definition of "optimality" is from a *control systems* perspective. Each prosthetic effector, in conjunction with its decoding algorithm, acts as a control system that the subject needs to learn how to use through trial-and-error. Robotic arms are, by definition, physical control systems, while brain-controlled computer cursors may or may not be programmed to follow physical laws. When interpreted this way, the “optimality” is defined as the *optimal cost* derived from the subject’s trying to control a BCI system to generate the intended movement. In this thesis, we take the control systems perspective. Instead of trying to mimic a subject’s biological decoding process to minimize training time, we assume that the subject will learn to use whatever mapping we give him, and develop an approach to determine which mapping will enable the most proficient control after learning has occurred.

Once we have a way to design an endpoint BCI system by optimizing the *post-learning* performance, the next question is how can this system be learnt by the subject or how can we assist the subject’s learning. Although subjects have shown the capability of controlling a imperfect BCI system with computer assistance [Taylor et al., 2002, Velliste et al., 2008], the learning process is probably the least understood aspect of the control loop and can take a very long time. An effective training schedule should manipulate the difficulty of the task to provide enough information to guide improvement without overwhelming the subject. In this thesis we pursue a first step towards solving that problem. By introducing a Bayesian framework for modeling a closed loop BCI learning process, we develop an adaptive algorithm to find the optimal training schedule for the performance improvement.

1.4 Outline of this thesis

In this thesis, I will first review some BCI fundamentals and cover current approaches to the decoding problem in chapter 2. Then, in chapter 3, I will motivate a different approach to the decoding problem from the standpoint of control theory. Under the control system perspective, I will detail a method for optimal BCI design that takes learning into account, and apply it to the special case of designing an optimal 2nd order physical control BCI system in chapter 4. Next, in chapter 5, I will outline one approach to modeling the learning process itself. Finally, in chapter 6, I will conclude with thoughts on what future directions may be taken with this line of work.

The thesis concludes with several addenda. In addendum A, I include some work I did developing a dual Kalman filter decoder to mitigate decoder instabilities. In addendum B, I provide detailed derivation of those algorithms proposed in chapter 4.

Chapter 2

Background

The input signal of a BCI system is the neural activity observed from the recording device. For intracortical microelectrodes, the collected neural signal is a time series of spikes generated based on the neuron's *underlying firing rate* λ. Different decoding algorithms formulate the spike train in different ways. The most widely used representation is the *recorded firing rates* y, which is computed as the number of spikes within a small bin divided by the bin size (typically of 50-100 ms), and is treated as a noisy observation of the underlying firing rate [Carmena et al., 2003, Collinger et al., 2013, Ganguly and Carmena, 2009, Gilja et al., 2012, Hochberg et al., 2006, 2012, Kim et al., 2008, Li et al., 2011, Mulliken et al., 2008, Orsborn et al., 2014, Serruya et al., 2002, Taylor et al., 2002, Velliste et al., 2008, Willett et al., 2013].

In this chapter, I briefly review some common approaches to the BCI decoding problem, and introduce some mathematical notation that will be used throughout the rest of the thesis. Notation used in this chapter is listed in Table 2.1.

Table 2.1: **Notations.**

$\lambda_{i,t}$	the instantaneous firing rate of neuron i at time t
$s_{i,t}$	the number of spikes of neuron i within the time interval $[t, t+\Delta]$
$y_{i,t}$	the observed firing rate of neuron i computed on the time interval $[t, t+\Delta]$
$\boldsymbol{y}_t$	the observed firing rates of a population of n neurons at time t
$\boldsymbol{d}_i$	the preferred direction of neuron i
m_i	the modulation depth of neuron i
$b_{0,i}$	the base firing rate of neuron i
$\boldsymbol{x}^*(\boldsymbol{p}^*, \boldsymbol{v}^*, \ldots)$	the subject's intended movement (position, velocity, ...)
$\hat{\boldsymbol{x}}\ (\hat{\boldsymbol{p}}, \hat{\boldsymbol{v}}, \ldots)$	the estimated movement (position, velocity, ...)
$\boldsymbol{x}\ (\boldsymbol{p}, \boldsymbol{v}, \ldots)$	the implemented movement (position, velocity, ...)

2.1 Population vector algorithm

In a groundbreaking experiment by Georgopoulos et al. [1982], it was found that many neurons in M1 could be well described as having a "cosine tuning" property to intended movement direction

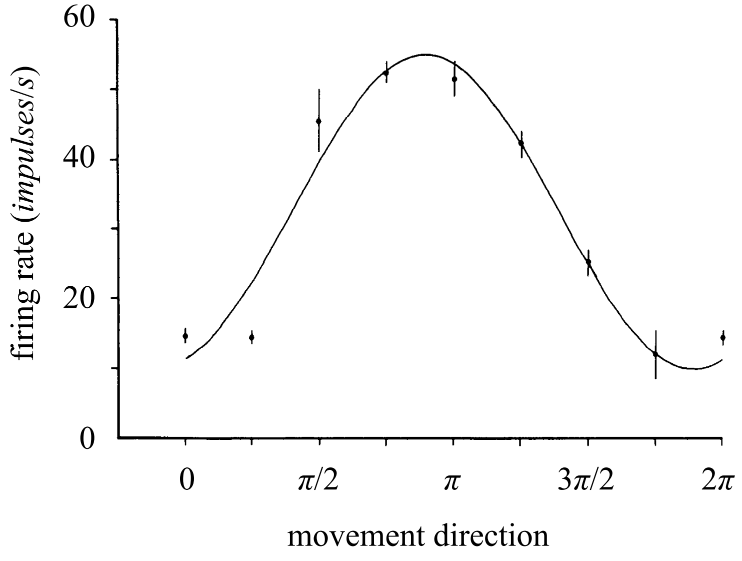


Figure 2.1: **Neural activity, measured in *impulses/s*, as a function of the direction of movement and the fitted cosine curve.** This figure is adapted from [Georgopoulos et al., 1982].

(Fig. 2.1). Later, Schwartz et al. [1988] found that the cosine tuning could be generalized to 3D movements. The original formulation in [Georgopoulos et al., 1982] had firing rate as a function of direction, not velocity. It wasn't until [Moran and Schwartz, 1999a] that they looked at modulation of the tuning curve as a function of movement speed. Based on this finding, the i-th neuron's underlying firing rate at time t, $\lambda_{i,t}$, is presumed to have a linear relationship with the subject's intended movement as

$$\lambda_{i,t} = b_{0,i} + m_i \boldsymbol{d}_i^T \boldsymbol{v}_t^*, \tag{2.1}$$

where $\boldsymbol{d}_i$, m_i and $b_{0,i}$ are the neuron's the preferred direction, modulation depth and the base firing rate respectively and T denotes the transpose operation. Based on this encoding model, Georgopoulos and colleagues introduced the population vector algorithm (PVA) in 1986 [Georgopoulos et al., 1986]. The basic idea is that each neuron "pushes" the cursor along its preferred direction, with the amount of the push being proportional to its firing rate. The final movement is determined by the aggregation of all the neurons' contributions and gets implemented in the BCI system. By using the recorded firing rate $y_{i,t}$ as an estimate of the underly firing rate $\lambda_{i,t}$, the PVA decoder is written as

$$\hat{\boldsymbol{v}}_t = \frac{k_s}{n} \sum_{i=1}^{n} \frac{y_{i,t} - b_{0,i}}{m_i} \boldsymbol{d}_i, \tag{2.2}$$

where n is the number of recorded neurons and k_s is a constant that scales the unitless decoded direction into a velocity [Chase et al., 2012].

Denoting the vector $\boldsymbol{y}_t = (y_{1,t}, y_{2,t}, \ldots, y_{n,t})^T$ as the recorded firing rates of a population of n neurons at time t, we can rewrite Eqn. 2.2 in matrix form by gathering the preferred directions into a single matrix D of size $d \times n$. Each column of D corresponds to the preferred direction of a single neuron. Further gathering the modulation depths m_i into a single diagonal matrix Λ_m, we have

$$\hat{\boldsymbol{v}}_t = (k_s/n) D \Lambda_m^{-1} \boldsymbol{y}_t. \tag{2.3}$$

where we assume the recorded firing rate $y_{i,t}$ is centralized and thus ignore $b_{0,i}$ from Eqn. 2.2. In the following parts of this thesis, without further specification, we assume the firing rate is alway centralized and ignore the base firing rate for simplicity.

2.2 Optimal linear estimator

The PVA is a biologically-inspired algorithm. However, in practice it has been shown to return biased estimates of motor intent when the preferred directions of the recorded neurons are not uniformly distributed [Chase et al., 2009, Kass et al., 2005, Salinas and Abbott, 1994]. To compensate for this bias, the *optimal linear estimator* (OLE) has been proposed. As the name implies, the OLE computes the optimal linear estimate according to square errors. Notice that Eqn. 2.1 can be generalized as

$$\boldsymbol{\lambda}_t = C \boldsymbol{v}_t^*. \tag{2.4}$$

where C is the coefficient matrix. Compare with Eqn. 2.1, we can see that if denote $\boldsymbol{c}_i$ as the i-th row of C, the neuron's modulation depth $m_i = \|\boldsymbol{c}_i\|$ and the neuron's preferred direction $\boldsymbol{d}_i = \boldsymbol{c}_i^T / \|\boldsymbol{c}_i\|$.

In order to get the optimal estimate of intended velocity, we need to make some assumption about the relationship between the recorded firing rates $\boldsymbol{y}_t$ and the underlying firing rates $\boldsymbol{\lambda}_t$. One widely used assumption is that is that $\boldsymbol{y}_t$ is a noisy version of $\boldsymbol{\lambda}_t$ and the noise is Gaussian,

$$\boldsymbol{y}_t = \boldsymbol{\lambda}_t + \boldsymbol{\varepsilon}_t, \tag{2.5}$$

where $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(\mathbf{0}, \Sigma_t)$ is zero mean Gaussian noise. Although the Gaussian distribution has some good mathematical properties, one major drawback making it not very proper for real neural applications is that the noise is independent of the signal, i.e, Σ_t is independent of $\boldsymbol{\lambda}_t$. Experimental results from real applications show that the larger the signal is, the larger the noise associated with the signal is [Churchland et al., 2006, Tolhurst et al., 1983]. In order to mimic this kind of behavior while preserving computational simplicity, signal-dependent Gaussian noise is usually adopted where the noise's covariance matrix Σ_t is a function the signal $\boldsymbol{\lambda}_t$. We will further discuss the signal-dependent noise and how it affects decoding in chapter 4.

Then, by minimizing the square error, the optimal estimated velocity of OLE decoder is given as

$$\hat{\boldsymbol{v}}_t = (C^T \Sigma_t^{-1} C)^{-1} C^T \Sigma_t^{-1} \boldsymbol{y}_t. \tag{2.6}$$

We can see that if recorded neurons are uncorrelated with each other and have a constant variance for all i and t, then $\Sigma_t \propto I$. If we further assume that the preferred directions of recorded neurons are uniformly distributed and modulation depths are the same, then $C^T C \propto I$ and the OLE is equivalent to the PVA. Experimental results from [Chase et al., 2009, Kass et al., 2005, Salinas and Abbott, 1994] have shown that the OLE overcomes the intrinsic bias in the PVA decoder.

2.3 Weiner filter

A problem with the state-less estimators like PVA and OLE is smoothing: when implemented on small time bins Δ (30ms is typical for real-time decoding), the movement estimates can be quite noisy unless there are a large number of cells. With limited number of recorded neurons, filtering is used in various decoding algorithms to handle the smoothing problem [Hochberg et al., 2006, Koyama et al., 2010b, Serruya et al., 2002, Wessberg et al., 2000]. The basic idea of filtering is that, instead of just using the instantaneous firing rate, the decoding algorithm also takes a history of the firing rates into consideration. One of the widely used filtering methods for BCI decoding is the Wiener filter [Carmena et al., 2003, Hatsopoulos et al., 2004, Hochberg et al., 2006, Serruya et al., 2002, Wessberg et al., 2000], which directly assumes a linear model of the decoded movement on the current firing rates and the historical firing rates. Discretizing the time into a series of time stamps, with Δ as the interval, the decoded movement is

$$\hat{\boldsymbol{x}}_t = \sum\nolimits_{u=\tau}^{0} M_u \boldsymbol{y}_{t-u} + \boldsymbol{c} + \boldsymbol{w}_t, \tag{2.7}$$

where $\hat{\boldsymbol{x}}_t$ is the estimated kinematics (e.g., position, velocity or gripping force) at time t, $\boldsymbol{y}_{t-u}$ is the neurons' recorded firing rates at time $t-u$, τ is the length of history we look back, M_u is the weights for the time lag u, $\boldsymbol{c}$ is some constant, and $\boldsymbol{w}_t$ is zero mean Gaussian noise.

The Wiener filter essentially assumes a linear mapping from the history of recorded firing rates to the estimated kinematics. By replacing the linear mapping with more complex non-linear functions, it is straightforward to extend the Wiener filter to capture the non-linear relationship between the neural activities and the intended movement for BCI decoding. Then the decoder design problem becomes a machine learning problem where a prediction model is trained to predict the kinematics from the recorded neural signals and we can take advantages of those novel machine learning techniques. For example, in [Wessberg et al., 2000] the Wiener filter is replaced by an artificial neural network and Sanchez et al. [2002, 2003, 2004] further extended this work by using a recurrent neural network. Another example is the support vector machine (SVM) used in [Olson et al., 2005], where neural activity is represented by the recorded firing rates and the decoding problem was formalized as a binary classification problem and solved by the SVM.

2.4 Kalman filter

Possibly the most widely used decoding algorithm is the Kalman filter [Black et al., 2003, Li et al., 2009, Wu and Hatsopoulos, 2008, Wu et al., 2003, 2006]. As opposed to the Wiener filter, which smooths the estimation by directly building a model on the historical firing rates, the Kalman filter formulates a state-space model where the information about historical firing rates is embedded in the state evolution [Brockwell et al., 2004, Brown et al., 1998, Eden et al., 2004, Koyama et al., 2010b]. State space methods suppose two types of models for use in decoding. The first is an observation model that dictates how the recorded observations relate to the underlying latent state to be estimated. This is the same as the neural encoding model. The second is the state evolution model, which describes how the latent state evolves over time. In

BCI decoding, the state is the intended kinematics we are trying to estimate, for example, the intended velocity $\boldsymbol{v}_t^*$ at time t. For the purpose of obtaining a smooth velocity trajectory, we assume a state model that constrains the sequence of states $\boldsymbol{v}_t^*$ so that they are likely to evolve with some degree of smoothness from one time step to the next. The observation model provides the likelihood of the current observation and the state model serves as the prior probability. The maximum a posteriori (MAP) estimate of the state is obtained by maximizing the updated posterior probability.

Here we first consider the linear case. The state evolution, which serves as a smooth prior for the intended kinematics, is given as

$$\boldsymbol{x}_t^* = A\boldsymbol{x}_{t-1}^* + \boldsymbol{\omega}_t, \tag{2.8}$$

where $\boldsymbol{\omega}_t \sim \mathcal{N}(\mathbf{0}, R)$ is assumed to be zero mean Gaussian noise. The linear observation equation, similar to Eqn. 2.4, is written as

$$\boldsymbol{y}_t = C\boldsymbol{x}_t^* + \boldsymbol{\varepsilon}_t, \tag{2.9}$$

where $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(\mathbf{0}, Q)$ is also zero mean Gaussian noise.

The state-space method is essentially a Bayesian inference method where the goal is to estimate the a posteriori probability of the intended kinematics when given the recorded spike counts. Under the linear and Gaussian assumption, the Kalman filter [Kalman, 1960], first introduced for BCI in [Wu et al., 2003], provides an efficient recursive algorithm to compute the posterior probability $p(\boldsymbol{x}_t^*|\boldsymbol{y}_{1,\dots,t})$. The optimal estimate $\hat{\boldsymbol{x}}_t$ of the intended kinematics $\boldsymbol{x}_t^*$ is the mean of this distribution, which can be estimated through the following set of recursive equations:

$$K_t = (A\hat{\Sigma}_{t-1}A^T + R)C^T(C(A\hat{\Sigma}_{t-1}A^T + R)C^T + Q)^{-1}, \tag{2.10}$$

$$\hat{\boldsymbol{x}}_t = (A - K_tCA)\hat{\boldsymbol{x}}_{t-1} + K_t\boldsymbol{y}_t, \tag{2.11}$$

$$\hat{\Sigma}_t = (I - K_tC)(A\hat{\Sigma}_{t-1}A^T + R). \tag{2.12}$$

Here K_t is the Kalman gain which computes the optimal mixing between reliance on information from the state evolution model and information from the observation model according to the noise assumed to be in each. $\hat{\Sigma}_t$ is the estimate of the covariance of $p(\boldsymbol{x}_t^*|\boldsymbol{y}_{1,\dots,t})$. To use the Kalman filter, an initial covariance matrix $\hat{\Sigma}_0$ and kinematics $\hat{\boldsymbol{x}}_0$ must be given. In practice, it is common to center the prosthesis so all terms in $\hat{\boldsymbol{x}}_0 = \mathbf{0}$ with certainty $\hat{\Sigma}_0 = 0$. The Kalman gain is time dependent. However, it tends to converge to a stable value within a few timesteps, independent of the observations $\boldsymbol{y}$ [Chui and Chen, 2009, Malik et al., 2011]. Another common practice is to initialize the system with $\hat{\boldsymbol{x}}_0$ and $\hat{\Sigma}_0$ as the matrix that ensures that K_t is stable [Dethier et al., 2011, Sadtler et al., 2014].

In different implementations of Kalman filter, the kinematics $\boldsymbol{x}_t^*$ can be just the velocity, i.e., $\boldsymbol{x}_t^* = \boldsymbol{v}_t^*$, which is called the velocity Kalman filter (VKF) [Hochberg et al., 2012, Kim et al., 2008] or the kinematics an be the concatenation of the position and the velocity, i.e., $\boldsymbol{x}_t^* = (\boldsymbol{p}_t^*; \boldsymbol{v}_t^*)$, which is called the position-velocity Kalman filter (PVKF) [Gilja et al., 2012, Gowda et al., 2014, Homer et al., 2013, Wu et al., 2006]. We will discuss the VKF and PVKF in more detail in chapter 3.

The Kalman filter is also a linear model, and various non-linear extensions have been proposed. Li et al. [2009] proposed an k-th order unscented Kalman filter (UKF) which uses of a non-linear (quadratic) model of neural tuning. In addition to allowing a non-linear relationship between neuronal firing rates and the hand's position and velocity, UKF also augments the movement state variables with a history of $k - 1$ recent states, which improves prediction of the desired command even before incorporating neural activity information and allows the tuning model to capture relationships between neural activity and movement at multiple time offsets simultaneously.

2.5 Point process filter

All the decoders reviewed in the previous sections use the recorded firing rate computed on a small time bin as the estimate of the underlying firing rate. However, selecting the time bin size Δ can be quite challenging. If Δ is too small, the computed firing rate can be quite noisy, and if Δ is too large, the underlying firing rate can change during the time interval and the computed firing rate may not represent the underlying firing rate very well. Another representation of the neuron's spike train does not compute the recorded firing rate explicitly but instead uses a time series of 0 and 1 to indicate the absence or presence of a spike within a certain time interval [Brown et al., 1998, 2002, Eden et al., 2004, Kass and Ventura, 2001, Koyama et al., 2010b, Truccolo et al., 2005]. Similar to the recorded firing rate, continuous time is also divided into consecutive bins, however, the bin size is much smaller, typically 1-5 ms. Such small bin size makes each interval contains at most one spike and the resulting time series will be a binary sequence. This binary time series can then be modeled as a Poisson point process. Denoting the number of spikes of neuron i within the time interval $[t, t + \Delta]$ by $s_{i,t} (s_{i,t} \in \{0, 1\})$, we have the probability of observing a spike from Poisson distribution is

$$P(s_{i,t}) = (\lambda_{i,t}\Delta)^{s_{i,t}} e^{-\lambda_{i,t}\Delta}. \tag{2.13}$$

Under this representation, instead of using the linear model in Eqn. 2.1, the log-linear model [McCullagh and Nelder, 1989] is often used as

$$\lambda_{i,t} = \exp\left(b_{0,i} + m_i \boldsymbol{d}_i^T \boldsymbol{v}_t^*\right). \tag{2.14}$$

Beside the advantage that the log-linear model guarantes the firing rate to be always positive, it also agrees with the von Mises tuning function which allows for a narrower tuning curve than the cosine tuning function, as has been found to hold true for most neurons in M1 [Amirikian and Georgopulos, 2000].

Given the observation model Eqn. 2.13 and assuming Gaussian state evolution model, we can also derive the recursion equation to compute the posterior probability $p(\boldsymbol{x}_t^* | \boldsymbol{s}_{1,\ldots,t})$. However, there is no closed-form solution for the point process decoder. One approach to get the posterior distribution is the particle filter, which is based on Monte Carlo simulation [Brockwell et al., 2004, Doucet et al., 2000, Ergün et al., 2007, Kitagawa, 1996]. The particle filter draws "particles" from the prior distribution and then, by applying the observation model, obtains particles corresponding to the posterior distribution that can be averaged to approximate the maximum likelihood estimate of motor intent. This method turns the recursive equations for the filtering distribution into a stochastic dynamical system of interacting particles, each representing one draw from that posterior.

While particle filtering has proven itself to be useful in practice [Brockwell et al., 2004, Doucet et al., 2000, Ergün et al., 2007], like any Monte Carlo scheme it can be computationally costly and the number of particles needed for a given accuracy grows rapidly with the dimensionality of the state space. For real-time processing, such as in BCI, the computational cost of effective particle filtering can quickly become prohibitive. Another approach to get the posterior distribution is the Laplace Gaussian filter, which uses a Laplace-type approximation and avoids the computational heavy Monte Carlo simulation [Brown et al., 1998, Eden et al., 2004, Koyama et al., 2010b, Shanechi et al., 2013, 2016, Truccolo et al., 2005]. This method is a deterministic approximation based on sequential application of Laplace's method to obtain estimates of the mean and variance of the posterior density, and then approximates that density by a Gaussian with that mean and variance. For real-time neural decoding, the Laplace Gaussian filter is shown to be more accurate than the particle filter under the same computational cost [Koyama et al., 2010b].

2.6 Summary

In this chapter, I have briefly reviewed some common approaches to the BCI decoding problem. Among them, the PVA and OLE assume a linear encoding model and derive the decoding model from it. The Kalman filter further assumes linear system dynamics which can make the decoded movement more smooth and accurate. The Wiener filter does not explicitly assume an encoding model, instead, it assumes a linear decoding model depending on the history of firing rates. The point process filter assumes a log-linear encoding model with Poisson distribution, which can mimic the neuron behavior more accurate than linear Gaussian model but also introduce more computational burden. In the following thesis, I will focus on linear Gaussian encoding models, since they are the most commonly used.

Chapter 3

BCI performance from a physical control systems perspective

In this chapter I review the performance of different BCI decoding algorithms and attempt to explain differences in performance based on the physical control system that each decoder represents. This work has been published as [Zhang and Chase, 2015].

3.1 Introduction

As discussed in the previous chapter, a BCI decoding algorithm specifies how recorded signals (like recordings from intracortical multielectrode arrays) get translated into movement of the prosthesis. Currently, nearly all BCI decoding algorithms are designed from an *estimation standpoint*: it is assumed that neurons are tuned to different intended movements, and recorded firing rates are treated as noisy observations of that underlying motor intent. Thus, differences in BCI decoding algorithms are typically interpreted to result from the different assumptions that they make about how neurons represent motor intent. Algorithms that fall into this class include linear estimators such as the PVA and the OLE, state-space models such as the Kalman filter, Laplace-Gaussian filter [Koyama et al., 2010a,b], and the unscented Kalman filter [Li et al., 2009], as well as a host of related variants (e.g., particle filter [Brockwell et al., 2004], steady-state Kalman filter [Malik et al., 2011], variational Bayesian regression [Li et al., 2011], recalibrated feedback intention-trained Kalman filter [Gilja et al., 2012], parameter tracker [Zhang and Chase, 2013], speed-dampening Kalman filter [Golub et al., 2014], and others). Differences in the performance of various algorithms are typically assumed to relate to how well the algorithms' assumptions about motor intent encoding match the true underlying encoding of the neurons.

However, another way to interpret BCI system design is from a *control system* perspective. Each prosthetic effector, in conjunction with its decoding algorithm, acts as a control system that the subject needs to learn how to use through trial-and-error. Robotic arms are, by definition, physical control systems, while brain-controlled computer cursors may or may not be programmed to follow physical laws. When interpreted this way, differences in the performance of two algorithms might instead stem from differences in the control systems themselves: some control systems may be more usable than others, due to their physical characteristics or ease of

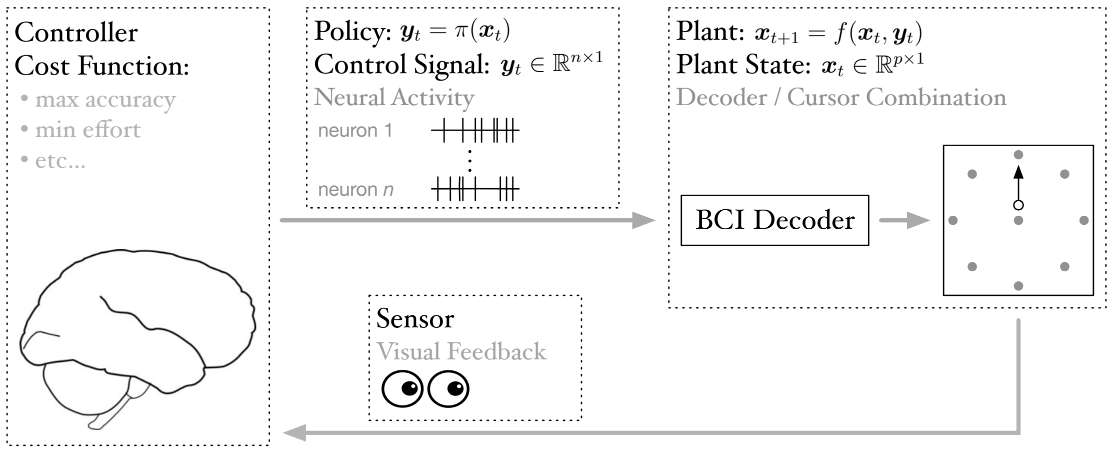

Controller
Cost Function:
• max accuracy
• min effort
• etc...
Policy: $\boldsymbol{y}_t = \pi(\boldsymbol{x}_t)$
Control Signal: $\boldsymbol{y}_t \in \mathbb{R}^{n\times 1}$
Neural Activity
neuron 1
neuron n
Plant: $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{y}_t)$
Plant State: $\boldsymbol{x}_t \in \mathbb{R}^{p\times 1}$
Decoder / Cursor Combination
BCI Decoder
Sensor
Visual Feedback

Figure 3.1: **Schematic of a BCI as a feedback control system.** The major components of a feedback control system (namely, the controller, control signals, plant, and feedback) are laid out on top of a typical BCI cursor control schematic, where the brain is identified as the controller, the control signals are neural activity (often tapped out of primary motor cortex), the plant is the combination of the BCI decoder and the cursor, and feedback is accomplished by watching the cursor movements.

conceptualization. Here we first introduce linear physical systems with control signals and derive the corresponding physical systems for some common BCI cursor decoding algorithms. We then re-interpret findings from the literature on which decoding algorithms work best in light of the physical systems that they represent. Intriguingly, when interpreted in this way, the literature suggests that BCI systems that follow physical laws are more usable than those that do not. Further, in on-line control it appears that BCI systems that reduce to equivalent physical forms tend to be equally-well controlled. These results have implications not only for BCI design, but may also shed light on the brain's ability to conceptualize motor effectors of varying forms.

3.2 Linear physical systems with control signals

Before proposing our view of BCI design from a control system perspective, we first look at a general control system, as shown in Fig. 3.1. In this system, control signals generated by the controller (the brain) are used to drive the plant (the decoder and cursor). Feedback about the system is observed by the sensors (the eyes) and used by the controller to generate new control signals. If it is a physical system, the system dynamics, which dictate the plant's evolution as driven by the control signal, should obey physical laws. Formally, the dynamics for a control system can be represented as

$$d\boldsymbol{x}/dt = f(\boldsymbol{x}, \boldsymbol{u}, t), \tag{3.1}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ is the plant's state in d dimensional space, $\boldsymbol{u} \in \mathbb{R}^n$ is the control signal and t is the current time. To make the control system a legitimate physical system, f should obey physical laws, i.e., velocity should be the derivative of position, acceleration should be the derivative of velocity, etc.

For computational simplicity, a linear discretized approximation of Eqn. 3.1 is typically used in practice. To do that, continuous time is discretized into a series of steps with the step size as a small time interval Δ ($\Delta = 20$ms for example), and the system dynamics at the t-th time step is expressed as

$$\boldsymbol{x}_{t+1} = H_t \boldsymbol{x}_t + M_t \boldsymbol{u}_t. \tag{3.2}$$

Here, $H_t \in \mathbb{R}^{d \times d}$ is a matrix that dictates the internal physics of the control system, and $M_t \in \mathbb{R}^{d \times n}$ is a matrix that dictates how the control system responds to its control inputs.

If we are studying a physical system driving a prosthetic effector's movement, then $\boldsymbol{x}$ is often the kinematics of the prosthetic effector, which could include the position, $\boldsymbol{p}$, the velocity, $\boldsymbol{v}$, and potentially higher-order terms as well. For example, the 1st order linear discretized approximation, where the state is the position $\boldsymbol{x}_t = \boldsymbol{p}_t$, has the form

$$\boldsymbol{p}_{t+1} = H_t \boldsymbol{p}_t + M_t \boldsymbol{u}_t. \tag{3.3}$$

In a 2nd order linear system, the state $\boldsymbol{x}_t$ will contain both position and velocity terms. To ensure that position is the integral of velocity, we need to have the following relationship:

$$\begin{pmatrix} \boldsymbol{p}_{t+1} \\ \boldsymbol{v}_{t+1} \end{pmatrix} = \begin{pmatrix} I & \Delta I \\ H_{p,t} & H_{v,t} \end{pmatrix} \begin{pmatrix} \boldsymbol{p}_t \\ \boldsymbol{v}_t \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ M_{v,t} \end{pmatrix} \boldsymbol{u}_t. \tag{3.4}$$

Note that the control signal affects the next velocity, only.

To gain some insight into the meaning of these control system parameters, consider a simple 2nd order physical system where the movement is on 1 dimension (Fig. 3.2). This system consists of a point mass, m_p, attached to a viscous damper and an elastic spring connected in parallel. The forces operating on the point mass come from external sources, F_{ex}, as well as internal from the spring and damper. The force from the spring is $F_s = -kp$ (where k is the spring constant and p is the position of the point mass, measured relative to the equillibrium position of the spring), and the force from the damper is $F_d = -\eta v$ (where η is the damping coefficient and v is the velocity of the point mass). Because these elements act in parallel, these two forces sum at the point mass, to create a total internal force F_{in}, where

$$F_{in} = F_d + F_s = -\eta v - kp. \tag{3.5}$$

Together with the external force the acceleration of the point mass may be derived from Newton's laws of motion:

$$a = F_{tot}/m_p = (F_{in} + F_{ex})/m_p = (-kp - \eta v + F_{ex})/m_p. \tag{3.6}$$

Discretized by a small time duration Δ, the updated velocity is given as

$$v_{t+1} = v_t + \Delta a_t = v_t + \Delta(-kp_t - \eta v_t + F_{ex,t})/m_p. \tag{3.7}$$

and the updated position is given as

$$p_{t+1} = p_t + \Delta v_t, \tag{3.8}$$

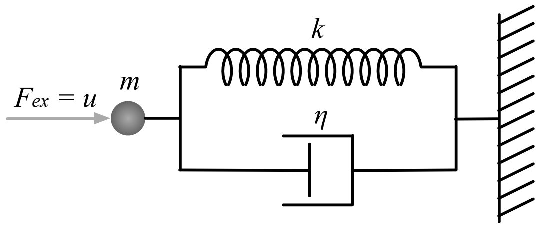


Figure 3.2: **A simple 2nd order physical control system.** Here, a point mass m is hooked up to a parallel combination of a spring (with spring constant k) and a damper (with damping coefficient η). This configuration corresponds to the well-known Kelvin-Voigt model from material science.

Putting this all together into matrix form, the discretized control system for the Kelvin-Voigt model in Fig. 3.2 is seen to be:

$$\begin{pmatrix} p_{t+1} \\ v_{t+1} \end{pmatrix} = \begin{pmatrix} 1 & \Delta \\ -\Delta k/m_p & 1 - \Delta\eta/m_p \end{pmatrix} \begin{pmatrix} p_t \\ v_t \end{pmatrix} + \begin{pmatrix} 0 \\ \Delta/m_p \end{pmatrix} F_{ex,t}. \tag{3.9}$$

Comparing Eqn. 3.9 with the general 2nd order physical control system presented in Eqn. 3.4 reveals that the external force $F_{ex,t}$ plays the role of the control signal $\boldsymbol{u}_t$, the spring-like elastic effects account for $H_{p,t}$, and the viscous damping effects account for $H_{v,t}$.

3.3 BCI systems from a physical control systems perspective

In this section we re-interpret some fairly common decoding algorithms, including the PVA, OLE and Kalman filter, in terms of the physical control systems to which they correspond. Fig. 3.1 casts the general BCI system into the control system framework. Here the motor cortex drives the BCI system by generating a proper set of neural activation patterns, which are the end result of a sequence of brain computations that take both visual feedback and goal information into account. Visual feedback of the prosthetic effector's movement is used to correct future movement. Therefore, BCI system design is essentially a control system design problem where one attempts to construct a BCI system that is as usable as possible. Formally, the dynamics of a BCI system can be represented as

$$d\boldsymbol{x}/dt = g(\boldsymbol{x}, \boldsymbol{y}, t). \tag{3.10}$$

where $\boldsymbol{y} \in \mathbb{R}^n$ is the recorded firing rates from a population of n neurons. The only difference between Eqn. 3.10 and Eqn. 3.1 is that the control signal $\boldsymbol{u}$ takes the form of neuronal firing rates $\boldsymbol{y}$.

When discussing the physical implementation of different BCI decoders, it will be necessary to distinguish the following three variables: motor intent, estimated motor intent, and implemented movement. Note that here and elsewhere in this document, we use the star notation to refer to intended movements (e.g., the intended velocity will be denoted as $\boldsymbol{v}^*$). Estimates of those intents will be indicated by an overhead carat (e.g., the estimated velocity will be denoted $\hat{\boldsymbol{v}}$). Finally, the movement that is implemented by the prosthesis will come without any notation (e.g., the implemented velocity will be denoted as $\boldsymbol{v}$). Please refer to Table 2.1 for all the notations.

3.3.1 Linear estimators

We start with one of the earliest decoding algorithms, the PVA [Georgopoulos et al., 1986]. The estimated velocity of the PVA is obtained by Eqn. 2.3. To implement the PVA estimate of velocity into an actual device, the velocity of the prosthesis is set equal to its estimate, i.e., $\boldsymbol{v}_t = \hat{\boldsymbol{v}}_t$. The implemented position is then set equal to the integral of these velocity commands, $\boldsymbol{p}_{t+1} = \boldsymbol{p}_t + \Delta \boldsymbol{v}_t$. Therefore, to represent the decoding from the physical system perspective, we have

$$\text{PVA physical system: } \boldsymbol{p}_{t+1} = \boldsymbol{p}_t + \Delta(k_s/n)D\Lambda_m^{-1}\boldsymbol{y}_t, \tag{3.11}$$

which is a special case of a 1st order linear physical control model (Eqn. 3.3), where $H_t = I$ and $B_t = \Delta(k_s/n)D\Lambda_m^{-1}$.

Similar to the PVA, the implemented velocity is also set equal to the estimated velocity in the decoding model of the OLE and the position of the prosthesis is derived by integrating velocity. Thus, given the estimated velocity obtained by Eqn. 2.6, the physical system corresponding to the OLE decoding is

$$\text{OLE physical system: } \boldsymbol{p}_{t+1} = \boldsymbol{p}_t + \Delta(C^T\Sigma_t^{-1}C)^{-1}C^T\Sigma_t^{-1}\boldsymbol{y}_t. \tag{3.12}$$

This is again another special case of a 1st order linear physical control system (Table 3.1).

Both the PVA and the OLE correspond to first-order physical control systems, albeit with slightly different mappings from neural firing to cursor movement. Experimental results from [Chase et al., 2009, Kass et al., 2005, Salinas and Abbott, 1994] have shown that the OLE overcomes an intrinsic estimation bias that the PVA decoder has when the preferred directions of recorded cells are not uniformly distributed. From an estimation standpoint, the OLE should be a better decoding algorithm than the PVA.

However, Chase and colleagues [2009] also demonstrated that the PVA and the OLE perform equivalently on-line: subjects were just as adept at controlling the PVA as they were at controlling the OLE, despite the fact that neural activity was mapped to different cursor movements under the two algorithms. One possible interpretation of this is that subjects learn the mapping from neural activity to decoder, be it a biased decoder like the PVA or an unbiased decoder like the OLE. The difference between the mapping from neural activity to cursor movement produced under these two decoders is akin to a visuomotor distortion, and visuomotor distortions are learned very quickly [Krakauer et al., 2000, Paz et al., 2005, Wu and Smith, 2013]. From this standpoint, the neurons are rapidly changing their activity to provide appropriate control signals to the device. Once learning is accomplished, both the OLE and the PVA give the subject a 1st order linear physical control system to control, and there appears to be no significant difference between the usability of these systems.

3.3.2 Linear state-space decoders

As discussed in chapter 2, a problem with the linear estimators described in the previous section is smoothing: when implemented on small time bins Δ, the movement estimates can be quite noisy. To compensate for this it is common to smooth the firing rate estimates [Koyama et al., 2010a]. Naturally, any smoothing also affects the physical system.

Linear state space models handle smoothing in a more elegant manner by applying a smooth prior to the evolution of the intended kinematics. In section 2.4 I have reviewed the well-known Kalman filter and its recursive equations (Eq .2.10-2.12) to obtain the estimated kinematics $\hat{\boldsymbol{x}}_t$. The relationship between the kinematics $\boldsymbol{x}_t$ that are actually implemented and the estimated kinematics $\hat{\boldsymbol{x}}_t$ depends on exactly how the Kalman filter is implemented. Here we introduce two popular implementations, the velocity Kalman filter (VKF) [Hochberg et al., 2012, Kim et al., 2008] and the position-velocity Kalman filter (PVKF) [Gowda et al., 2014, Homer et al., 2013, Wu et al., 2006].

Velocity Kalman filter

The VKF assumes that neurons are tuned to the intended velocity. Thus, the state evolution equation is

$$\boldsymbol{v}_t^* = A\boldsymbol{v}_{t-1}^* + \boldsymbol{\omega}_t \tag{3.13}$$

and the observation equation is

$$\boldsymbol{y}_t = C\boldsymbol{v}_t^* + \boldsymbol{\varepsilon}_t. \tag{3.14}$$

From Eqn. 2.11, we have the estimated intended velocity as

$$\hat{\boldsymbol{v}}_t = (A - K_tCA)\hat{\boldsymbol{v}}_{t-1} + K_t\boldsymbol{y}_t. \tag{3.15}$$

The implemented velocity $\boldsymbol{v}_t$ in this case is set equal to $\hat{\boldsymbol{v}}_t$ and the position is the integral of the velocity [Hochberg et al., 2012, Kim et al., 2008]. In matrix form, the implemented movement can be written as

$$\text{VKF physical system: } \begin{pmatrix}\boldsymbol{p}_t\\ \boldsymbol{v}_t\end{pmatrix} = \begin{pmatrix}I & \Delta I\\ \mathbf{0} & A - K_tCA\end{pmatrix}\begin{pmatrix}\boldsymbol{p}_{t-1}\\ \boldsymbol{v}_{t-1}\end{pmatrix} + \begin{pmatrix}\mathbf{0}\\ K_t\end{pmatrix}\boldsymbol{y}_t. \tag{3.16}$$

Comparing with Eqn. 3.4, we can see that the VKF is a 2nd order linear physical control system where the system state includes position and velocity, $\boldsymbol{x}_t = (\boldsymbol{p}_t, \boldsymbol{v}_t)^T$, with an elastic term, $H_{p,t}$, that is equal to zero and a viscous term, $H_{v,t} = A - K_tCA$. It is interesting to note the parallel between force and velocity representations that emerge in this implementation of the Kalman filter. Even though the VKF makes the assumption that neurons are driven by intended velocities, they play the role in Eqn. 3.16 of providing a force input to the system.

To our knowledge, nobody has directly compared the VKF to either the OLE or the PVA in on-line control. However, Koyama and colleagues demonstrated that a variant of the VKF called the Laplace-Gaussian filter (LGF) outperformed the PVA and OLE on-line [Koyama et al., 2010a]. The LGF and the VKF are both state-space decoders, and differ in only two main respects: the LGF fits an observation model that assumes Poisson noise statistics and log-linear tuning to intended velocity, whereas the VKF assumes linear Gaussian tuning of neurons to intended velocity. Importantly, this implies that the physical control system representing the VKF and the LGF will be of the same form: second order with no elastic terms.

Koyama and colleagues performed simulations and off-line trajectory reconstructions to determine that the key factor that allowed the LGF to outperform the PVA and OLE was its state-space formulation: they found no significant differences in the performance of the LGF relative to the VKF. We therefore take this as indirect evidence that the VKF would outperform the OLE and the PVA on-line. Why should this be the case? From an estimation standpoint, the interpretation would be that intended velocities really do evolve smoothly over time as implied by Eqn. 3.13, and so incorporating the fact enables better estimates of the velocity intent. However another interpretation is that the VKF and OLE are fundamentally different control systems, and 2nd order physical control systems may simply be easier to control than 1st order physical control systems.

Position velocity Kalman filter

Another widely used Kalman filter model is the PVKF. In contrast to the VKF, the PVKF assumes that neurons are tuned to both the intended position and intended velocity. Thus, the state is $\boldsymbol{x}_t^* = (\boldsymbol{p}_t^*, \boldsymbol{v}_t^*)^T$ and to encourage the state evolution model to obey physical laws, it is typically set to be

$$\begin{pmatrix} \boldsymbol{p}_t^* \\ \boldsymbol{v}_t^* \end{pmatrix} = \begin{pmatrix} I & \Delta I \\ \mathbf{0} & A_v \end{pmatrix} \begin{pmatrix} \boldsymbol{p}_{t-1}^* \\ \boldsymbol{v}_{t-1}^* \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \boldsymbol{\omega}_{v,t} \end{pmatrix}. \tag{3.17}$$

The PVKF observation model is

$$\boldsymbol{y}_t = (C_p, C_v) \begin{pmatrix} \boldsymbol{p}_t^* \\ \boldsymbol{v}_t^* \end{pmatrix} + \boldsymbol{\varepsilon}_t. \tag{3.18}$$

From Eqn. 2.10 we can compute K_t. Dissociating K_t into two parts corresponding to position and velocity as $K_t = (K_{p,t}, K_{v,t})^T$, we can write the estimated intended position and velocity as

$$\begin{pmatrix} \hat{\boldsymbol{p}}_t \\ \hat{\boldsymbol{v}}_t \end{pmatrix} = \begin{pmatrix} I - K_{p,t}C_p & \Delta I - K_{p,t}O \\ -K_{v,t}C_p & A_v - K_{v,t}O \end{pmatrix} \begin{pmatrix} \hat{\boldsymbol{p}}_{t-1} \\ \hat{\boldsymbol{v}}_{t-1} \end{pmatrix} + K_t \boldsymbol{y}_t, \tag{3.19}$$

where $O = \Delta C_p + C_v A_v$

In the PVKF, the estimated position is not, in general, equal to the integral of the estimated velocity. Even though the state evolution equation (Eqn. 3.17) biases estimates of position and velocity to obey this rule, it is not a hard constraint: a compromise between position and velocity will be estimated that best explains the observed spike rates. This then leaves one with a choice when trying to implement the PVKF, since both the position and velocity estimates cannot be simultaneously implemented. One common method is to use the estimated position as the implemented position, and to allow the implemented velocity to evolve as $\boldsymbol{v}_t = (\hat{\boldsymbol{p}}_{t+1} - \hat{\boldsymbol{p}}_t)/\Delta$ [Wu et al., 2006]. Here we denote this method as PVKF$_{position}$ and its state evolution equation is as

PVKF$_{position}$, **not** a simple physical control system:

$$\begin{pmatrix} \boldsymbol{p}_t \\ \hat{\boldsymbol{v}}_t \end{pmatrix} = \begin{pmatrix} I - K_{p,t}C_p & \Delta I - K_{p,t}O \\ -K_{v,t}C_p & A_v - K_{v,t}O \end{pmatrix} \begin{pmatrix} \boldsymbol{p}_{t-1} \\ \hat{\boldsymbol{v}}_{t-1} \end{pmatrix} + K_t \boldsymbol{y}_t. \tag{3.20}$$

With this implementation, the estimated velocity, $\hat{\boldsymbol{v}}_t$, becomes a latent variable that keeps the system from having a simple physical control system correlate. Experimental results from [Kim et al., 2008] show that when the PVKF is implemented in this way, its performance is inferior to the performance of the velocity-only Kalman filter. This fact is hard to rationalize from the estimation viewpoint, since the encoding model assumed by the PVKF typically fits the firing rates better than the encoding model assumed by the VKF. From the control system viewpoint, however, a possible explanation of this result is that the PVKF with position implementation is not a simple physical control system, and is therefore difficult to use.

There are other ways to implement the PVKF. One way is to use the estimated velocity as the implemented velocity and treat the estimated position $\hat{\boldsymbol{p}}_t$ as the hidden variable. Another is to use a linear combination of estimated velocity and estimated position as the implemented velocity. This latter method was shown in [Homer et al., 2013] to work better than the position-implementation of the PVKF. However, none of these versions give the subject a simple physical system without hidden states to learn to control.

There is one implementation of the PVKF, however, that does correspond to a simple physical control system with no hidden states. To do this, the implemented velocity is made equal to the estimated velocity, and *the estimated position becomes the integral of the estimated velocity*. Here we denote this method as PVKF$_{velocity}$ and its state evolution equation is as

PVKF$_{velocity}$, physical control system:

$$\begin{pmatrix} \boldsymbol{p}_t \\ \boldsymbol{v}_t \end{pmatrix} = \begin{pmatrix} I & \Delta I \\ -K_{v,t}C_p & A_v - K_{v,t}O \end{pmatrix} \begin{pmatrix} \boldsymbol{p}_{t-1} \\ \boldsymbol{v}_{t-1} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ K_{v,t} \end{pmatrix} \boldsymbol{y}_t. \tag{3.21}$$

As it turns out, this implementation of the PVKF has been used by Gilja and colleagues to achieve the best BCI control demonstrated to date [Gilja et al., 2012]. Their implementation of this equation was one of two design innovations of their ReFIT-KF algorithm, the other being a new method for re-calibrating the device from on-line training data. Although they derived Eqn. 3.21 in a different way, using a causal-intervention step that forces the variance of the position estimate to go to zero, the net effect is the same, and leads to a second order physical system with both elastic and damping terms.

The velocity-implementation of the PVKF has been shown to handily outperform the VKF [Gilja et al., 2012]. One interpretation of why this is true is that the causal intervention step better captures the fact that subjects know the position of the cursor from sensory feedback, so there should be no uncertainty about it. While this clearly cannot be exactly true, due to sensory feedback delays and sensory noise [Golub et al., 2013], it may be true enough to allow for better control. Another interpretation might be that neurons are driven by a non-volitional, positional signal representing the real position of the cursor, and this equation allows that "nuisance variable" term to be removed. However, yet another interpretation might be that this implementation of the PVKF results in a simple, 2nd order physical control system that still allows for neurons to modulate as a function of position. It may further be the case that 2nd order physical control systems that include a certain elastic component (a non-zero $H_{p,t}$ in Eqn. 3.4) are more usable than those systems, like the VKF, that do not have that component.

3.3.3 Higher order models

Of all the decoding algorithms we reviewed, none went beyond 2nd order. Given that 2nd order systems appear more controllable than 1st order systems, it is interesting to speculate as to

Table 3.1: **BCI decoders under physical system perspective.**

1st order physical system	PVA	$\boldsymbol{p}_{t+1} = \boldsymbol{p}_t + \Delta(k_s/n)DM^{-1}\boldsymbol{y}_t$
	OLE	$\boldsymbol{p}_{t+1} = \boldsymbol{p}_t + \Delta(C^TC)^{-1}C^T\boldsymbol{y}_t$
2nd order physical system	VKF	$\begin{pmatrix}\boldsymbol{p}_t\\ \boldsymbol{v}_t\end{pmatrix} = \begin{pmatrix}I & \Delta I\\ \mathbf{0} & A - K_tCA\end{pmatrix}\begin{pmatrix}\boldsymbol{p}_{t-1}\\ \boldsymbol{v}_{t-1}\end{pmatrix} + \begin{pmatrix}\mathbf{0}\\ K_t\end{pmatrix}\boldsymbol{y}_t$
	PVKF$_{velocity}$	$\begin{pmatrix}\boldsymbol{p}_t\\ \boldsymbol{v}_t\end{pmatrix} = \begin{pmatrix}I & \Delta I\\ -K_{v,t}C_p & A_v - K_{v,t}O\end{pmatrix}\begin{pmatrix}\boldsymbol{p}_{t-1}\\ \boldsymbol{v}_{t-1}\end{pmatrix} + \begin{pmatrix}\mathbf{0}\\ K_{v,t}\end{pmatrix}\boldsymbol{y}_t$
Not a simple physical system	PVKF$_{position}$	$\begin{pmatrix}\boldsymbol{p}_t\\ \hat{\boldsymbol{v}}_t\end{pmatrix} = \begin{pmatrix}I - K_{p,t}C_p & \Delta I - K_{p,t}O\\ -K_{v,t}C_p & A_v - K_{v,t}O\end{pmatrix}\begin{pmatrix}\boldsymbol{p}_{t-1}\\ \hat{\boldsymbol{v}}_{t-1}\end{pmatrix} + K_t\boldsymbol{y}_t$

whether a 3rd or 4th order system would be even easier to control. These higher-order systems may actually be a closer match to the human arm: in [Liu and Todorov, 2007], Liu and colleagues model the arm as a 3rd order linear physical system and are able to capture many of the emergent features of natural reaching movements. There have been instances in the literature that have included acceleration and higher order terms in their decoding algorithms. For example, Wu and colleagues compared the performance of a Kalman filter decoder with up to 6th order terms, and found that the 3rd order model consisting of position, velocity, and acceleration terms provided the best performance in their off-line trajectory reconstruction [Wu et al., 2006]. However, they used the position-implementation of their decoder, which we have already demonstrated does not correspond to a simple physical system beyond 1st order. It would be interesting to test how physical implementations of higher order control systems might perform on-line.

3.4 Discussion

The optimal way to implement a BCI decoding algorithm is an important question relevant to clinical deployment of neural prostheses. Here we recast this problem from the perspective of control system design, and derive the physical control systems corresponding to various types of decoders commonly used in BCI cursor control. This process enables new insights into BCI design, and suggests novel explanations about why some decoders have been shown to perform better than others. In particular, the literature suggests that: 1) 2nd order physical systems tend to be more usable than 1st order physical systems, 2) decoders that cannot be expressed as simple physical control systems do not appear to work as well as those that can be expressed this way, and 3) a 2nd order control system with elastic terms seems to work better than one without.

Recent work has highlighted the utility of approaching BCI design as a separate problem from inferring natural behavior [Chase and Schwartz, 2011, Tillery and Taylor, 2004]. Marathe and Taylor [2011] studied the effect of mapping one control parameter (e.g., position, velocity, or goal) to the control of another. They found that the optimal mapping was not necessarily one-to-one, but rather changed as a function of different types of decoding noise. Gowda et al.

Table 3.2: **BCI decoders comparison.**

decoders	scenario (winning decoder)	reference
PVA vs. OLE	off-line trajectory reconstruction (OLE)	[Salinas and Abbott, 1994] [Kass et al., 2005] [Chase et al., 2009]
	on-line closed loop control (equivalent)	[Chase et al., 2009] [Koyama et al., 2010a]
PVA/OLE vs. VKF	off-line trajectory reconstruction (VKF)	[Wu et al., 2006] [Koyama et al., 2010a]
	simulation (VKF)	[Koyama et al., 2010a]
PVKF$_{position}$ vs. VKF	on-line closed loop control (VKF)	[Kim et al., 2008]
PVKF$_{velocity}$ vs. VKF	on-line closed loop control (PVKF$_{velocity}$)	[Gilja et al., 2012]

[2014] have presented a thorough investigation of the dynamical systems properties of the PVKF. They found that certain implementations of the decoder could create workspace attractor points that might be detrimental to BCI control. These studies point out the gains that may be realized when BCI control is not constrained to reflect the neural encoding of natural arm dynamics, and emphasize the importance of the physical control system perspective when interpreting BCI performance.

3.4.1 Alternate views of motor cortical recruitment

Decades of motor control studies have established that neural activity in motor cortex correlates with various features of movement. In BCI design, it is common to interpret these correlations by thinking of neurons as being tuned to the desired outcome or intended movement of an effector. However, neural activity can be flexibly dissociated from the effector [Schieber, 2011]: using operant conditioning, individual neurons can be trained to correlate and decorrelate from particular muscles [Fetz, 1969, Fetz and Finocchio, 1971], even when spike triggered averaging of EMG traces provide evidence of monosynaptic connections between the neuron and the muscle [Davidson et al., 2007]. Evidence of a flexible relationship between neural activity and behavior also comes from natural movement paradigms. Neural activity in motor cortex readily changes during motor learning [Gandolfo et al., 2000, Ganguly and Carmena, 2009, Jarosiewicz et al., 2008, Paz et al., 2003, Sadtler et al., 2014, Wise et al., 1998]. Neural activity has also been shown to change during associative learning: neurons in M1 will respond to the color of a target during an associative learning task, and will often maintain that tuning after color is no longer relevant [Zach et al., 2008]. Even context can modulate neural tuning. Hepp-Reymond and colleagues [1999] demonstrated that neurons in primary motor cortex are sensitive to the context of an isometric force task, and will change their firing to particular force levels according to how many force targets are presented in the task.

These studies suggest that motor cortical activity is extremely fungible, restructuring itself in the face of new task demands in a manner that is not at all well understood. It is hard to reconcile this view with a static view of tuning to intended movements, unless one assumes that these motor intent signals can themselves be dissociated from the motor outcome [Chase and Schwartz, 2011]. If this is the case, there may be only subtle differences among the viewpoints that neurons tune to flexible motor intent signals, that neurons act as control signals to drive an effective behavior, or that populations of neurons act as a flexible pattern generator on which movements can be built [Shenoy et al., 2013]. In this review, we mainly wish to highlight the importance of these multiple viewpoints when attempting to interpret BCI performance and, ultimately, design the optimal decoding algorithm.

3.4.2 Relationships to embodiment, internal models, and natural motor control

Natural motor control is fraught with computational difficulties, not least of which is the necessity to compensate for noisy, delayed sensory feedback. To generate fast, dexterous movements, it is necessary to compensate for these sensory delays. It is widely believed that we do this with the aid of internal models that allow us to predict, in real time, the outcomes of our motor commands before sensory feedback becomes available [Crapse and Sommer, 2008, Shadmehr et al., 2010]. These internal (forward) models are thought to take as input efference copies of our motor commands, and use them to predict the sensory consequences, such as the new arm or eye position, that result from those commands [Sommer and Wurtz, 2002, Wolpert et al., 1995]. Essentially, these models embody our internal conception of the physics of our limbs and how they respond to our motor commands. These predicted locations can then be used as the basis for planning the next movement before real sensory feedback becomes available, allowing for faster motor sequence production.

Internal models may also be used in BCI control. Motor commands of subjects using a BCI to control a computer cursor in a center-out movement task are more appropriate to the real time position of the cursor than the last sensory feedback position, indicating that subjects compensate for sensory feedback delays while using a BCI [Golub et al., 2012]. Further, differences between a subject's internal model of the decoder and the actual cursor dynamics may explain errors in on-line control [Golub et al., 2013, 2015].

It is intriguing to speculate on the utility of internal models in BCI control. Internal models are thought to be a key component in motor adaptation [Kawato, 1999, Shadmehr et al., 2010], and thus subjects who build internal models of BCIs may be able to take advantage of all of the computational motor adaptation machinery that natural motor control relies upon. A reliable internal model of the BCI device may also be the key to embodiment, when the device feels like a natural extension of one's body [Giummarra et al., 2008, Schwartz et al., 2006]. However, can subjects build internal models of *any* BCI device? While the answer to this question is well beyond the scope of this review, we posit that it might be easier to build internal models of physical systems than it is to build internal models of non-physical systems, simply because all of the internal models we build in the context of natural reaching are, by definition, models of physical systems. It could be that the reason decoding algorithms corresponding to physical systems appear to be more useable than those that do not is because they are more readily conceptualized with an internal model.

3.5 Conclusion

In this chapter, we have focused primarily on the overall form of the physical control system represented by different decoders: e.g., whether it is 1st order or 2nd order and whether it contains both viscous and elastic elements, etc. We have not focused on the particular values of those parameters as much, except to point out that the PVA and OLE, which have different mappings between firing rates and cursor movement, perform similarly on-line [Chase et al., 2009]. Under the estimation-framework, performance will be best with mappings from neural activity to cursor movement that best capture the natural neural tuning. Under the control framework, performance will be best with those mappings that most accurately capture the volitionally usable correlation structures within the neural population. Of course, all of this ignores long-term learning, which may allow subjects to become proficient even at non-intuitive mappings [Ganguly and Carmena, 2009, 2010]. Further work will need to be done to determine the precise details that determine which mappings from neural activity to movement perform better than others, and how learning may ultimately play a role in sculpting their performance [Orsborn et al., 2014]. In chapter 4, we take a first step on solving the problem about how to design a BCI system which is optimal after the subject fully learnt how to control it.

Chapter 4

Design of a provably-optimal BCI system

In the previous chapter we recast the BCI system design problem from the perspective of control system design, and derived the physical control systems corresponding to various types of decoders commonly used in BCI cursor control. This process suggested novel explanations about why some decoders have been shown to perform better than others. In particular, we found that: 1) 2nd order physical systems tended to be more controllable than 1st order physical systems, 2) decoders that could not be expressed as simple physical control systems did not appear to work as well as those that could be expressed this way, and 3) a 2nd order control system with elastic terms seemed to work better than one without. Exactly why this might be so is not certain. It could be that certain physical systems are better matched to the signal and noise properties of the recorded neural populations. Another intriguing possibility is that the brain is able to conceptualize some systems better than others. Natural motor control appears to rely on the use of internal models that can help compensate for sensory feedback delays and enable the selection of appropriate motor commands [Golub et al., 2015, Shadmehr et al., 2010]. It could be that the most controllable system is the one that can best take advantage of the brain's ability to model it.

We have focused primarily on the overall form of the physical control system represented by different decoders: e.g., whether it is 1st order or 2nd order, whether it contains both viscous and elastic elements, and for the 2nd order model, Eqn. 3.4, how to adjust $H_{p,t}$, $H_{v,t}$ and $M_{v,t}$. We have not focused on the particular values of those parameters as much, except to point out that the PVA and OLE, which have different mappings between firing rates and cursor movement, perform similarly on-line [Chase et al., 2009]. Do the details of the control system parameters matter at all?

The answer is certainly yes. Sadtler and colleagues have recently demonstrated that some mappings between neural activity and cursor movement are easily learned, while others are very difficult to learn [Sadtler et al., 2014]. In that study they discovered that subjects could easily learn to map an existing pattern of neural activity to an arbitrary movement; what was difficult was creating new correlation patterns in the neurons themselves. This suggests that, provided the mapping from neural activity to cursor movement preserves the natural correlations within the neural population, two physical systems of the same form may be equivalently-controllable. However, Gilja and colleagues [Gilja et al., 2012] have noted a significant improvement in control if tuning curves are re-estimated from on-line training data, which would seem to conflict with the previous statement. Further, while Sadtler and colleagues noted substantial improvement in performance with short amounts of practice on novel ‘within-manifold’, i.e., correlation preserving, mappings, the performance under those mappings was still substantially worse than performance with the initially-estimated natural tuning curves. Under the estimation-framework, performance will be best with mappings from neural activity to cursor movement that best capture the natural neural tuning. Under the control-framework, performance will be best with those mappings that most accurately capture the volitionally-controllable correlation structures within the neural population. For example, in the velocity Kalman filter, $A_{p,t} = \mathbf{0}$, $A_{v,t}$ and $B_{v,t}$ are determined by the recursive relationship from the system evolution equation and observation equation. Those parameters are optimal under the estimation perspective [Wu et al., 2006], however, they are not guaranteed to be optimal under the control system perspective.

In this chapter, we will focus on studying the precise details that determine which mappings from neural activity to movement perform better than others, and how learning may ultimately play a role in sculpting their performance [Orsborn et al., 2014]. A preliminary version of this work that introduced the control theoretic approach to BCI was published in [Zhang and Chase, 2016]. A full version of the results in this chapter has been published as [Zhang and Chase, 2018].

4.1 Introduction

Recent clinical successes highlight an important question in BCI design: what mapping between neural activity and device movement will enable the most proficient long-term control? Currently, most BCI decoding algorithms are designed from a *biomimetic*, or *estimation* standpoint, in which neurons are assumed to represent upcoming intended movements and recorded firing rates are noisy observations of that underlying intent. Under this framework, BCI design is properly treated as a signal estimation problem, where the goal is to move the prosthesis to achieve as close a match to the decoded intent as possible. A BCI designed in this fashion will tend to be biomimetic, since it leverages the endogenous tuning to motor intent.

If tuning to motor intent were static, BCIs designed from an estimation standpoint would be statistically optimal provided the assumptions made about the form of intention tuning were correct. From the previous chapter, we can see the differences between various decoding algorithms tend to stem from differences in the assumptions they make about neural tuning. However, tuning to motor intent is not necessarily static: in arm control learning studies, tuning curves change in a manner that reflects increased task performance [Gandolfo et al., 2000, Mandelblat-Cerf et al., 2011, Paz et al., 2005, Richardson et al., 2012, Wise et al., 1998], and non-biomimetic BCI mappings induce tuning curve changes that also increase behavioral performance [Chase et al., 2012, Fan et al., 2014, Ganguly et al., 2011, Jarosiewicz et al., 2008]. Given that tuning curves can change, it is no longer guaranteed that a decoder designed for one set of neural tunings would be better than a decoder designed for a different set that could be learned.

Here we present a method to design a BCI mapping that is optimal *post-learning*. Instead of trying to mimic a subject’s biological decoding process to minimize training time, we instead assume that the subject will learn, over weeks of practice, to use whatever mapping we give him, and we develop an approach to determine which mapping will enable the most proficient control after learning has occurred. Note that it is not clear the extent to which the assumption that the

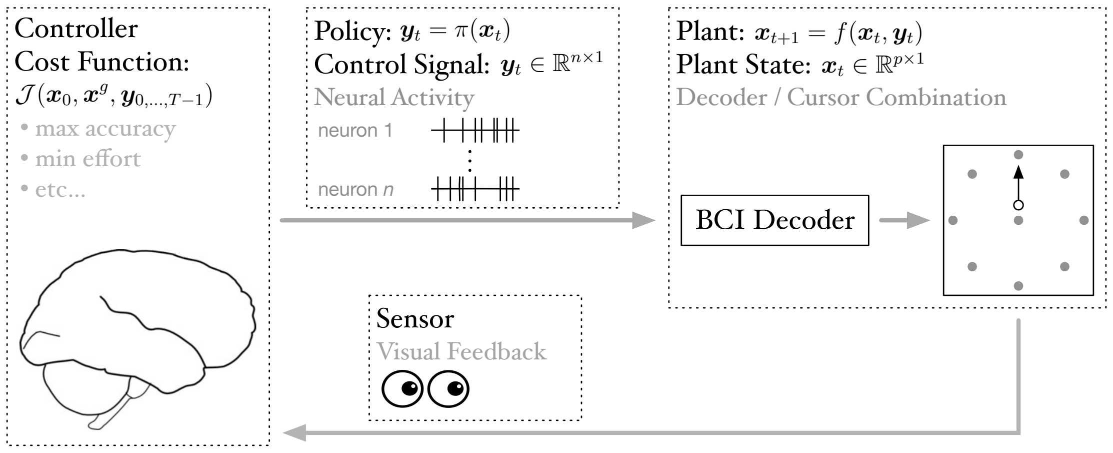

Controller
Cost Function:
$\mathcal{J}(\boldsymbol{x}_0, \boldsymbol{x}^g, \boldsymbol{y}_{0,\dots,T-1})$
• max accuracy
• min effort
• etc...
Policy: $\boldsymbol{y}_t = \pi(\boldsymbol{x}_t)$
Control Signal: $\boldsymbol{y}_t \in \mathbb{R}^{n\times 1}$
Neural Activity
neuron 1
neuron n
Plant: $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{y}_t)$
Plant State: $\boldsymbol{x}_t \in \mathbb{R}^{p\times 1}$
Decoder / Cursor Combination
BCI Decoder
Sensor
Visual Feedback

Figure 4.1: **Schematic of a BCI as a feedback control system.** The major components of a feedback control system (namely, the controller, control signals, plant, and feedback) are laid out on top of a typical BCI cursor control schematic, where the brain is identified as the controller, the control signals are neural activity (often tapped out of primary motor cortex), the plant is the combination of the BCI decoder and the cursor, and feedback is accomplished by watching the cursor movements. This figure is essentially the same as Fig. 3.1 while adds the formulation of the cost function.

subject will learn whatever mapping we give him is true, a fact that we take up in section 4.5. Specifically, we take an optimal control theory approach and treat the BCI system as a control system in which the prosthetic effector, together with the decoding algorithm, act as a motor plant that the subject must learn to control through trial-and-error (Fig. 4.1). We then make two assumptions: (1) the brain acts as an optimal controller, and (2) the end result of learning is to produce optimal control signals, i.e., signals that minimize an internal cost function. If these two conditions are met, it is possible to formulate a rigorous definition for the *usability* of a device, and the provably optimal (i.e., most usable) mapping can be written as the solution of a constrained optimization problem.

Our approach is similar in spirit to other approaches that take subject learning into account [Shenoy and Carmena, 2014], such as the closed-loop decoder algorithms employed by Carmena and colleagues [Dangi et al., 2014, Orsborn et al., 2014, Shanechi et al., 2016]. These co-adaptive procedures have been rigorously studied by Merel and colleagues [Merel et al., 2015], who have linked the process of user intention estimation to a novel class of imitation learning problem [Merel et al., 2016]. Our approach extends this body of work by formally linking it to a particular class of optimal control problem that we term “optimal plant design”. This enables all neural prosthetic design problems, be they robotic hands, computer cursors, spelling devices, or other configurations, to be treated under a common unifying framework.

In section 4.2, we define this problem formally by providing a rigorous definition of device usability. In section 4.3, we outline a formal algorithm for solving this problem for the specific case of a fixed linear mapping with a quadratic cost function (under the case of both signal-dependent and signal-independent noise). In section 4.4 we apply this algorithm to design an optimal 2D BCI for point-to-point reaching tasks. Finally, in section 4.5 we conclude with a discussion.

4.2 Problem statement

How should systems be designed for optimal usability? It is possible to state this question within a general mathematical framework using ideas from optimal control theory. An optimally controlled system must have three components (Fig. 4.1). First, there is the *plant*: the system being controlled. This system can be quite general. It could be mechanical, like a robotic arm or car, or it could be chemical, like a petroleum processing plant. The plant is driven into different states by control signals: external forces that can change the state of the plant. If we let $\boldsymbol{x}_t \in \mathbb{R}^{p \times 1}$ denote the p-dimensional state of the plant at time t, then we can capture the physics of the plant through the equation

$$\text{Plant:} \quad \boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{y}_t), \tag{4.1}$$

where $\boldsymbol{y}_t \in \mathbb{R}^{n \times 1}$ denotes the n-dimensional control input at time t. In a BCI, this plant would capture the combination of the decoding algorithm that maps neural activity into device movement and the physics of the device itself.

The second component of an optimally controlled system is the *cost function*, which determines the overall utility of different types of movements and control signals. In a petroleum plant, the cost might incorporate terms relating to the efficiency of a chemical reaction or the concentration of unwanted byproducts. For a BCI, the cost function should incorporate the user's goal (choose this letter or pick up this object) and might also capture the effort to produce particular control signals. If a movement extends over some time $t = 0 \ldots T$, the overall trajectory cost would depend on the initial state of the device, $\boldsymbol{x}_0$, the goal $\boldsymbol{x}^g$ (which may itself be a function of time), and the series of control signals that one uses to pilot the device, $\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{T-1}$. We can write the cost $\mathcal{J}$ of this trajectory explicitly as

$$\text{Cost function:} \quad \mathcal{J}(\boldsymbol{x}_0, \boldsymbol{x}^g, \boldsymbol{y}_{0,\ldots,T-1}), \tag{4.2}$$

Control signals which minimize this cost are, by definition, optimal, and denoted $\boldsymbol{y}_t^*$, while the minimal cost that those control signals produce is denoted as $\mathcal{J}^*(\boldsymbol{x}_0, \boldsymbol{x}^g) = \mathcal{J}(\boldsymbol{x}_0, \boldsymbol{x}^g, \boldsymbol{y}^*_{0,\ldots,T-1})$.

The third component of an optimally controlled system is the *control policy*, which determines the control signals that one should use to perform a particular task when the plant is in state $\boldsymbol{x}_t$. The control policy can be captured through the general equation

$$\text{Control policy:} \quad \boldsymbol{y}_t = \pi(\boldsymbol{x}_t). \tag{4.3}$$

An optimally controlled system would use an optimal control policy that chooses control signals incurring minimal cost through Eqn. 4.2; we denote such a system as π^*. Thus, $\boldsymbol{y}_t^* = \pi^*(\boldsymbol{x}_t)$.

A number of studies have found evidence that physiological motor control conforms to predictions from optimal control theory [Todorov and Jordan, 2002]: task-irrelevant errors tend to be left uncorrected [Liu and Todorov, 2007, van Beers et al., 2013], movements have been shown to distribute optimally across effectors [Diedrichsen, 2007, Fagg et al., 2002, Haruno and Wolpert, 2005], trajectory dynamics during control of complex effectors can be predicted using optimal feedback control models [Nagengast et al., 2009], and reflexes are programmed to respond with gains that reflect the task geometry and goal [Nashed et al., 2012, Omrani et al., 2013, Pruszynski and Scott, 2012]. With this as justification, we take as an assumption that the brain acts as an optimal controller, and that practice with a given BCI device will eventually lead, through learning, to the subject controlling the device with optimal control signals (i.e., neural activity patterns) according to an optimal policy. We discuss the validity of these assumptions in section 4.5.

To design a BCI that is most usable, we must first rigorously define what we mean by usability. Here, we introduce the mathematical definition of BCI usability in section 4.2.1, and frame a special case of the problem in section 4.2.2 that we solve in section 4.3.

4.2.1 Optimal plant design

It is important to realize that the cost of a movement depends on the plant. Imagine a situation in which one is trying to accurately position a block. If the block is light it might be possible to position it using only a small amount of force to guide it into place. If the block is quite heavy, however, larger forces will be required during positioning. These larger forces might require more energy, and therefore might incur larger costs through the cost function, even if the cost function itself has not changed. We can explicitly represent the plant parameters, ϕ_f, in the cost function to emphasize this point:

$$\mathcal{J}^*(\boldsymbol{x}_0, \boldsymbol{x}^g; \phi_f) = \min_{\boldsymbol{y}_{0,\ldots,T-1}} \mathcal{J}(\boldsymbol{x}_0, \boldsymbol{x}^g, \boldsymbol{y}_{0,\ldots,T-1}; \phi_f). \tag{4.4}$$

With this in mind, we can now define the usability of a BCI as the average cost of using the device in an optimal manner, where the average is taken over all of the movements that one might want to perform. Formally, we define the usability, $\mathcal{U}$, as

$$\text{Usability:} \quad \mathcal{U}(\phi_f) = -\mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{x}^g}\left[\mathcal{J}^*(\boldsymbol{x}_0, \boldsymbol{x}^g; \phi_f)\right], \tag{4.5}$$

where the expectation is taken over all initial states and goals. The negative sign ensures that increases in usability correspond to decreases in expected cost. In words, we can say that a plant with parameters ϕ_1 is more usable than a plant with parameters ϕ_2 if, for the same set of tasks, the expected optimal cost under ϕ_1 is less than the expected optimal cost under ϕ_2. The optimal plant is defined as the plant which globally maximizes this usability:

$$\text{Optimal plant:} \quad \phi_f^* = \arg\max_{\phi_f} \mathcal{U}(\phi_f). \tag{4.6}$$

In practice, it is likely that other design factors, such as overall device weight or size, might constrain the choice of plant parameters. Among the allowable set of parameters, however, choosing parameters that maximize Eqn. 4.6 are guaranteed to result in the lowest expected cost once learning has occurred.

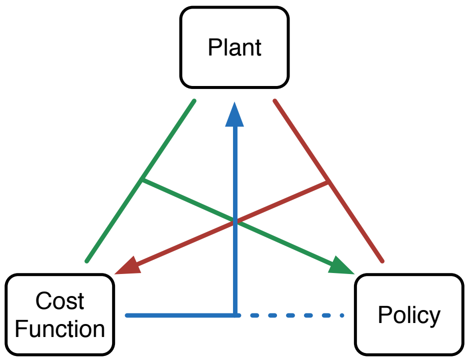


Figure 4.2: **Cost function, plant, policy, with arrows connecting them to illustrate the various types of control theory problems.** *Green*: optimal forward control. *Red*: inverse control. *Blue*: optimal plant design.

Relationship to other problems in control theory

The optimal plant design problem is related to, but distinct from, typical classes of problems in control theory. In the forward control problem, one is given knowledge of the plant and the cost function, and the job is to find the control signals that achieve the task goals with minimal cost [Anderson and Moore, 1989] (Fig. 4.2, green arrow). In the inverse control problem, one is given knowledge of the plant and a set of observed control signals that were issued for particular plant states, and the job is to infer the cost function for which those control signals would be optimal [Abbeel and Ng, 2004, Dvijotham and Todorov, 2010] (Fig. 4.2, red arrow).

In the optimal plant design problem, one is given the cost function, and it is assumed that the ultimate plant will be operated in an optimal fashion. Thus, while the policy is not directly specified, it could be derived for any particular plant as a forward control problem. The goal of optimal plant design is to find the parameters of the plant that result in the least expected cost (Fig. 4.2, blue arrow).

4.2.2 A special case: optimal design of linear plants under quadratic costs

The optimal plant design problem as stated in section 4.2.1 is too general to permit solution without additional assumptions. Here we outline the specifics of a special case of this problem that we will solve in section 4.3.

Linear plant assumption:

We will restrict ourselves to the case of linear BCI mappings that can be parameterized in the following way as Eq 3.2:

$$\boldsymbol{x}_{t+1} = H\boldsymbol{x}_t + M\boldsymbol{y}_t, \tag{4.7}$$

where H captures the system dynamics that describe how the system will evolve without the control signal input, and M describes how neural activity changes the device state. $\boldsymbol{y}_t$ are mean-centered neural firing rates. Different from Eq 3.2, here we only consider time invariant systems, so that H and M do not depend on t. Among those existing widely used decoding algorithms, the PVA [Georgopoulos et al., 1986, Taylor et al., 2002, Velliste et al., 2008] and the OLE [Chase et al., 2009, Wang et al., 2007] are both static. The Kalman filter [Hochberg et al., 2012, Sadtler et al., 2014, Wu et al., 2006] technically has time varying parameters, but in practice the Kalman gain converges to a constant within a few timesteps, and many researchers even initialize it at the converged values to keep it time-invariant [Malik et al., 2011, Sadtler et al., 2014].

We further restrict ourselves to consider only physical control systems (e.g., those in which position is a proper integral of velocity), because we found in chapter 3 that physical systems appear to be more easily controlled than non-physical systems. A second order physical system has a dynamics term, H, that can be further parameterized as

$$H = \begin{pmatrix} I & \Delta I \\ H_p & H_v \end{pmatrix}, \tag{4.8}$$

where H_p and H_v represent the elastic and viscous components of the prosthesis dynamics, respectively, and Δ is a small discretized timestep. As reviewed in chapter 3, two physical systems with the same order but different system parameters, such as the ReFIT-Kalman Filter [Gilja et al., 2012] and the velocity Kalman filter [Kim et al., 2008], can have very different overall performances (also, see [Gowda et al., 2014]). In this manuscript, we focus on the 2nd order system, where the state comprises the implemented position and velocity:

$$\begin{pmatrix} \boldsymbol{p}_{t+1} \\ \boldsymbol{v}_{t+1} \end{pmatrix} = \begin{pmatrix} I & \Delta I \\ H_p & H_v \end{pmatrix} \begin{pmatrix} \boldsymbol{p}_t \\ \boldsymbol{v}_t \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ M_v \end{pmatrix} \boldsymbol{y}_t, \tag{4.9}$$

where M_v defines the mapping between recorded neural activity and the control signals driving state evolution.

To be realistic, we cannot ignore that the neural activity, $\boldsymbol{y}_t$, will be a mixture of both signal and noise. The noise could be in the Poisson-like firing of the neurons themselves [Churchland et al., 2006, Tolhurst et al., 1983], or it could be misclassification of recording artifacts as spiking events. We can make this explicit in our formulation, and write $\boldsymbol{y}_t$ as the sum of a signal component, $\boldsymbol{z}_t$, together with both signal-dependent and signal-independent Gaussian noise:

$$\boldsymbol{y}_t = \boldsymbol{z}_t + \sum_{i=1}^{n} \sqrt{\kappa_i} \epsilon_{i,t} I_i \boldsymbol{z}_t + \boldsymbol{\omega}_t. \tag{4.10}$$

The summation represents signal-dependent noise, where I_i is an $n \times n$ matrix with a 1 at location (i,i) and 0 elsewhere, $\epsilon_{i,t} \sim \mathcal{N}(0,1)$ is standard Gaussian noise, and $\kappa_i > 0$ controls the magnitude of the variance of the noise. This is equivalent to assuming that the ith neuron's firing rate is distributed as $\mathcal{N}(z_{i,t}, \kappa_i z_{i,t}^2)$, which makes the firing rate approximately Poisson-like. The remaining term, $\boldsymbol{\omega}_t \sim \mathcal{N}(\mathbf{0}, W)$ represents the signal-independent noise.

Quadratic cost assumption:

Once the subject is familiar with the BCI system, he will try to generate control signals which can minimize the cost function. In this manuscript, we consider the finite horizon quadratic cost function which has been widely used in motor learning studies [Diedrichsen, 2007, Shadmehr and Krakauer, 2008, Todorov and Jordan, 2002], written as

$$\mathcal{J}(\boldsymbol{x}_0, \boldsymbol{x}^g, \boldsymbol{z}_{0,\dots,T-1}) = \sum_{t=0}^{T} \mathbb{E}\left(\boldsymbol{x}_t^T Q_t \boldsymbol{x}_t\right) + \sum_{t=0}^{T-1} \boldsymbol{z}_t^T R_t \boldsymbol{z}_t, \tag{4.11}$$

where $Q_t \succeq 0$ and $R_t \succ 0$. Generally speaking, the first term measures the movement accuracy and implicitly takes the goal $\boldsymbol{x}^g$ into consideration. The second term measures the total effort required to finish the task. The magnitude of matrices Q_t and R_t provides a weighting between those two terms. The expectation in Eqn. 4.11 is taken over all of the random variables defining the noise in the recorded neural activity, $\{\epsilon_{i,t}, \boldsymbol{\omega}_t\}$.

4.3 Problem solutions

In this section, we solve for the optimal plant parameters (H_p, H_v, and M_v from Eqn. 4.9) under the assumption of learned optimal control of a finite horizon quadratic cost function (as in Eqn. 4.11). To solve the optimal plant design problem, we first solve the forward control problem for a given instantiation of a plant. The solution to the forward control problem is a modified version of an LQR controller, adjusted to account for both signal-dependent and signal-independent noise. This is reviewed in section 4.3.1. We next demonstrate how the optimal dynamics can be solved for by a gradient descent procedure, and present a modification of the forward control algorithm that computes the gradient (section 4.3.2). Finally, we derive the optimal control mapping in section 4.3.3.

4.3.1 Optimal forward control

To find the system with lowest optimal cost, we first compute the optimal cost for a specific control system. For linear plant dynamics and a quadratic cost function, the solution (without signal-dependent noise) is the classic linear quadratic regulator (LQR) that can be solved using a Riccati recursion [Kwakernaak and Sivan, 1972]. This algorithm must be modified slightly for the signal-dependent noise that neurons tend to exhibit (Eqn. 4.10). Following an approach similar to [Todorov, 2005], we derive the optimal cost for a given set of plant parameters as

$$\mathcal{J}^*(\boldsymbol{x}_0, \boldsymbol{x}^g; H, M) = \mathrm{tr}(P_0 X_0) + \sum_{t=0}^{T-1} \mathrm{tr}\left(P_{t+1} M W M^T\right), \tag{4.12}$$

where $X_0 = \boldsymbol{x}_0 \boldsymbol{x}_0^T$ and P_0 is derived through backwards recursion as the result of a modified Riccati algorithm outlined in Algorithm 1. A detailed derivation of this algorithm is provided in Appendix B.1. For problems like target reaching, which we will study in section 4.4, the target $\boldsymbol{x}^g$ is embedded into the state $\boldsymbol{x}_t$ while system parameters H, M and cost function parameters R, Q are independent of $\boldsymbol{x}^g$. In such a case, P_t does not depend on either $\boldsymbol{x}_0$ or $\boldsymbol{x}^g$, and the usability

Algorithm 1 Modified Riccati Recursion with Signal-Dependent Noise

```
set $P_T = Q_T$
for $t = T-1, \ldots, 0$ do
    $D_t = R_t + M^T P_{t+1} M + \sum_{i=1}^n \kappa_i I_i M^T P_{t+1} M I_i$
    $L_t = -D_t^{-1} M^T P_{t+1} H$
    $P_t = Q_t + H^T P_{t+1}(H + ML_t)$
    the optimal policy: $\boldsymbol{z}_t^*(\boldsymbol{x}) = L_t \boldsymbol{x}$
end for
```

of the BCI system becomes

$$\begin{aligned}\mathcal{U}(H, M) &= -\mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{x}^g}\left[\mathcal{J}^*(\boldsymbol{x}_0, \boldsymbol{x}^g; H, M)\right] \\ &= -\mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{x}^g}\left[\operatorname{tr}(P_0 X_0) + \sum\nolimits_{t=0}^{T-1} \operatorname{tr}\left(P_{t+1} M W M^T\right)\right] \\ &= -\operatorname{tr}\big(P_0\, \mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{x}^g}(X_0)\big) - \sum\nolimits_{t=0}^{T-1} \operatorname{tr}\left(P_{t+1} M W M^T\right). \end{aligned} \tag{4.13}$$

In this case, P_t (and the optimal policy L_t) may be computed as in Algorithm 1. For notational simplicity, in the following sections, we will also use X_0 to denote $\mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{x}^g}(X_0)$.

4.3.2 Optimization of decoder dynamics

Now that we can compute the usability of a particular plant, we derive a method for optimizing the usability across plants. We start by optimizing the system's usability over the decoder dynamics terms, H from Eqn. 4.7 (parameterized as in Eqn. 4.8), under the assumption that M is fixed. These parameters are found through gradient descent. Below we outline this solution and present a modification of Algorithm 1 that computes the gradients on each iteration.

Generally, the usability $\mathcal{U}$ can be a complex function of H and there exists no closed-form solution of this optimization problem. We instead solve for H^* through gradient descent. Starting from an initial guess H^0, we iteratively update our estimate of H according to the gradient update rule

$$H^{\tau+1} = H^{\tau} - \eta_\tau \partial\mathcal{U}/\partial H \tag{4.14}$$

where η_τ is the learning rate.

To compute the derivative $\partial\mathcal{U}/\partial H$, we note that the derivative on a matrix is a matrix of the derivative on each element of that matrix. So, denoting the element of H as h, $\partial\mathcal{U}/\partial H$ is a matrix with the element $\partial\mathcal{U}/\partial h$ at the position corresponding to h. From Eqn. 4.13, $\partial\mathcal{U}/\partial h$ can be computed as

$$-\frac{\partial\mathcal{U}}{\partial h} = \operatorname{tr}\left(\frac{\partial P_0}{\partial h} X_0\right) + \sum_{t=0}^{T-1} \operatorname{tr}\left(\frac{\partial P_{t+1}}{\partial h} M W M^T\right). \tag{4.15}$$

It thus remains to compute $\partial P_t/\partial h$ for $0 \le t \le T$. Since P_t is computed backward iteratively, we also use dynamic programming to compute the partial derivative of P_t. Notice M, Q_t, and R_t do not depend on h, so derivatives of M, Q_t and R_t with respect to h will all be $\mathbf{0}$. Algorithm 2

Algorithm 2 Dynamic Programming to Compute the Gradient $\partial\mathcal{U}/\partial h$

```
set $P_T = Q_T$ and $P'_T = \mathbf{0}$
for $t = T-1, \ldots, 0$ do
    $D'_t = M^T P'_{t+1} M + \sum_{i=1}^{n} \kappa_i I_i M_t^T P'_{t+1} M_t I_i$
    $L'_t = -D_t^{-1} \left( D'_t L_t + M^T P'_{t+1} H + M^T P_{t+1} H' \right)$
    $P'_t = (H'^T P_{t+1} + H^T P'_{t+1})(H + M L_t) + H^T P_{t+1}(H' + M L'_t)$
end for
```

outlines a procedure for solving $\partial P_t/\partial h$, where for notational simplicity we denote the derivative of a matrix F with respect to h as F'.

Note that Algorithms 1 and 2 can be combined to simultaneously solve for the optimal control signals (and costs) for a given plant, as well as the gradient of that cost across dynamics parameter H.

Finally, it should be noted that the gradient descent procedure can get stuck in local minima. For the particular linear plant with quadratic cost function outlined here, it turns out the costs are smooth enough that this is not an issue. Examples verifying this will be shown in section 4.4. For situations in which the cost landscape is more complex, various sampling or annealing techniques would need to be employed to mitigate the effects of local minima.

4.3.3 Optimization of control signal mappings

Next we derive the best way to map neural activity to the control of cursor velocity, by optimizing the system's usability as a function of M_v from Eqn. 4.9. We can interpret each column of M_v, $\boldsymbol{m}_{v,i}$, as the "push" contributed by neuron i to the cursor. This push can be further dissected into its magnitude, $\|\boldsymbol{m}_{v,i}\|$, and its direction $\boldsymbol{m}_{v,i}/\|\boldsymbol{m}_{v,i}\|$. We first study the effect of changes in the pushing magnitudes on the system's usability. We will then study how changes in the direction vectors of the mapping matrix affect BCI usability.

We can study general changes in pushing magnitude by defining a scaling factor for the pushing magnitude of the ith neuron as $\gamma_i > 0$. Applying these changes to every neuron leads to a new mapping matrix $M_v\Gamma$, where Γ is a diagonal matrix with elements γ_i. We prove in Appendix B.2 that this change in the mapping matrix leaves P_t unchanged, and results in a new usability of

$$\mathcal{U}^* = -\mathrm{tr}(P_0 X_0) - \sum_{t=0}^{T-1} \mathrm{tr}\left(P_{t+1} M \Gamma W \Gamma M^T\right), \tag{4.16}$$

with optimal policy

$$\boldsymbol{z}_t^*(\boldsymbol{x}) = \Gamma^{-1} L_t \boldsymbol{x}. \tag{4.17}$$

We can see that the first term of the usability, $-\mathrm{tr}(P_0 X_0)$, does not depend on our gain change Γ. In fact, Γ only affects the usability through the signal-independent noise (with covariance W from Eqn. 4.10). Therefore, if W is fixed, to minimize the optimal cost, we want γ_i as small as possible. From Eqn. 4.17 we can see that reducing γ_i will require the firing rates to increase. This makes intuitive sense: large firing rates lead to larger signal-to-noise ratios (SNRs), because the signal moves farther from the signal-independent noise floor (while maintaining a constant ratio to the signal-dependent noise).

However, neurons cannot fire at arbitrarily large rates; rather firing rates should be constrained to lie within some physiological range, $y_{i,\min} \leq y_i \leq y_{i,\max}$. Thus, γ_i cannot be arbitrarily small. The optimal γ_i^* should take the smallest value consistent with keeping the optimal firing rates computed by Eqn. 4.17 within their physiological range.

Next we fix the modulation depth $\boldsymbol{m}_{v,i}$ and study the neuron's pushing direction. For simplicity, here we assume the modulation depth for each neuron is 1, i.e., $\boldsymbol{m}_{v,i}$ is a unit vector as $\|\boldsymbol{m}_{v,i}\| = 1$. Directly optimizing the usability on $\boldsymbol{m}_{v,i}$ is very challenging. We tackle the problem indirectly in the following way. Notice that by taking Eqn. 4.10 back into Eqn. 4.9 and introducing the *intended push*, defined as

$$\boldsymbol{u}_t = M_v \boldsymbol{z}_t, \tag{4.18}$$

we have

$$\begin{pmatrix} \boldsymbol{p}_{t+1} \\ \boldsymbol{v}_{t+1} \end{pmatrix} = \begin{pmatrix} I & \Delta I \\ H_p & H_v \end{pmatrix} \begin{pmatrix} \boldsymbol{p}_t \\ \boldsymbol{v}_t \end{pmatrix} + \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{u}_t \end{pmatrix} + \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{\xi}_t \end{pmatrix}, \tag{4.19}$$

where $\boldsymbol{\xi}_t = M_v \left(\sum_{i=1}^n \sqrt{\kappa_i} \epsilon_{i,t} I_i \boldsymbol{z}_t + \boldsymbol{\omega}_t \right)$ is the noise term.

Now solving the forward control problem can be divided into two steps. First, find the optimal solution, $\boldsymbol{u}^*$, for the system with dynamics given in Eqn. 4.19. Then, find the corresponding optimal $\boldsymbol{z}^*$ where $\boldsymbol{u}^* = M_v \boldsymbol{z}^*$. Notice when we have more neurons than the movement degrees of freedom, i.e., $n > d$, the equation $\boldsymbol{u}^* = M_v \boldsymbol{z}$ is under-determined. To find the optimal $\boldsymbol{z}^*$ given $\boldsymbol{u}^*$, we maximize the SNR. For system dynamics as given in Eqn. 4.19, when the signal $\boldsymbol{u}^*$ is given, to increase the SNR we must reduce the noise $\boldsymbol{\xi}$. Therefore, we want the magnitude of each neuron's firing rate $\boldsymbol{z}$, $\|\boldsymbol{z}^*\|^2$, to be as small as possible to keep the variance of the signal-dependent noise low. To compute the usability we need to take the expectation of the starting points and targets, which we will assume are symmetrically distributed, and we can assume the optimal intended pushes $\boldsymbol{u}^*$ are uniformly distributed on a unit circle (or, in 3D, a unit sphere). Taking the expectation on $\boldsymbol{u}^*$, we have the new optimization problems on M_v and $\boldsymbol{z}$ as

$$\begin{aligned} \min_{M_v, \boldsymbol{z}} \quad & \mathbb{E}_{\boldsymbol{u}^*} \|\boldsymbol{z}\|^2 \\ \text{s.t.} \quad & \boldsymbol{u}^* = M_v \boldsymbol{z} \\ & \|\boldsymbol{m}_{v,i}\| = 1 \quad (1 \leq i \leq n) \end{aligned} \tag{4.20}$$

The solution of $\boldsymbol{z}$ with minimal norm is given as $\boldsymbol{z}^* = M_v^T (M_v M_v^T)^{-1} \boldsymbol{u}^*$ and the magnitude of $\boldsymbol{z}^*$ is $\|\boldsymbol{z}^*\|^2 = \boldsymbol{u}^{*T} (M_v M_v^T)^{-1} \boldsymbol{u}^*$. Thus, we have

$$\begin{aligned} \min_{M_v} \quad & \mathbb{E}_{\boldsymbol{u}^*} \boldsymbol{u}^{*T} (M_v M_v^T)^{-1} \boldsymbol{u}^* \\ \text{s.t.} \quad & \|\boldsymbol{m}_{v,i}\| = 1 \quad (1 \leq i \leq n) \end{aligned} \tag{4.21}$$

A full derivation of the solution of this optimization problem is given in Appendix B.3. For $n = 2$, the only solution is that $\boldsymbol{m}_{v,1}$ and $\boldsymbol{m}_{v,2}$ are orthogonal. Denoting the angle of the i-th neuron's pushing direction as θ_i, it means the angle between θ_1 and θ_2 is $\pm 90^\circ$. For $n > 2$, one set of solutions to the above optimization problem is for θ_i to be uniformly (circularly) distributed on $[0, k \times 360^\circ)$, where k is any positive integer.

Those solutions are not the only solutions to Eqn. 4.21. However, since we only care about finding pushing directions that maximize the BCI system's usability, we are not interested in finding all the solutions. Therefore, for the general case in which there are more than 2 recorded neurons, to get the optimal BCI system we can set M_v to have uniformly distributed pushing directions for each neuron and the largest possible modulation depths consistent with the physiological range of the neurons.

4.4 Results

4.4.1 Simulation task

In this section we derive the optimal mapping for a 2D linear BCI used to perform point-to-point reaches and compare its expected performance relative to other 2D linear BCI mappings in simulation. For the movements we simulate an 8 target 2D center-out and back cursor movement task, similar to those performed in many BCI laboratories, e.g., [Fraser et al., 2009, Ganguly and Carmena, 2010, Hochberg et al., 2006]. Specifically, the subject is required to move a cursor in a 2D plane from origin $\boldsymbol{p}_0$ to target position $\boldsymbol{p}^*$ within T_r time steps, and hold the cursor at the target for another T_h time steps. Then the subject needs to move the cursor back to the origin within another T_r time steps and hold at the origin for another T_h time steps. There are 8 targets uniformly distributed on a circle with radius equal to 10cm. The reaching time T_r is set to 20 time steps (of 100ms) and the holding time is set to either 1 time step (no hold) or 20 time steps. For the signal-dependent noise, κ (from Eqn. 4.10) is set to 1 to mimic the Poisson-like distribution. For the signal-independent noise, the variance-covariance matrix is set as diagonal, i.e., $W = \sigma_\omega^2 I$ and $\sigma_\omega = 0.1(s^{-1})$.

We use the 2nd order linear physical system (Eqn. 4.9) as our decoder. The state $\boldsymbol{x}_t$ comprises the position $\boldsymbol{p}_t$ and the velocity $\boldsymbol{v}_t$. Also, to make this task solvable under the linear-quadratic framework, the state is augmented with the target position $\boldsymbol{p}^*$ as

$$\boldsymbol{x}_t = (\boldsymbol{p}_t, \boldsymbol{v}_t, \boldsymbol{p}^*)^T. \tag{4.22}$$

The system dynamics are written as

$$\begin{pmatrix} \boldsymbol{p}_{t+1} \\ \boldsymbol{v}_{t+1} \\ \boldsymbol{p}^* \end{pmatrix} = \begin{pmatrix} I & \Delta I & \mathbf{0} \\ H_p & H_v & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I \end{pmatrix} \begin{pmatrix} \boldsymbol{p}_t \\ \boldsymbol{v}_t \\ \boldsymbol{p}^* \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ M_v \\ \mathbf{0} \end{pmatrix} \left(\boldsymbol{z}_t + \sum_{i=1}^{n} \sqrt{\kappa_i} \epsilon_{i,t} I_i \boldsymbol{z}_t + \boldsymbol{\omega}_t \right), \tag{4.23}$$

and the cost function is

$$\begin{aligned} \mathcal{J} &= \sum_{t=T_r}^{T_r+T_h} \mathbb{E}\|\boldsymbol{p}_t - \boldsymbol{p}^*\|^2 + \lambda_u \sum_{t=0}^{T_r+T_h-1} \mathbb{E}\|\boldsymbol{u}_t\|^2 \\ &= \sum_{t=T_r}^{T_r+T_h} \mathbb{E}\|\boldsymbol{p}_t - \boldsymbol{p}^*\|^2 + \lambda_u \sum_{t=0}^{T_r+T_h-1} \mathbb{E}\boldsymbol{z}_t^T M_v^T M_v \boldsymbol{z}_t. \end{aligned} \tag{4.24}$$

The first term measures the movement accuracy during the holding period (between T_r and $T_r + T_h$), and the expectation is taken over the noise $\epsilon_{i,t}$ and $\boldsymbol{\omega}_t$. The second term measures the total effort required to finish the task (in this case, measured as the expected deviation in firing rate from the mean, or baseline, values). λ_u controls the balance between those two terms. In some approaches, a term measuring the accuracy of the velocity during the holding period, i.e., $\sum_{t=T_r}^{T_r+T_h} \mathbb{E}\|\boldsymbol{v}_t\|^2$, is also considered [Liu and Todorov, 2007, Todorov and Jordan, 2002]. This term requires the velocity during the holding period to be as close to zero as possible. While those terms are ignored here for simplicity, it is straightforward to extend our approach and include them.

To write the cost function in the quadratic form as Eqn. 4.11, we have

$$Q_t = \begin{cases} \begin{pmatrix} I & \mathbf{0} & -I \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -I & \mathbf{0} & I \end{pmatrix} & \text{for } T_r \le t \le T_r + T_h \\ \mathbf{0} & \text{otherwise,} \end{cases} \tag{4.25}$$

and

$$R_t = \lambda_u M_v^T M_v \qquad \text{for } 0 \le t \le T_r + T_h - 1. \tag{4.26}$$

4.4.2 Solution for H

We first look at BCI system usability as a function of different types of dynamics H (from Eqn. 4.7). Here we leave M_v fixed, and simulate 10 neurons with unit modulation depths and pushing directions uniformly distributed on the unit circle. To simplify parameters even further, we assume the H_p and H_v in Eqn. 4.23 are each proportional to the identity matrix, i.e., $H_p = h_p I(s^{-1})$ and $H_v = h_v I$. Physically, we know h_p represents the spring-like elastic effect and is 0 if there are no such effects. For example, the velocity Kalman filter [Hochberg et al., 2012, Kim et al., 2008] has $h_p = 0$. Viscous damping effects are captured by h_v, which takes on a value of 1 if there are no such effects. Common parameters for a Kalman filter often are often close to 0.6 - 0.8 (e.g., these were the parameters used in [Sadtler et al., 2014]). Since our goal is to improve on standard decoders, we search for optimal parameter values near those that are typically used. To do this, we initialize our gradient descent at a random guess between $[-0.5, 0.5]$ for h_p and $[0.5, 1.5]$ for h_v. In many of the analyses below, we conduct a brute force search over those values to characterize the cost landscape and assess the accuracy of our gradient descent procedure.

Fig. 4.3 summarizes BCI usability as a function of h_p and h_v when the subject is asked to perform the center-out out-center task with long target holds ($T_h = 20$). This plot shows the usability for the case when λ_u (the trade-off between accuracy in effort from Eqn. 4.24) is equal to 1. The colormap denotes the usability of the BCI system in question, with values ranging from red (least usable) to blue (most usable). For these settings, there is a clear, well defined maximum in the usability for a relatively simple system in which $h_p = 0$ and $h_v = 1$, denoted by a dot. Optimal trajectories for this system are shown in Fig. 4.4. For simplicity, only results for the movement towards the upper target are shown here; results for other movements are similar by symmetry. Several views of the cursor trajectories are provided, including cursor distance from the starting position, cursor velocity projected on the target direction, and the push effort

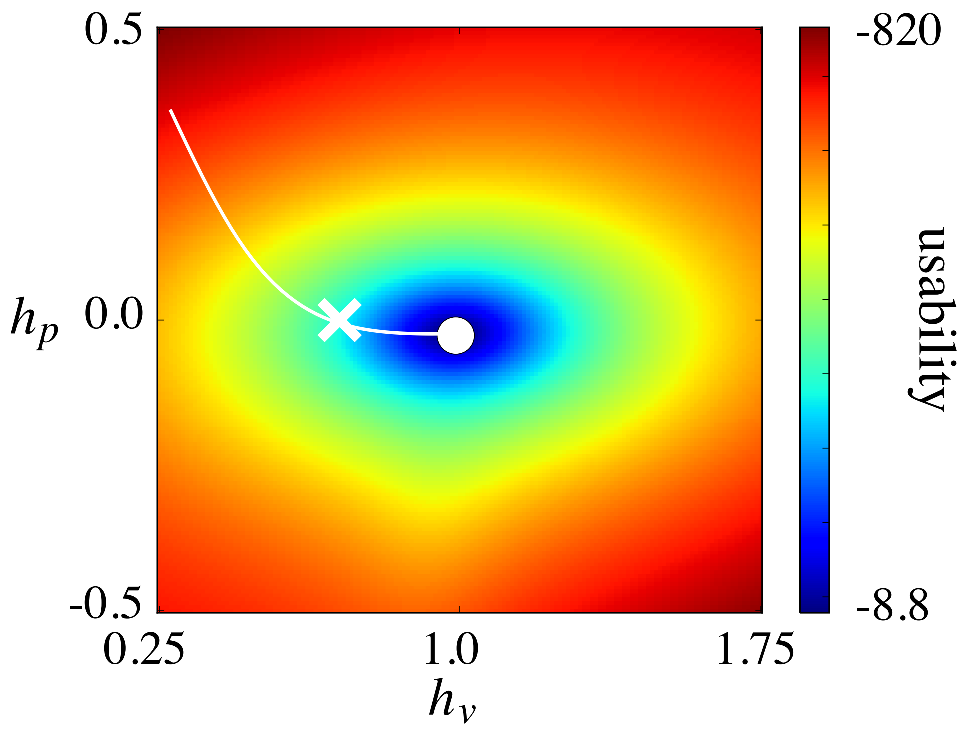
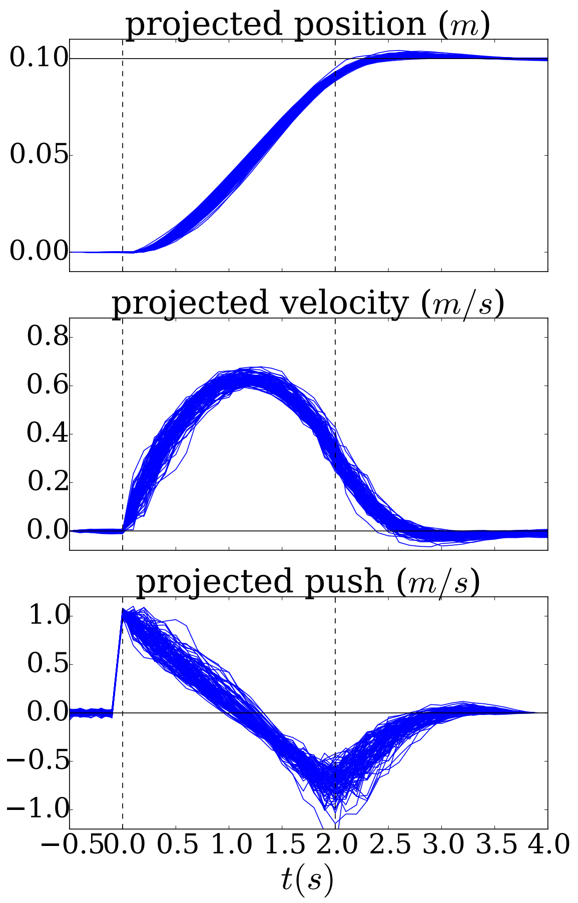


Figure 4.3: **BCI usability depends strongly on system dynamics parameters.** Usability is plotted as a function of h_p and h_v, with red values denoting the least usable systems and blue values denoting the most usable. The optimal BCI dynamics parameters are denoted by the dot, and the path for a single gradient ascent run of the algorithm is shown as the white line. Typical parameters of a velocity Kalman filter ($h_p = 0$ and $h_v = 0.75$) are denoted by the '$\times$', for reference. $\lambda_u = 1$ for this simulation.

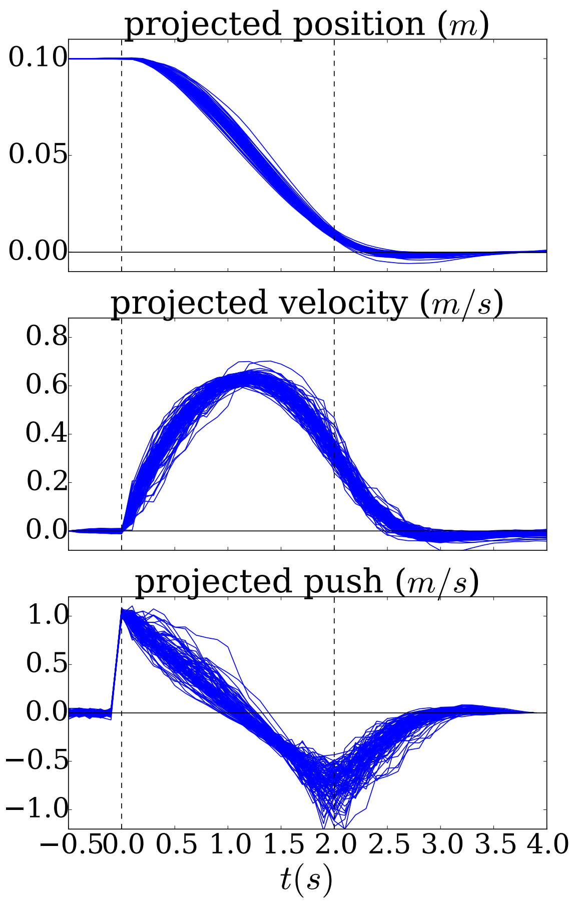
projected on the target direction. The noise on repeated identical trials to one target results from different instantiations of the signal-dependent and signal-independent noise terms from Eqn. 4.23.

For comparison, we can investigate the trajectories for a non-optimal system denoted by the '$\times$', in which $h_p = 0$ and $h_v = 0.75$ (Fig. 4.5). This is in the range of the typical parameters used for a velocity Kalman filter, one of the most ubiquitous BCI decoding algorithms in use today [Wu et al., 2006]. Comparing Figs. 4.4 and 4.5, we see that this non-optimal system requires a more sustained neural push to counteract the viscous damping terms, which typically results in noisier trajectories, as seen in the increased variance of the projected velocity traces. The average cost of this system is more than three times the cost of the optimal system.

We next explored the usability of BCI systems under other task conditions and effort/accuracy trade-offs (Fig. 4.6). The left column of Fig. 4.6 shows BCI usability for a short hold-time center-out/out-center task in which T_h was set to 1. The right column has the same results for the long hold-time task in which $T_h = 20$. This figure is organized just like Fig. 4.3, with red values denoting the least usable systems and blue values denoting the most usable systems. The colorbar to the right of each plot shows the range of usabilities for each condition. Note that a



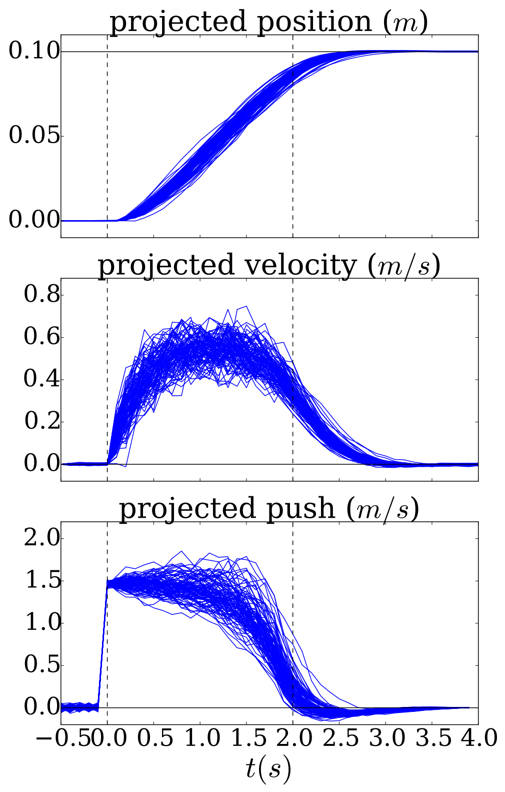
A. Center-Out



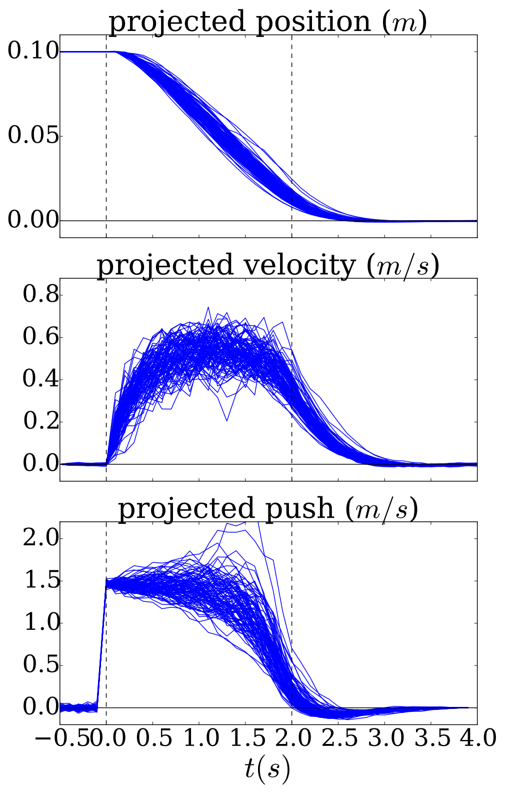
B. Out-Center

Figure 4.4: **Optimal movements of the most useable system.** Movement trajectories under optimal use of the most usable BCI system from Fig. 4.3 ($h_p = 0$ and $h_v = 1$). *Left*: Center-out movement to the upper target. *Right*: Out-center movement back from the upper target. *Top*: Position of the cursor along the y-axis. *Middle*: Velocity of the cursor along the y-axis. *Bottom*: Push effort $\boldsymbol{u}$ from Eqn. 4.18 along the y-axis

different range applies to each plot. Three plots are shown for each task, differing in the effort vs accuracy tradeoff. In the top row, λ_u takes on a relatively small value (0.01), favoring accuracy. In the bottom row, λ_u takes on a relatively large value (100), favoring effort. In the middle, row, λ_u takes on the intermediate value of 1. Note that the left middle plot is identical to Fig. 3, for reference. The usability maps are relatively smooth over this range of parameters, and the path for a single gradient ascent run of the algorithm is shown as the white line, which terminates at the optimal BCI dynamics values for that task (denoted by the dot). Surprisingly, we find that the typical values for velocity Kalman filter do not result in the most usable BCI under these simulation conditions. Here we use $h_p = 0$ and $h_v = 0.75$ to approximate the parameters of velocity Kalman filter and the locations in the figures are denoted by the '$\times$'. Across both long and short hold time tasks and a range of effort/accuracy trade-offs, the cost of the velocity



A. Center-Out



B. Out-Center

Figure 4.5: **Optimal movements of a system with typical parameters used for a velocity Kalman filter.** Movement trajectories under typical parameters used for a velocity Kalman filter ($h_p = 0$ and $h_v = 0.75$). *Left*: Center-out movement to the upper target. *Right*: Out-center movement back from the upper target. *Top*: Position of the cursor along the y-axis. *Middle*: Velocity of the cursor along the y-axis. *Bottom*: Push effort $\boldsymbol{u}$ from Eqn. 4.18 along the y-axis

Kalman filter is often an order of magnitude or more greater than the cost of the optimal plant. Another notable feature of these plots is the consistency in the optimal BCI parameters across tasks, for a large range of λ_u values. To further summarize these results, the optimal parameters for the short and long hold time task is shown as a function of λ_u in left and right columns of Fig. 4.7, respectively. For the long hold time task, the optimal h_p^* is slightly less than 0 and the optimal h_v^* is close to 1 for all λ_u. For the short hold time task, the optimal h_p^* increases from 0 to around 0.2 as λ_u increases and the optimal h_v^* increases from around 0.9 to around 1.15. It may at first seem rather surprising for the optimal h_v^* to be greater than 1, which would lead to an unstable equilibrium when holding the BCI cursor stationary. However, any extra effort exerted during the hold phase is made up for by the decrease in effort required when moving the cursor. In fact, as the ratio of time spent holding the cursor to the time spent moving the cursor increases, h_v^* decreases (compare the red lines in Fig. 4.7A and 4.7B).

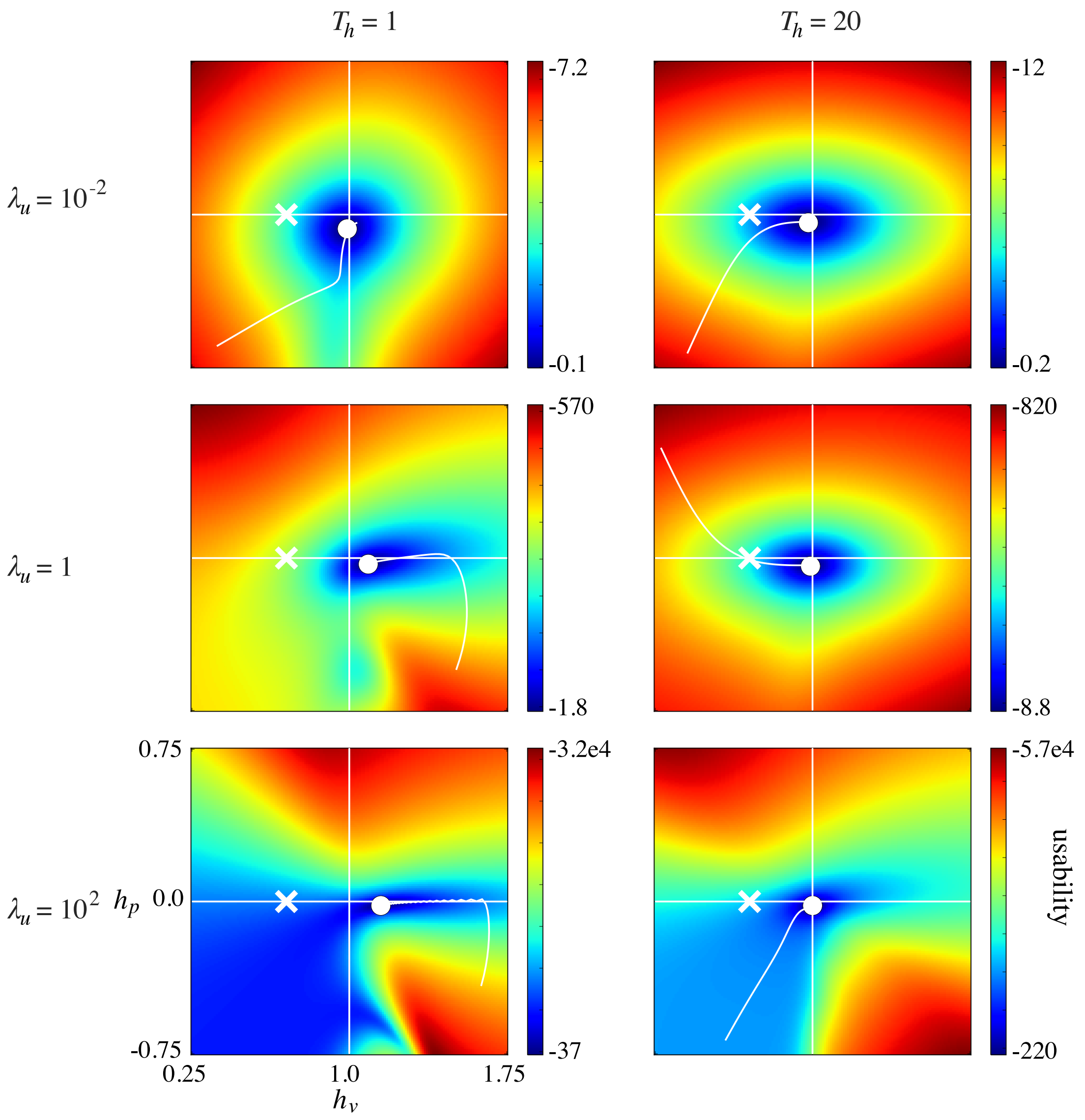
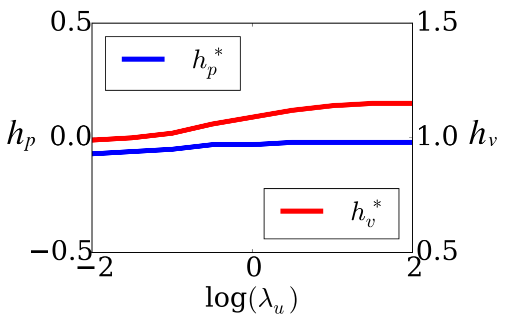


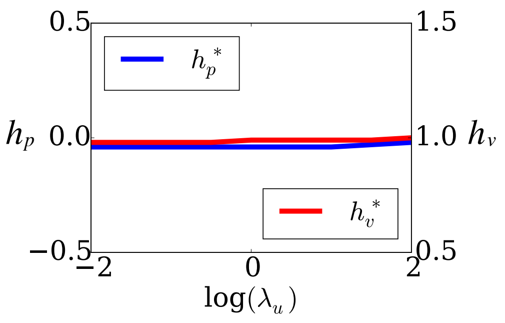
Figure 4.6: **BCI usability varies with task and energetics constraints.** The usability of a second order linear BCI is plotted as a function of dynamics parameters h_p and h_v under 6 conditions. The two columns are for two different center-out-back tasks, one with a short target hold requirement ($T_h = 1$; Left), and one with a long hold requirement ($T_h = 20$; Right). The three rows are for three different energetics constraints, where λ_u varies from 0.01 (Top) to 1 (Middle) to 100 (Bottom). The format of each colormap is the same as Fig. 4.3: the dot denotes the most usable parameter settings and the cross denotes the parameters of a typical velocity Kalman filter. For clarity, the horizontal line where $h_p = 0$ and the vertical line where $h_v = 1$ are also plotted. $\kappa = 1$ in all conditions.

4.4.3 Solution for H under a random target pursuit task

In the center-out/out-center task studied above in section 4.4.2, the total length of each trial is relatively short. The most usable BCI can exploit these short trial times. For example, when



A. $T_h = 1$



B. $T_h = 20$

Figure 4.7: **The post-learning optimal dynamics parameters are robust to energy/accuracy tradeoffs.** Optimal dynamics parameters h_p^* (blue) and h_v^* (red) are plotted as a function of λ_u, for both the short hold period task (A) and the long hold period task (B). These dynamics parameters show little change over several orders of magnitude of λ_u (note the logarithmic scale of the x-axis). $\kappa = 1.0$ for all simulations.

energetic costs are high and target hold times are small, the optimal dynamics parameter h_v is greater than 1, which could lead to unstable control for longer trial times. To explore these effects more thoroughly, we simulated a random target pursuit task [Flint et al., 2013, Hochberg et al., 2006, Kim et al., 2008]. In this task, the subject is required to move the cursor from the origin toward a target that is randomly placed on a 20cm×20cm screen within T_r time steps and hold there for another T_h time steps. After the holding period, the current target disappears and the next target immediately appears at another random location, keeping only one target at a time on the screen. The subject is then required to reach and hold at the new target. The central difference between the center-out task and the random target pursuit task is that, instead of resetting the position and the velocity at the beginning of each reach, the starting position and velocity is the ending position and velocity of the previous reach. The whole process repeats K times and the total cost is the sum of the cost of each reach.

For the k-th reach of the random target pursuit task, the cost $\mathcal{J}_k$ is similar to the center-out/out-center task, with an extra term of the penalty on the norm of the velocity during the holding time. Thus, if $\boldsymbol{p}_k^*$ denotes the location of the k-th target, the cost is

$$\mathcal{J}_k = \sum\nolimits_{t=T_r}^{T_r+T_h} \mathbb{E}\|\boldsymbol{p}_t - \boldsymbol{p}_k^*\|^2 + \lambda_v \sum\nolimits_{t=T_r}^{T_r+T_h} \mathbb{E}\|\boldsymbol{v}_t\|^2 + \lambda_u \sum\nolimits_{t=0}^{T_r+T_h-1} \mathbb{E}\|\boldsymbol{u}_t\|^2. \tag{4.27}$$

In the simulations described below, we set the reaching time T_r to 20 time steps (the same as for the center-out task), we set the holding time T_h to 1 time step for each reach, and we set the number of repeats K to 10. We analyzed various combinations of λ_v and λ_u, ranging from 10^{-4} to 1. For each set of parameters, we repeated the simulation 1000 times. The average cost of each set of simulations is plotted as a heat map in Fig. 4.8.

The cost heat maps reveal that the optimal h_p and h_v in the random target pursuit task are indeed smaller than those in the center-out task, as expected. The optimal h_p is always strictly negative (around -0.1) and the optimal h_v is always strictly less than 1 (around 0.8-0.9). Further, increasing h_v leads to a dramatic increase of the cost, indicating that large h_v will cause the

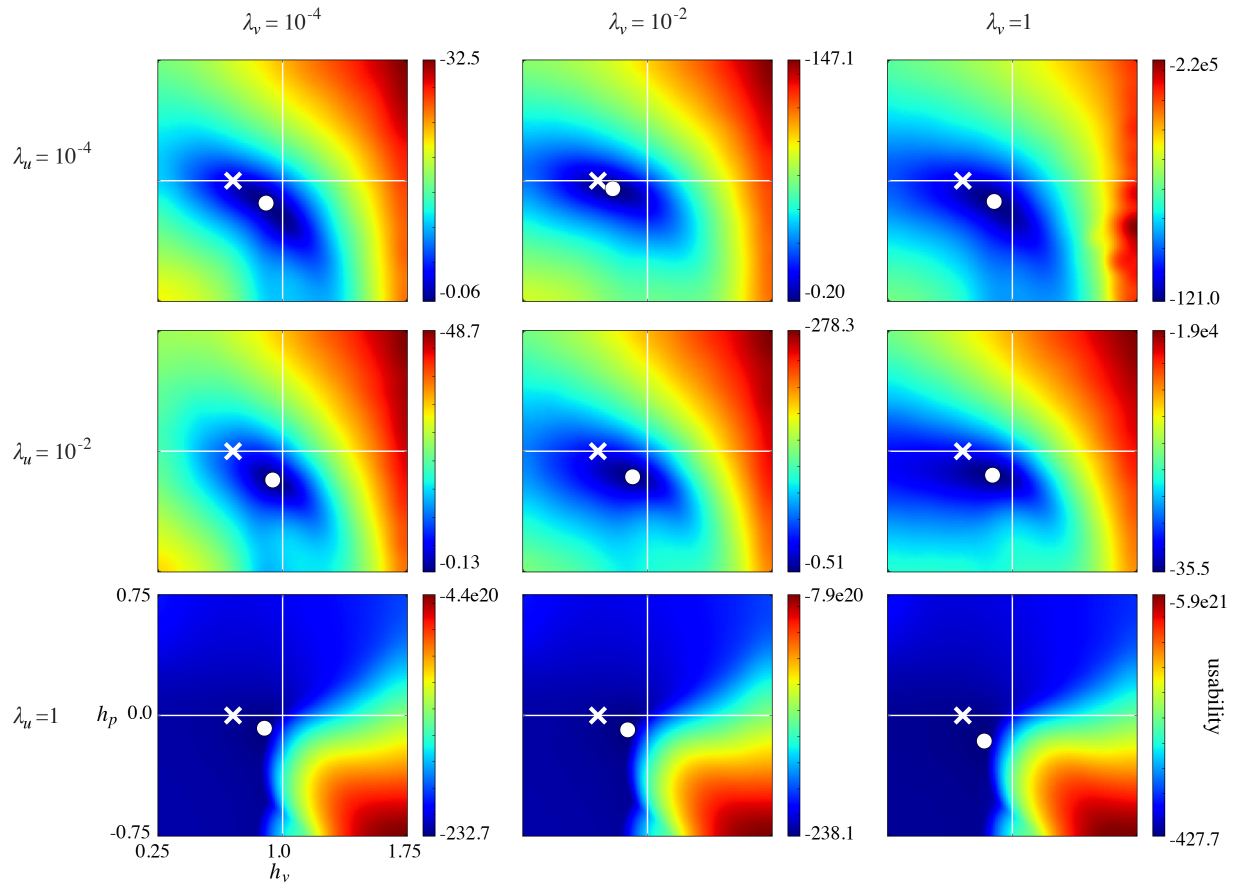


Figure 4.8: **BCI usability for the random target pursuit task.** The usability of a second order linear BCI is plotted as a function of dynamics parameters h_p and h_v under 3×3 combinations of λ_v and λ_u for the random target pursuit task. The format of each colormap is the same as Fig. 4.3: the dot denotes the most usable parameter settings and the cross denotes the parameters of a typical velocity Kalman filter. For clarity, the horizontal line where $h_p = 0$ and the vertical line where $h_v = 1$ are also plotted. $\kappa = 1$ in all conditions.

system to become unstable. Overall, these results indicate that the trial length (or the expected 'on' time of the BCI) should be taken into account when designing usable BCIs.

4.4.4 Comparison between the usability of the 1st order system and the 2nd order system

How does the usability of an optimally controlled 2nd order system compare to that of a 1st order system? In chapter 3, we argued that one explanation for why the VKF might outperform the OLE is that a 2nd order physical control system might simply be easier to control than 1st order physical control systems. In this section, we test this argument by computing the usability of an optimally controlled 1st order system and comparing it to the usability of an optimally controlled 2nd order system. We find that 2nd order systems are more usable than 1st order systems.

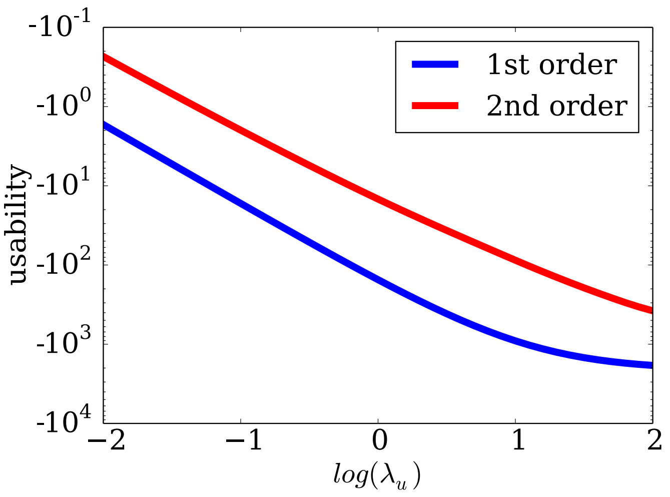


Figure 4.9: **The usability of the 1st order system and the 2nd order system under different λ_u.** Usability of the 1st order system (blue) and the 2nd order system (red) are plotted as a function of λ_u, for the center-out out-center task with the long hold period. The 2nd order system is always more usable than the 1st order system. $\kappa = 1.0$ for all simulations.

In the simulation, the subject performs a center-out / out-center task with long target holds ($T_h = 20$), just as in section 4.4.2. For the 1st order linear physical system, the state $\boldsymbol{x}_t$ comprises only the position $\boldsymbol{p}_t$. As for the 2nd order system, to make this task solvable under the linear-quadratic framework the state is augmented with the target position $\boldsymbol{p}^*$ as

$$\boldsymbol{x}_t = (\boldsymbol{p}_t, \boldsymbol{p}^*)^T. \tag{4.28}$$

The system dynamics are written as

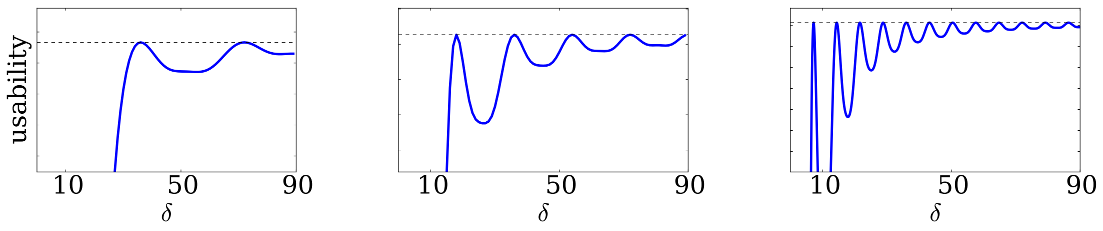
$$\begin{pmatrix} \boldsymbol{p}_{t+1} \\ \boldsymbol{p}^* \end{pmatrix} = \begin{pmatrix} H_p & \boldsymbol{0} \\ \boldsymbol{0} & I \end{pmatrix} \begin{pmatrix} \boldsymbol{p}_t \\ \boldsymbol{p}^* \end{pmatrix} + \begin{pmatrix} M_v \\ \boldsymbol{0} \end{pmatrix} \left(\boldsymbol{z}_t + \sum_{i=1}^{n} \sqrt{\kappa_i} \epsilon_{i,t} I_i \boldsymbol{z}_t + \boldsymbol{\omega}_t \right). \tag{4.29}$$

To compare the usability across systems with different orders, the cost function is the same for both systems, and is given as Eq. 4.24. We sampled a range of energy / accuracy trade-offs in the cost function by by changing λ_u from 10^{-2} to 10^2. κ is set to be 1 for all cases. Under all values of λ_u, the 2nd order system was more usable than the 1st order system (Fig. 4.9).

4.4.5 Solution for M

We now switch our attention to optimizing the matrix mapping neural activity to cursor movement, M. To verify our conclusion that the optimal pushing directions are uniformly distributed on a circle, we compare the optimal costs under different pushing direction distributions. In the following center-out / out-center simulations, H is fixed with $h_p = 0$ and $h_v = 1$, λ_u is set to be 1, and κ is also set to be 1. We also set $\|\boldsymbol{m}_{v,i}\|$ to be 0.01, which has the effect of restricting the mean-centered firing rates to range roughly between -40 and 40 spikes/s.

First we consider a special distribution where angles between two consecutive pushing directions are constant. Denoting the constant angle as δ, we know from our theoretical result



A. 10 neurons B. 20 neurons C. 50 neurons

Figure 4.10: **Usability depends on the uniformity of pushing directions.** Usability is plotted as a function of the angle between two consecutive pushing directions, delta, for BCIs controlled by 10 (A), 20 (B), or 50 (C) neurons.

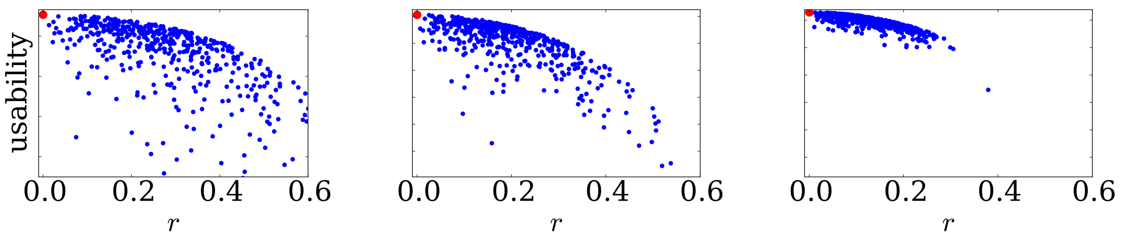
that the optimal δ should be $360°/n$. The optimal cost as a function of δ is shown in Fig. 4.10. For 10 neurons, minimum cost values are found at $\delta = 36°$ and $72°$, which both correspond to cases where the pushing directions are uniformly distributed on the unit circle (Fig. 4.10A). For 20 neurons, cost minima occur at $\delta = 18°, 36°, 54°, 72°, 90°$, again corresponding to cases in which the pushing directions are uniformly distributed (Fig. 4.10B). This answer also holds for 50 neurons (Fig. 4.10C).

To evaluate more random distributions, we randomly generated n pushing directions within $[0°, 360°)$ and use the *vector norm* as a measure of their uniformity. The vector norm, denoted as r, is defined as the normalized norm of the sum of the pushing directions in vector form, i.e,

$$r = \frac{1}{n}\sqrt{\left(\sum_{i=1}^{n} \cos(\theta_i)\right)^2 + \left(\sum_{i=1}^{n} \sin(\theta_i)\right)^2}. \tag{4.30}$$

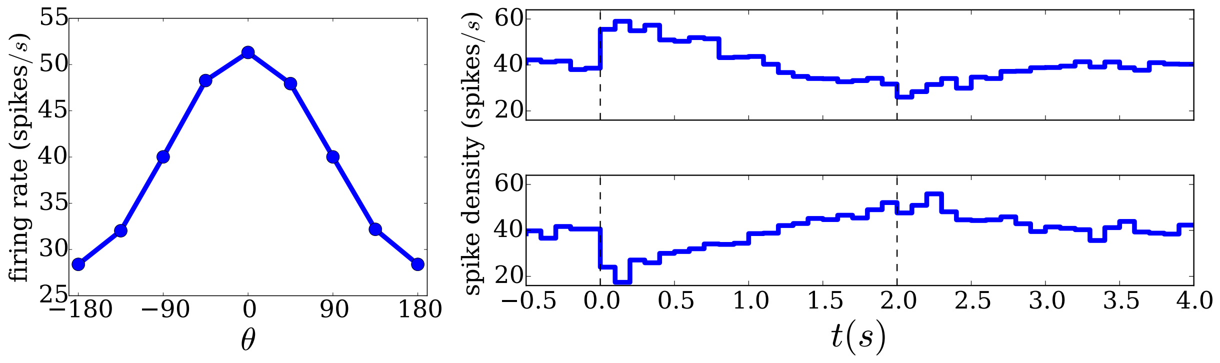
The vector norm r ranges from 0 to 1. When the pushing directions are uniformly (circularly) distributed on $[0°, k \times 360°)$, r equals 0, and when the pushing directions are all the same, r equals 1. We computed the cost of these random distributions and compared to the uniform distribution, and repeated with 500 different random draws for neuron counts $n = 10$, 20, and 50, and plotted the cost of each random distribution as a function of r (Fig. 4.11). The uniform distribution corresponds to the lowest optimal cost.

Finally, to gain an intuition for how neurons behave when driving the optimal plant, we show an example of how firing rates evolve during optimal control of an optimal plant. The optimal plant we simulate has 10 neurons with uniformly distributed pushing directions, assumes $\lambda_u = 1$, and has dynamics terms $h_p = 0$ and $h_v = 1$, performing the center out task with hold time $T_h = 20$ ($2s$). We assumed the minimum firing rate was 0 and the maximum firing rate was around 80 spikes/s. In Fig. 4.12A, we plot the tuning curve of the neuron (its average firing rate over the first $1s$ of the reach as a function of target direction), for the neuron whose pushing direction is at $0°$. Note the emergence of cosine tuning, a ubiquitous feature of motor encoding [Georgopoulos et al., 1982]. In Fig. 4.12B, we plot the peristimulus time histogram (PSTH) for repeated reaches to the target at $0°$ (top) and $180°$ (bottom). The temporal profile of the PSTH is a scaled version of the projected push shown in Fig. 4.4A (bottom). For parameters corresponding to the velocity Kalman filter, these neurons will show a more extended period of firing over the duration of the reach. Beyond the cosine tuning property, it is difficult to compare the simulated



A. 10 neurons B. 20 neurons C. 50 neurons

Figure 4.11: **Usability increases for more uniform distributions.** Usability is plotted as a function of the uniformity measure, r, from Eqn. 4.30. The red dot indicates the case where neurons' pushing directions uniformly distributed on a circle and the blue dot indicates the case where neurons' pushing directions are randomly generated.



A. tuning curve B. PSTH

Figure 4.12: **Tuning curve and PSTH.** The tuning curve is plotted of the neuron's average firing rate over the first half of the reach as a function of the target direction (A). The neuron's pushing direction is at $0°$. The peristimulus time histogram (PSTH) for repeated reaches to the target at $0°$ (top) and $180°$ (bottom) is plotted as a function of time (B).

firing rates with the firing rates from real experiments. The reason is that here we assume the subject has fully leant how to used the system which is hard to verify in the real experiments.

4.5 Discussion

We have developed an approach to BCI design that attempts to take user learning into account to create an optimal BCI, post-learning. Using ideas from optimal control theory, we have presented a rigorous definition for the usability of a BCI, and showed that this usability can be optimized as a function of plant parameters. We have applied this system to the design of a second order linear BCI control system, as is often used for controlling cursors on computer screens. The post-learning optimal BCI is surprisingly simple: it has little to no elastic or viscous components, and requires that the recorded neurons have uniformly distributed pushing directions that utilize their full dynamic firing rate range. Surprisingly, we find under these quadratic cost assumptions that typical implementations of second order linear BCIs, like the velocity Kalman filter, are not expected to provide the greatest usability after learning has occurred. Of course, we made a number of assumptions in developing our BCI optimization approach, which we discuss below in turn. Until these assumptions are proven true, or the optimization framework amended to take deviations of these assumptions into account, closed-loop experiments should always be the final arbiter of which control system might work best [Chase et al., 2009].

4.5.1 Can the optimal BCI be learned?

To solve for the optimal BCI post-learning, we made the strong assumption that the subject could learn to produce the optimal control signals for any given BCI mapping. It is unknown to what extent this might be true. Several groups have demonstrated the impressive ability of subjects to learn non-intuitive mappings between motor control signals and cursor movements. For example, subjects can learn a host of mappings between joint angles recorded with a cyber glove and cursor movements [Liu and Scheidt, 2008, Liu et al., 2011, Mosier et al., 2005]. They can also learn to control a muscle computer interface, where recorded EMG activity is used to directly drive a cursor, with arbitrary mappings between muscle activity and movement [Arduin et al., 2013, Radhakrishnan et al., 2008]. Further, experiments by Eb Fetz and others have demonstrated that subjects can easily learn through operant conditioning to volitionally modulate most arbitrarily chosen single neurons [Arduin et al., 2013, Chase and Schwartz, 2011, Fetz, 1969, Moritz and Fetz, 2011, Wetzel, 1986]. However, on short time-scales it has been found that BCI mappings that are consistent with the existing correlation structure of neural population activity are easier to learn than those that are not [Sadtler et al., 2014], and it has been suggested that learnable patterns may be limited to the natural motor repertoire [Hwang and Andersen, 2013]. Further, when learning does occur it is not known whether it proceeds to the optimal end point or asymptotes at some less behaviorally optimal point. Finally, it is worth noting that, to date, non-biomimetic BCI decoders have not been shown to outperform their biomimetic counterparts [Shenoy and Carmena, 2014]. How, then, might an optimal BCI be learned?

Learning the optimal BCI would likely be a long-term process, occurring over days or weeks. Long-term learning does not appear to be subject to the same correlational constraints as short-term learning. Ganguly and Carmena have shown that subjects could vastly improve their performance when using a decoder with arbitrarily scrambled parameters, provided they had several days to practice [Ganguly and Carmena, 2009]. This long-term learning may rely on a form of “credit-assignment” learning that proceeds independently across neurons [Jarosiewicz et al., 2008] but explains only a small amount of the overall learning that occurs on short time-scales [Chase et al., 2012]. There are hints that this type of learning may continue to grow with long-term practice, enabling long-term neural reorganization that is unconstrained by the correlation structure measured on day 1 [Oby et al., 2015, Zhou et al., 2015]. Still, it remains to be seen whether these mechanisms could truly result in optimal control signals as defined in this work. If learning is constrained, the optimization problem outlined in Eqn. 4.6 would have to be reformulated as a constrained optimization problem that ensures the obligatory correlations across neurons are maintained. Further work investigating the optimality and limitations on learning will need to be done to determine the extent to which any arbitrary mapping might be learned.

4.5.2 Can the motor system be treated as an optimal control system?

Our approach to optimizing BCI usability relies on the assumption that the brain acts as an optimal controller, i.e., that it produces control signals that are "optimal" in a statistical sense in that they minimize some cost function [Scott, 2012, Todorov, 2004]. It is not yet known whether or not this is the case. While many studies make assumptions that the cost function is linear and quadratic, as we have done, the precise nature of the cost function is not known, and certain adaptive responses have been found that deviate from expected predictions of optimal control theory. For example, de Rugy and colleagues demonstrated that although subjects were capable of adapting to the various perturbations they applied, the resulting patterns of muscle activation were not structured to yield the lowest overall force activations still consistent with task success [de Rugy et al., 2012]. In fact, it is entirely possible that the brain settles on a strategy that is instead "good enough", i.e., that achieves performance that is useful on average but not necessarily optimal [Loeb, 2012]. The extent to which the brain may deviate from optimality, or how the brain might determine when behavior is "good enough", is an active area of research.

We chose a specific form of the cost function for our simulations, which we implemented as a finite horizon cost function with quadratic penalty terms on positional accuracy and overall energy (as measured by the magnitude of the control signal). We chose this form of the cost function for two reasons: it is a commonly assumed cost function in motor control (e.g., [Diedrichsen, 2007, Shadmehr and Krakauer, 2008, Todorov and Jordan, 2002]), and the forward control problem can be solved as a relatively straightforward extension of the well-known LQR problem [Kwakernaak and Sivan, 1972]. There are three aspects of this cost function worth discussing: the finite horizon, the accuracy constraint, and the energy constraint. The finite horizon refers to our task implementation, which required the subject to arrive on target at a specific time and hold for another specific period of time. Self-paced movements do not reveal such a sharp cutoff: movement speed is typically a complex function of effort and reward [Shadmehr et al., 2016]. We found that typical values of h_v became smaller when hold times became longer (Fig. 4.7), as well as when trial times became longer (as in the random target pursuit simulation of Fig. 4.8). However, all of the optimal h_v values found across our simulations were larger than the typical Kalman filter implementation used in practice. To perform a more complete optimization that takes a distribution of trial times and hold times into account, one would need to reformulate the simulations. This could be done, for example, by randomly sampling from a distribution of trial and hold times and taking the average cost across runs. The quadratic accuracy constraint we assumed is a reflection of our BCI task design, but to be more realistic the constraint ought to be binary: one either hits the target or one does not. In natural motor control, movements typically respect discontinuous boundaries by building a "buffer region" into the motor endpoint that can vary with the configuration of the failure costs [Trommershäuser et al., 2003]. It remains to be seen whether a discontinuous implementation of the accuracy cost would substantially affect our derivation. The quadratic energy constraint is commonly used in optimal control models of arm movements for minimizing endpoint variance and effort [Diedrichsen et al., 2010, Fagg et al., 2002]. Very recent results hint that this constraint may not apply to neural firing, at least for short term exposure to a particular mapping [Hennig et al., 2017]. We attempted to mitigate this choice by exploring a range of trade-offs between accuracy and energy (the λ_u parameter in Eqn. 4.24). Ultimately, future work will be required to detail the appropriate cost functions for particular motor control problems.

4.5.3 Is the optimal linear decoder biological plausible?

In solving for the post-learning optimal linear decoder, we made a number of simplifying assumptions about the structure of motor cortical activity to balance mathematical tractability with biological plausibility. Below, we briefly discuss these assumptions and their potential impact on our conclusions. One assumption we made was that neural activity could be independent from time-point to time-point. This may not be realistic, as it has been shown that there is substantial structure in the time course of motor cortical activity [Churchland et al., 2012, Kao et al., 2015]. That said, with the exception of the onset transient, we don't find many temporal discontinuities in our simulated firing rates (Fig. 4.12B). Further, it is worth noting that onset rates of motor cortical neurons can also be quite rapid (see, e.g., Fig. 1D of [Churchland and Shenoy, 2007]). Our temporal independence assumption is also unlikely to reproduce the rotational dynamics that have been measured in populations of motor cortical neurons [Churchland et al., 2012]. However, it is currently unknown whether these dynamics are an obligatory feature of motor cortical firing or whether they can be unlearned. The neural activity will also not express the same underlying co-fluctuation structure that can be captured using dimensionality reduction techniques such as factor analysis [Sadtler et al., 2014, Yu et al., 2009]. One way to reproduce these sorts of constraints would be to add correlated noise to the neural firing rates, as opposed to the independent noise we added through Eqn. 4.23. This would not impact the derivation of the optimal dynamics terms h_v^* and h_p^*, but would likely change the optimal control terms M. However, this assumes that correlated noise is obligatory within the nervous system that cannot be changed even with extended practice. Further work will be required to determine how realistic these constraints are and to determine whether they would limit BCI usability.

Finally, other than using signal-dependent noise and restricting the firing rates to a physiologically reasonable range, we made little effort to build in well-known features of motor tuning into our neural simulations. Some of these features, like cosine tuning to direction [Georgopoulos et al., 1982], emerge from our implementation (Fig. 4.12). Other features may deviate from measured properties of neurons. As one example, we did not build any control delays into our system. We feel this is unlikely to have much impact on the results, as in a well-learned system there is evidence that subjects compensate for latencies in ongoing control [Golub et al., 2015]. As another example, we assumed each neuron contributed linearly to the control input $\boldsymbol{u}$ used to drive the BCI, even though it is not unreasonable to suppose that a biomimetic control signal would be represented by a non-linear transform of neural activity. However, building in this non-linear relationship is akin to assuming that it is an obligatory feature of neural encoding that cannot be unlearned, which may not be correct (see, e.g., [Taylor et al., 2002]). We also did not implement any influence of sensory-driven responses that might be unrelated to the volitional movement, even though accounting for these has been shown to improve BCI performance [Gilja et al., 2012]. Again, further work will be required to determine which features of neural encoding are obligatory, so that BCI usability may be optimized in light of these constraints.

4.6 Conclusion

Typical BCI systems attempt to maximize the biomimetic properties of the device, to limit the need for extensive training. However, it is unclear if this approach would ultimately be superior to performance that might be achieved with a non-biomimetic device, once the subject has engaged in extended practice and learned how to use it. In this chapter, we approach this problem using ideas from optimal control theory. Under the assumption that the brain acts as an optimal controller, we present a formal definition of the usability of a device, and show that the optimal, post-learning mapping can be written as the solution of a constrained optimization problem. We then derive the optimal mappings for particular cases common to most brain-computer interfaces. Our results suggest that the common approach of creating biomimetic interfaces may not be optimal when learning is taken into account. More broadly, our method provides a blueprint for optimal device design in general control-theoretic contexts.

Chapter 5

Modeling the BCI learning process

In the previous chapter, I proposed a way to design a BCI decoder which provides the optimal usability when subjects get familiar with the system and learn to control it in an optimal way. The derived BCI system is no longer optimal if the subject has not fully learned to control it. Actually, learning is necessary not only for decoders designed from the control system perspective but also for those decoders which aim to provide bio-mimic control. For those decoders, due to limitations in the number of neurons recorded and our understanding of the neural encoding process, subjects still likely could not control those BCI systems smoothly without some amount of learning. Therefore, successful control of the BCI system will depend critically on the subject's ability to learn how to produce the appropriate neural activity patterns [Orsborn and Pesaran, 2017, Shenoy and Carmena, 2014]. Learning is possible in closed-loop control due to the feedback about the discrepancy between the subject's intended movement and the actual movement the BCI system generates. Feedback about these errors can potentially help the subject correct poor estimates of intention on-line [Chase et al., 2009, Golub et al., 2015, Koyama et al., 2010a] and can be used to update the subject's knowledge of the BCI system. In this chapter, I take a first step towards studying how a BCI system might be learned by the subject. This work has been published as [Zhang et al., 2012].

5.1 Introduction

Successful implementation of a brain-computer interface depends critically on the subject's ability to learn how to modulate the neurons controlling the device. However, the subject's learning process is probably the least understood aspect of the control loop. How should training be adjusted to facilitate dexterous control of a prosthetic device? An effective training schedule should manipulate the difficulty of the task to provide enough information to guide the improvement without overwhelming the subject. In this chapter, we introduce a Bayesian framework for modeling the closed-loop BCI learning process that treats the subject as a bandwidth-limited communication channel. We then develop an adaptive algorithm to find the optimal difficulty-schedule for performance improvement. Simulation results demonstrate that our algorithm yields faster learning rates than several other heuristic training schedules, and provides insight into the factors that might affect the learning process.

In closed-loop control, poor estimates of intention can potentially be corrected on-line by the subject. However, to know the appropriate corrective action, the subject has to already have some facility at control. If the subject has no understanding of the mapping between neural activity and effector movement, even the corrective movement will likely be wrong. In practice, computer assistance is often used at the beginning stages of training to facilitate this learning process. Without any assistance, the errors that the subject makes may be too big for the subject to figure out how to compensate, which may not only make the task too difficult to learn, but may also frustrate the subject. Using a proper amount of computer assistance, the errors can be reduced to a certain degree which will not be too difficult for the subject to learn, but which also allow the subject to gain enough information about the task. In this chapter, I will take one specific type of computer assistance, shared-mode control (SMC), which was developed by Andrew Schwartz's lab [Taylor et al., 2002, Velliste et al., 2008], as an example to study how we should assist the subject so that it can learn as fast as possible. The basic idea of SMC is that, instead of moving the effector in the direction decoded from the subject's neural activity, the actual movement is the decoded movement corrected by some factor. By limiting the errors that the subject produces, SMC increases motivation and keeps the subject engaged in the learning process. Intuitively, at the beginning of training, more computer intervention is needed since the decoded movement will be quite different from the intended movement. As learning takes place, the amount of computer intervention will be less and less and at the end, SMC will be completely removed and the subject will fully control the BCI system.

Since SMC provides a means of directly manipulating the difficulty of the task, how should this assistance be scheduled to maximize learning rate? If the difficulty is too low, the errors will be artificially small and the subject will have no pressure to learn. If the difficulty is too high, the errors may be too large to be meaningfully interpreted, and the learning rate may be small as a consequence. The task difficulty must be carefully titrated to the subject's ability to promote rapid learning. Here we introduce a Bayesian framework for modeling a closed-loop BCI learning process that incorporates SMC. By treating the subject as a bandwidth-limited communication channel, we demonstrate an explicit link between the difficulty-schedule and the learning rate. We then develop an adaptive algorithm to find the optimal difficulty schedule for performance improvement. In simulation, our adaptive difficulty-control strategy promotes a marked improvement in learning rate.

In the following chapter, I will first introduce mathematically formalize the idea of SMC (section 5.2). Then in section 5.3 I will present a Bayes learning framework to capture the subject's learning process, and also an adaptive difficulty control algorithm to maximize learning rate. Then, in section 5.4, the proposed difficulty schedule is evaluated by running some simulations. Finally, I conclude this chapter in section 5.5.

5.2 Computer assistance: shared-mode control (SMC)

Shared-mode control (SMC) provides the subject assistance by mixing the subject's volitional signal with a computer-generated "correct" signal to generate the final output. Therefore, the error the subject makes is alleviated by a certain degree which is controlled by a single parameter λ. In this section, I will introduce the formal definition of SMC in the context of the center-out

Table 5.1: **Notations.**

θ^*	the system's parameter the subject need to learn
θ_{t-1}	the subject's guess about the system parameter at the beginning of the t-th step
ψ_t^*	the cursor's desired movement direction at the t-th step
φ_t	the system's input, i.e., the subject's aiming direction at the t-th step
ψ_t	the system's output, i.e., the cursor's actual movement direction at the t-th step
$\tilde{\psi}_t$	the subject's perception about ψ_t at the t-th step
λ_t	the weight of the linear combination in SMC at the t-th step

reaching task and take a look at how SMC assists the subject's learning. In this chapter, since we are studying the center-out reaching task with 1D kinematics, the notation is a little different from other parts of this thesis. All the variables used in this chapter are listed in Table 5.1 for reference.

In the center-out reaching task, the subject will control the BCI system to move a cursor from the centered home position to a predefined target. For simplicity, in this chapter, the cursor's movement is quantified by its direction, a 1D angle ψ, and the BCI system's mapping function f is assumed to directly decode from the subject's aiming direction, φ. Notice, in practice we can only decode from the subject's neural activity, which is generated from the intended movement by the encoding process. However, in this chapter, instead of studying the decoding process, I am focusing on studying the subject's learning process and how to design an optimal difficulty schedule. To create some learning pressure, we here assume that the decoder is wrong and biased. This is modeled as a control system mapping function f that outputs a movement direction that is rotated from the aiming direction φ by θ^*, i.e.,

$$\hat{\psi} = f(\varphi;\, \theta^*) = \varphi + \theta^* \tag{5.1}$$

where $\hat{\psi}$ is the decoded movement direction and θ^* is the system's parameter. The ideal system which requires no subject learning is the one which can perfectly recover the subject's aiming direction, i.e., $\theta^* = 0$. When $\theta^* \neq 0$, the subject's learning process is to find a "re-aiming" direction that compensates for this bias.

At the beginning when the subject is just starting to use the BCI system, the subject will not be familiar with the system bias. Therefore, it would be difficult for the subject to control the BCI system smoothly without any learning. To assist the subject's learning, under SMC, the external device's actual movement direction is the decoded output corrected by a factor λ. To do that, notice in the center-out reach task, the subject is required to do some predefined movement. Therefore, at any time we know that the ideal movement direction ψ^* points from the device's current position to the target, and we assume this is the movement the subject wants to exert. We can decompose the subject's decoded movement direction into two parts: the part that is correct, and points at the target, and the error component,

$$\hat{\psi} = \psi^* + (\hat{\psi} - \psi^*). \tag{5.2}$$

The first term is the ideal movement direction the subject intends to make while the second term is the error the subject made. SMC attenuates the error term by a scalar factor λ that is less than or equal to 1. Concretely, the real movement direction after SMC is

$$\begin{aligned}\psi &= \psi^* + \lambda(\hat{\psi} - \psi^*) \\ &= \lambda\hat{\psi} + (1-\lambda)\psi^* \\ &= \lambda f(\varphi;\ \theta^*) + (1-\lambda)\psi^*.\end{aligned} \tag{5.3}$$

From the above equation we can see the real movement direction under SMC is actually a linear combination of the decoded output $\hat{\psi}$ and the correct movement direction ψ^*. When $\lambda = 1$, we have $\psi = \hat{\psi} = f(\varphi;\ \theta^*)$ which means there is no assistance at all and the subject fully controls the device's movement. When $\lambda = 0$, we have $\psi = \psi^*$ which means the subject has no control on the system and the device will always move in the ideal direction. Therefore, λ reflects the difficulty of the task and our purpose is to adjust λ adaptively to facilitate the subject's learning.[1]

5.3 The Bayes learning framework

To model the subject's learning process about the system parameter θ^*, here we propose a Bayes learning framework. Under this framework, the model subject's knowledge of the system parameter θ^* as a random variable θ with probability density function $p(\theta)$. At the beginning of the learning task, $p(\theta)$ should be fairly flat because the subject is quite uncertain about the system. The goal of training is for the subject to gain veridical knowledge of the system, i.e., to build a distribution that is shapely peaked around the ground truth system parameter θ^*.

If the subject aims straight at the target (i.e., $\varphi = \psi^*$) and SMC is off, the device movement direction will be $\psi^* + \theta^*$ from Eqn. 5.1. Therefore, the error the subject makes by controlling the BCI system is θ^*. In order to compensate for this error, the subject may adjust the aiming direction φ from ψ^* to $\psi^* - \theta$. As a result, the decoded movement direction becomes

$$\begin{aligned}\hat{\psi} &= f(\varphi;\ \theta^*) \\ &= f(\psi^* - \theta;\ \theta^*) \\ &= \psi^* - \theta + \theta^*.\end{aligned} \tag{5.4}$$

When the re-aiming direction balances the system bias θ^*, the decoded movement direction becomes the correct direction. According to the experiments conducted by Jarosiewicz et al. [2008], not just the intended movement, but also the encoding function can also be modified to compensate the system bias. Here, for the demonstration simplicity, we assume all the compensation comes from the subject's adjustment of the intended movement. This has been found to be the dominant form of learning over short timescales of an hour or so [Chase et al., 2012, Golub et al., 2018].

The decoder bias θ^* is hidden from the subject. Therefore, the subject's leaning process is essentially a system identification process where the subject needs to find θ^* through a series of observations about the system's inputs and outputs. Notice that because of SMC, the system's output is the actual movement direction ψ and the decoded movement direction $\hat{\psi}$ is hidden from

[1] Actually λ is the percentage of task difficulty, not the absolute amount. The absolute task difficulty is determined by the experimental design.

the subject. From the subject's viewpoint, the actual movement direction ψ is the decoded result, even when SMC is applied for assistance and the actual movement direction is different from the decoded one.

5.3.1 One step updating

To model the subject's learning process, we will first divide the task into consecutive steps. At each step, the subject will try to gain some knowledge from the error he/she made and update his/her knowledge about the system's parameter. Formally, at time t, to produce a desired movement direction ψ_t^*, the subject needs to modulate the aiming direction φ_t so that the decoded direction $\hat{\psi}_t = f(\varphi_t;\ \theta^*)$ should be as close to the desired direction ψ_t^* as possible.

Our model of the learning process proceeds as follows. At the beginning of the t-th step, the subject first generates a guess θ_{t-1} about the system parameter θ^* from his current knowledge $p_{t-1}(\theta) = \mathcal{N}(\mu_{t-1}, \sigma_{t-1}^2)$ by randomly sampling from his/her knowledge distribution $\theta_{t-1} \sim p_{t-1}(\theta)$. Then the subject adjusts the aiming direction to make the device move along ψ_t^* under the assumption that the system's parameter is this random guess θ_{t-1}, as

$$\varphi_t = \psi_t^* - \theta_{t-1}. \tag{5.5}$$

From Eqn. 5.4, the decoded movement direction $\hat{\psi}_t$ will then be

$$\hat{\psi}_t = f\left(\varphi_t;\ \theta^*\right) = \psi_t^* - \theta_{t-1} + \theta^*. \tag{5.6}$$

Under SMC, the external device's actual movement direction is the linear combination of the decoded direction $\hat{\psi}_t$ and the correct direction ψ_t^*, i.e.,

$$\begin{aligned} \psi_t &= \lambda_t \hat{\psi}_t + (1 - \lambda_t)\psi_t^* \\ &= \lambda_t f(\varphi_t;\ \theta^*) + (1 - \lambda_t)\psi_t^* \\ &= \psi_t^* + \lambda_t(\theta^* - \theta_{t-1}). \end{aligned} \tag{5.7}$$

At the end of the t-th step, the external device moves to a new position determined by ψ_t.

The subject's perception about the external device's movement direction will deviate from the real direction due to noise, and here we treat the perception, $\tilde{\psi}_t$, as a noisy version of ψ_t, i.e.,

$$\tilde{\psi}_t = \psi_t + \varepsilon_{\psi_t} \tag{5.8}$$

where $\varepsilon_{\psi_t} \sim \mathcal{N}(0, \sigma_{\psi_t}^2)$ models this perceptual noise. This noise will turn out to be critical to understanding why SMC might actually hasten learning. This is discussed in more detail in Sec. 5.3.2. The input-output pair $\{\varphi_t, \tilde{\psi}_t\}$ that is, the movement the subject expected the device to make, versus the movement the device actually made, provides new information about the system so that the subject can update his/her knowledge.

We assume the subject updates his/her knowledge using a Bayesian procedure. We first notice the subject's knowledge about θ^* at the beginning of the t-th step, $p_{t-1}(\theta)$, is the posterior probability after observing the ideal movement direction and the device's actual movement direction sequence in the first $(t-1)$ steps, $p_{t-1}(\theta) = p(\theta | \{\psi_\tau^*, \tilde{\psi}_\tau\}_{\tau=1}^{t-1})$. At the beginning of the

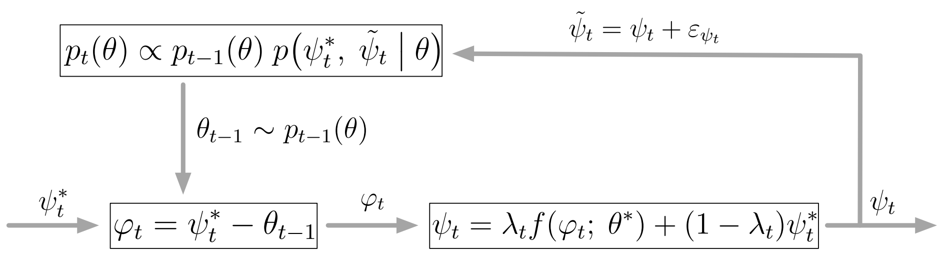

$\tilde{\psi}_t = \psi_t + \varepsilon_{\psi_t}$
$p_t(\theta) \propto p_{t-1}(\theta)\, p\big(\psi_t^*,\, \tilde{\psi}_t \mid \theta\big)$
$\theta_{t-1} \sim p_{t-1}(\theta)$
ψ_t^*
$\varphi_t = \psi_t^* - \theta_{t-1}$
φ_t
$\psi_t = \lambda_t f(\varphi_t;\, \theta^*) + (1-\lambda_t)\psi_t^*$
ψ_t

Figure 5.1: **One step updating.**

t-th step, $p_{t-1}(\theta)$ can be treated as the prior probability before the new observation at t. The likelihood of the observation can be decomposed into two parts

$$p\big(\psi_t^*,\, \tilde{\psi}_t \mid \theta\big) = p\big(\tilde{\psi}_t \mid \psi_t^*,\, \theta\big)p\big(\psi_t^* \mid \theta\big), \tag{5.9}$$

where $\tilde{\psi}_t$ is the subject's perceived output at the t-th step. Notice, since the subject is unaware of the existence of SMC, from the subject's view the perceived device's movement direction $\tilde{\psi}_t$ is solely determined by the decoding process, as

$$\tilde{\psi}_t = f(\varphi;\, \theta^*) = \varphi + \theta^*. \tag{5.10}$$

Further, since ψ_t^* is independent of θ, we can treat $p\big(\psi_t^* \mid \theta\big)$ as constant. Therefore, we have the likelihood as

$$p\big(\psi_t^*,\, \tilde{\psi}_t \mid \theta\big) \propto p\big(\tilde{\psi}_t \mid \psi_t^*,\, \theta\big) \propto \mathcal{N}\big(\varphi + \theta^*,\, \sigma_{\psi_t}^2\big). \tag{5.11}$$

With this expression of likelihood, we can update the posterior probability, which is the subject's knowledge about θ^* after perceiving data $\{\varphi_t, \tilde{\psi}_t\}$ by Bayes rule,

$$p_t(\theta) \propto p_{t-1}(\theta)\, p\big(\psi_t^*,\, \tilde{\psi}_t \mid \theta\big). \tag{5.12}$$

The subject's initial knowledge $p_0(\theta)$, which is the prior probability before any observations, is assumed to be a Gaussian distribution as $p_0(\theta) = \mathcal{N}(\mu_0, \sigma_0^2)$, where μ_0 is an arbitrary guess and σ_0 is fairly large. Notice the likelihood at each step is also Gaussian, if the decoding function f is linear (as is true in our case), then $p_t(\theta)$ is also Gaussian. Assuming $p_{t-1}(\theta) = \mathcal{N}\left(\mu_{t-1}, \sigma_{t-1}^2\right)$, we can update the posterior as $p_t(\theta) = \mathcal{N}\left(\mu_t, \sigma_t^2\right)$, where

$$\mu_t = \left(\mu_{t-1}\sigma_{t-1}^{-2} + \big(\tilde{\psi}_t - \varphi_t\big)\sigma_{\psi_t}^{-2}\right)\sigma_t^2, \tag{5.13}$$

$$\sigma_t^2 = \left(\sigma_{t-1}^{-2} + \sigma_{\psi_t}^{-2}\right)^{-1}. \tag{5.14}$$

A schematic of the subject's knowledge updating at the t-th step is shown in Fig. 5.1.

5.3.2 Why might shared mode control assist learning?

Empirically it has been found that SMC hastens learning, however, it is not immediately clear why this should be so. In this section, we derive a rationale for this empirical finding under the assumption that the subject's sensory feedback error travels along a bandwidth limited channel.

First, we notice that the subject's *observed error* is the difference between the perceived movement direction $\tilde{\psi}_t$ and the ideal direction ψ_t^*, i.e.,

$$\begin{aligned} error_{observed} &= (\tilde{\psi}_t - \psi_t^*)^2 \\ &= (\psi_t + \varepsilon_{\psi_t} - \psi_t^*)^2 \end{aligned} \tag{5.15}$$

From Eqn. 5.7 we can see with SMC, the observed error can be written as

$$\begin{aligned} error_{observed} &= (\psi_t - \psi_t^* + \varepsilon_{\psi_t})^2 \\ &= (\lambda_t(\theta^* - \theta) + \varepsilon_{\psi_t})^2 \end{aligned} \tag{5.16}$$

On the other side, the *decoding error*, which is the error the subject actually makes (before SMC corrects it), is the difference between the actual decoded direction and the ideal direction, i.e.,

$$\begin{aligned} error_{decoding} &= (\hat{\psi}_t - \psi_t^*)^2 \\ &= (\psi_t^* - \theta + \theta^* - \psi_t^*)^2 \\ &= (\theta - \theta^*)^2 \end{aligned} \tag{5.17}$$

Although decreasing λ_t can improve performance, it also makes the *observed error* deviate from the *decoding error* the subject actually made. Under SMC, some information about the system is hidden and the subject cannot get veridical feedback. For the subject to get the most veridical feedback of aiming error SMC must be off. This seems inconsistent with the observation that SMC hastens learning.

The key to this problem is the perceptual noise. If the variance of the perceptual noise is constant all the time, the above conclusion that SMC is of no help is correct. However, if the variance of the perceptual noise is not constant but depends on the subject's performance, then SMC can actually aid learning. Since the movement direction the subject expects to see is the ideal direction, ψ_t^*, intuitively, when the system's output is close to what the subject expects, it may be easier for the subject to accept the feedback so the perceptual noise is relatively small. On the other hand, when the system's output is far away from what the subject expects, it may be harder for the subject to accept the feedback, so the perceptual noise is relatively large. In the next paragraph, we will formulate the relationship between the subject's performance and the perceptual noise in a mathematical way.

Suppose we treat the subject as a communication channel with a limited bandwidth. From the Shannon-Hartley theorem we know that the channel capacity C_{max}, or, in our case, the upper bound on the information rate the subject can acquire each step, is

$$C_{max} \geq C_t = B \log\left(1 + S_t/N_t\right), \tag{5.18}$$

where C_t is the information rate at the t-th step, B is a constant related to the bandwidth of the channel, and S_t and N_t are the powers of the signal and the noise at the t-th step, respectively.

As we have discussed before, when the system's input is φ_t, the subject's expected system output is ψ_t^*. Thus, the difference between the subject's expected output and the actual output can be considered as the new information from this step. Here we use mean squared error (MSE) to measure such difference, therefore, the signal power can be obtained as

$$S_t = (\psi_t - \psi_t^*)^2. \tag{5.19}$$

The noise power comes from the noise term ε_{ψ_t}. Since the noise is Gaussian, the noise power equals its variance, i.e.,

$$N_t = \sigma_{\psi_t}^2. \tag{5.20}$$

Thus, we get the information rate at the t-th step as

$$C_t = B \log\left(1 + (\psi_t - \psi_t^*)^2 / \sigma_{\psi_t}^2\right). \tag{5.21}$$

Assuming the information rate reaches its upper bound, i.e., $C_t = C_{max}$, we have the expression of the noise's variance as

$$\sigma_{\psi_t}^2 = \kappa(\psi_t - \psi_t^*)^2, \tag{5.22}$$

where $\kappa = \left(e^{C_{max}/B} - 1\right)^{-1}$ is a constant determined by the subject's channel capacity. From Eqn. 5.22, we can see the variance of the subject's perception is proportional to the error. Large κ indicates that the subject is less capable of acquiring new information since the same error will lead to larger perceptual noise.

This result reveals that when SMC is minimal, i.e., $\lambda_t = 1$, the subject receives veridical feedback about its movement, i.e., $\psi_t = f(\varphi_t;\ \theta^*)$, with the error that provides the most useful information for updating its internal conception of the motor transform. However, large error also leads to increased perceptual noise, ε_{ψ_t}, limiting the subject's ability to utilize the feedback. This conclusion is similar to the statement in [Guadagnoli and Lee, 2004], where authors argued that there is a maximum volume of new information that subjects can acquire each step. If the information provided by the task exceeds this threshold, more information will harm the subject's perception. [Ahissar and Hochstein, 1997] also demonstrated the dependence of the degree of specificity on the task difficulty. This tension suggests that optimal learning may be driven by intermediate levels of assistance that decrease as the subject gains proficiency at the task. Finding the proper schedule of λ_t to optimize the rate of learning is the goal of this chapter.

Some Observations: From Eqn. 5.13, we can see the updated posterior mean is the linear combination of μ_{t-1}, which comes from prior knowledge, and $(\tilde{\psi}_t - \varphi_t)$, which comes from the new perceived feedback. $(\tilde{\psi}_t - \varphi_t)$ can be expanded into two parts as

$$\tilde{\psi}_t - \varphi_t = \big(\lambda_t \theta^* + (1 - \lambda_t)\theta_{t-1}\big) + \varepsilon_{\psi_t}. \tag{5.23}$$

The first part $\big(\lambda_t \theta^* + (1 - \lambda_t)\theta_{t-1}\big)$ reflects how much information about θ^* the current step provides. When λ_t is close to 1, this part is close to θ^* which means that much information about the ground-truth parameter θ^* is provided. On the other side, when λ_t is close to 0, this part is close to θ_{t-1}, which means the subject's observation is quite similar to what it has already learnt and little information about θ^* is provided. So, from this point of view, to make the subject learn as much as possible, λ_t should be set as large as possible.

The second part ε_{ψ_t} is a random variable with variance $\sigma_{\psi_t}^2$. From the above discussion, this variance may be proportional to λ_t. Thus, from this point of view we want to keep λ_t small. Our goal is to find an optimal λ_t that provides the most veridical feedback while limiting the perceptual noise.

5.3.3 Adaptive difficulty control

In the above section, we argued that optimal learning may be driven by intermediate levels of assistance that decrease as the subject gains proficiency at the task. In this section, we design a strategy which can automatically adjust the task's difficulty at each step so that the subject can learn as fast as possible. Specifically, when the system obtains the decoded movement direction $\hat{\psi}_t$, we wish to implement SMC such that the corrected real movement direction ψ_t can help the subject improve its knowledge as much as possible.

To do this, we first need to define the risk of the subject's current knowledge, which should measure the distance of the current knowledge from the ground-truth. One intuitive way to do that is using the expected mean squared error between $p_t(\theta)$ and θ^*, i.e.,

$$R(p_t) = E_{p_t(\theta)}(\theta - \theta^*)^2 = (\mu_t - \theta^*)^2 + \sigma_t^2, \tag{5.24}$$

where $(\theta - \theta^*)^2$ is the bias, which measures the distance between the mean of the subject's current knowledge and the ground-truth, and σ_t^2 is the variance, which measures the subject's confidence about his knowledge. At the beginning, the subject has no knowledge about the BCI system's bias, so the mean is random guess and σ_t^2 is very large. As learning takes place, the mean of the subject's knowledge will be closer to the ground-truth and the subject will become more and more confident about his knowledge.

To make the risk converge to 0 as fast as possible, the most intuitive heuristic is to minimize the expected risk of the next step, i.e.,

$$\lambda_t = \arg\min_{\lambda \in [0,1]} E_{\varepsilon_{\psi_t}} \left[R(p_t) \right]. \tag{5.25}$$

However, in the simulation results that we will present in section 5.4, we find that this intuitive method doesn't work very well. This is because in the first few steps, the λ_t that minimizes Eqn. 5.25 tends to focus on decreasing the variance of the risk at the expense of keeping the bias high. In this case, the subject will be quite confident about some wrong knowledge after a few initial steps. Thus, more steps are needed to correct it. To prevent this case, instead of minimizing the expected risk, we try to minimize the expected bias while keeping the variance untouched. Replacing $R(p_t)$ in Eqn. 5.25 by the bias term $(\mu_t - \theta^*)^2$, we have the new optimization problem,

$$\lambda_t = \arg\min_{\lambda \in [0,1]} E_{\varepsilon_{\psi_t}} \left(\mu_t(\lambda) - \theta^* \right)^2. \tag{5.26}$$

This process successfully leads to improved learning rates.

5.4 Simulation results: optimizing assistance in shared mode control

In our simulation, all angles are confined to $(-\pi, \pi]$. Thus, the distribution of the subject's knowledge $p_t(\theta)$ is the wrapped Gaussian distribution, i.e., $p_t(\theta) = \sum_{j \in \mathbb{Z}} q_t(\theta + 2\pi j)$, where $q_t(\cdot) = \mathcal{N}(\mu_t, \sigma_t^2)$ and $\mathbb{Z}$ is the set of integers. The distance between two angles θ and θ^* is defined as $\min_{j \in \mathbb{Z}} (\theta - \theta^* + 2\pi j)^2$.

At the beginning, the subject's knowledge p_0 is a uniform distribution on $(-\pi, \pi]$. It is equivalent to the wrapped Gaussian distribution with infinite variance. The home position is $(0, 0)$ on the 2D surface and the target is randomly placed on a circle of radius 50. The cursor's step length is 1. We repeat our simulation of the learning process 100 times, where each time the system parameter θ^* is uniformly sampled from $(-\pi, \pi]$. The results shown are averaged over those 100 trials.

The learning curves of risk defined in Eqn. 5.24 corresponding to different difficulty control strategies are compared.

5.4.1 Results

In the first simulation, we fix the task difficulty λ throughout the whole learning process to study the properties of the proposed learning framework when SMC is constant. The results are shown in Fig. 5.2. Here we consider three values of signal dependent noise κ (from Eqn. 5.22), a small one ($\kappa = e^0$), a median one ($\kappa = e^2$), and a large one ($\kappa = e^4$) and three values of task difficulty λ, also ranging from easy ($\lambda = 0.2$) through medium ($\lambda = 0.6$) to hard ($\lambda = 1$). As we have seen in Sec. 5.3.2, when κ increases, the ability of the subject to acquire new information decreases. Thus $\kappa = e^0$ corresponds to the case where the subject can acquire the most information. In this case, simply setting $\lambda = 1$ allows the subject to learn fastest, as shown in Fig. 5.2. As κ increases, the subject's ability to acquire new information decreases and $\lambda = 1$ becomes too difficult for the subject. Thus, at the beginning of the learning process, the convergence rate is quite slow. In this case, making the task easy when the subject knows little about the system is more helpful. Another interesting observation is that as κ increases, the convergence rate corresponding to the easy task also increases. This is because small λ_t will make the subject's apparent performance very good, while small κ corresponds to small perceptual noise variance. That means the subject can become quite confident about its knowledge, even though that knowledge is likely wrong. In this case, it will take more steps to correct the subject's wrong knowledge. These observations argue in favor of an adaptive difficulty control schedule.

To demonstrate the effectiveness of our proposed difficulty control strategy (denoted as ADP-B), we first compare it with the strategy of fixed difficulty (denoted as FIX). The results are shown in Fig. 5.3. The blue curves correspond to the learning curves under fixed difficulty and the red one corresponds to our adaptive strategy. The averaged λ trajectory under our strategy is also shown in figures as the red dash curves. From the results we can see that our strategy is almost always better than any fixed difficulty scheduling. That demonstrates the adaptive difficulty control is necessary for fast learning and our strategy provides a good choice.

Finally, we compare ADP-B with two other control strategies. The first alternate schedule (denoted as AVGTRJ) is to use the averaged λ trajectory obtained from ADP-B universally. In other words, this schedule blindly implements the averaged λ obtained from multiple runs of the ADP-B simulation without regard to the details of the errors the subject actually makes. The second alternate schedule (denoted as ADP-R) is to choose λ to minimize the subject's expected risk at each time point, as opposed to just the bias (as discussed section 5.3.3). The results are shown in Fig. 5.4. The blue dashed curve corresponds to the averaged λ trajectory under ADP-R and the red dash curve corresponds to the averaged λ trajectory under ADP-B and AVGTRJ (which are equivalent by construction). The deficiency of AVGTRJ compared to ADP-B

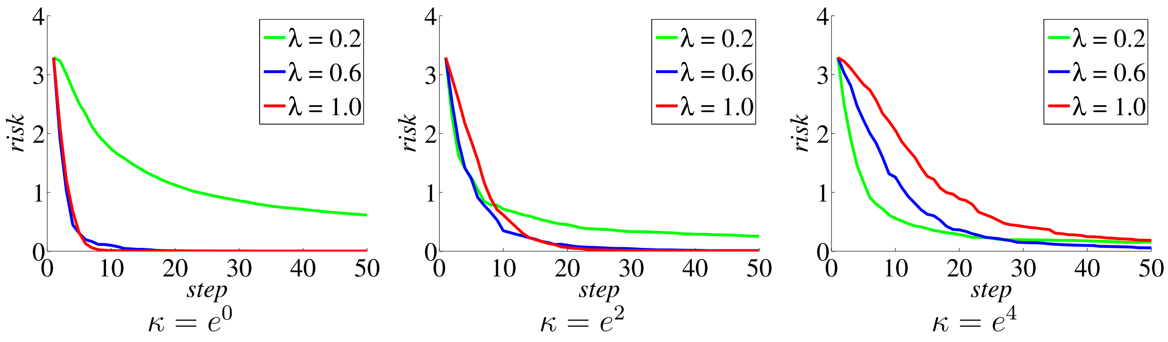


Figure 5.2: **Comparison between learning curves under fixed difficulty scheduling.**

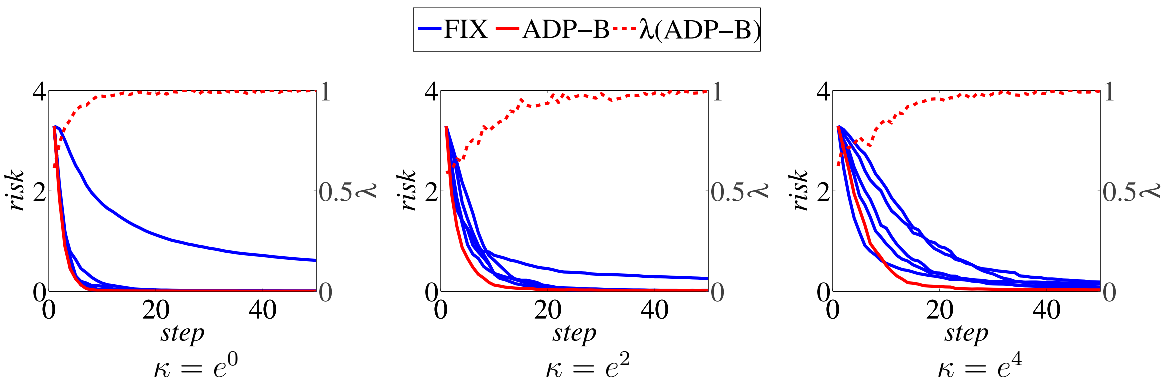


Figure 5.3: **Comparison between learning curves under adaptive difficulty scheduling and fixed difficulty scheduling.** λ trajectories are shown as dash curves.

demonstrates there exists no universal difficulty scheduling that can work well on all trials and the optimal control strategy should be adaptive on different trials.

Just as the discussion in Section 5.3.3, ADP-R performs much worse than ADP-B from the results. We find that ADP-R initially decreases quickly, but converges relatively slowly. From Section 5.3.3 we know this is because ADP-R focuses on reducing the variance of the first several steps. ADP-B, as a greedy heuristic to avoid this problem, results in fairly fast reductions in overall risk, while still allowing rapid convergence of risk to optimal levels.

5.5 Conclusions

We argued in chapter 4 that the optimal BCI should be achieved as the end result of a learning process. Here we provided a first attempt at modeling this BCI learning process. The Bayesian learning framework developed here has two novel features that capture the specifics of learning in a BCI context. First, we explicitly incorporate the shared-mode control process used by many BCI labs to assist in subject training [Sadtler et al., 2014, Taylor et al., 2002, Velliste et al., 2008]. Second, we treat the subject as a band-limited communication process. The resulting training schedule adjusts task difficulty to improve the subject's learning rate. Our simulation results demonstrate the effectiveness of the adaptive training strategy.

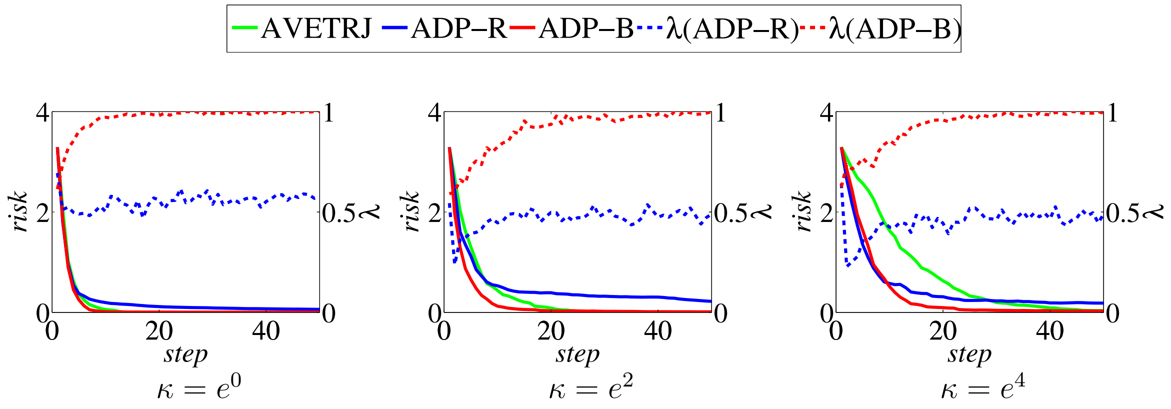


Figure 5.4: **Comparison between learning curves under adaptive difficulty scheduling and universal difficulty scheduling.** λ trajectories are shown as dash curves.

We developed our framework by analogy of the subject as a limited-bandwidth communication channel, which links the perceptual noise in the system to the overall success rate: the greater the error in the subject's output, the larger the noise in the observed movement. An identical framework results if one instead assumes that the subject's motivation depends on overall success rate. Because motivation affects attention, and attention impacts perceptual noise Lu and Dosher [1998], increased computer assistance leads to decreased perceptual noise. From this perspective our simulations show that learning rate improves when task difficulty is manipulated adaptively while explicitly accounting for subject motivation. These promising results suggest that Bayesian learning can offer useful insights and methods for teaching subjects to use a brain-computer interface device.

Chapter 6

Conclusions and future work

In this chapter I summarize the contributions of the thesis, discuss some possible avenues for future work, and conclude with a brief summary of the thesis.

6.1 Contributions of this thesis

The key contributions of this thesis are as follows.

Design adaptive difficulty-scheduling to hasten subject's learning BCI control. We introduce a Bayesian framework for modeling the closed-loop BCI learning process and develop an adaptive algorithm to find the optimal difficulty-schedule for performance improvement. Simulation results demonstrate that our algorithm yields faster learning rates than several other heuristic training schedules.

- Y. Zhang, A. B. Schwartz, S. M. Chase, and R. E. Kass. Bayesian learning in assisted brain- computer interface tasks. In *Proceedings of the 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 2740-2743, 2012.

Recast the BCI decoder design problem from a physical control system perspective and develop a post-learning optimal BCI decoder from optimal control theory. Different from most approaches that design BCI decoders from an estimation standpoint, we recast the decoder design problem from a physical control system perspective, present a formal definition of the usability of a device, and show that the optimal, post-learning mapping can be written as the solution of a constrained optimization problem. We then derive the optimal mappings for particular cases common to most brain-computer interfaces.

- Y. Zhang and S. M. Chase. Recasting brain-machine interface design from a physical control system perspective. *Journal of Computational Neuroscience*, 39(2):107-118, 2015.
- Y. Zhang and S. M. Chase. A control-theoretic approach to brain-computer interface design. In *American Control Conference*, pages 5765-5771, 2016.
- Y. Zhang and S. M. Chase. Optimizing the usability of brain computer interfaces. *Neural Computation*, 30(5):1323-1358, 2018.

Apply a dual estimation procedure to adaptively capture changes in BCI decoding parameters. We develop a parameter tracking algorithm based on the dual Kalman filter to adaptively capture changes in decoding parameters in order to solve the instability problem prevalent in BCI systems. In simulation, we find that our stabilized dual Kalman filter can run autonomously for hundreds of thousands of trials with little change in performance and in an off-line task to estimate arm trajectories from neural data recorded over five consecutive days, it outperforms the daily calibrated Kalman filter. The details of this algorithm are presented in Appendix A.

- Y. Zhang and S. M. Chase. A stabilized dual Kalman filter for adaptive tracking of brain-computer interface decoding parameters. In *Proceedings of the 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 7100-7103, 2013.

6.2 Conclusions

The optimal way to implement a BCI decoding algorithm is an important question relevant to clinical deployment of neural prostheses. Different from typical BCI decoding algorithms to maximize the biomimetic properties of the system, in chapter 3, we recast this problem from the perspective of control system design and try to interpret differences in the performance of two decoding algorithms from differences in the control systems themselves: some control systems may be more usable than others, due to their physical characteristics or ease of conceptualization. We found that 1) 2nd order physical systems tend to be more usable than 1st order physical systems, 2) decoders that cannot be expressed as simple physical control systems do not appear to work as well as those that can be expressed this way, and 3) a 2nd order control system with elastic terms seems to work better than one without.

We next introduced a formal definition of the usability of a BCI, and formulated the optimal, post-learning decoder design problem as a constrained optimization problem. In the forward optimal control problem, one knows the plant and the cost function, and the problem is to compute the control signals that drive the plant to the desired states with minimal cost. The linear quadratic regulator (LQR) is a special case of the general forward control problem in which the plant is linear and the cost function is quadratic. In the inverse control problem, one is given instances of plant states and control signals that were generated in an optimal fashion (i.e., by minimizing the cost function), and the goal is to infer the cost function. An example of this is path planning [Abbeel and Ng, 2004, Ng and Russell, 2000, Ratliff et al., 2006, Ziebart et al., 2008]. In chapter 4, we introduced a third class of problem, which we term optimal plant design. In the optimal plant design problem, one is given the cost function, and the goal is to design a plant that can perform that task with minimal overall cost. We used this approach to design an optimal linear, second order BCI system that might be used clinically to aid patients with motor control disorders. We found that the common approach of creating biomimetic interfaces may not be optimal when learning is taken into account. Going beyond BCI design, we believe the ideas could be applied to any system that can be defined in optimal control theoretic terms. If so, the applications of optimal plant design could potentially be wide-ranging, from the design of robotic systems, to chemical processing plants, to medical devices or beyond.

Once we obtain the optimal BCI decoding algorithm, we want the subject to learn how to control it as fast as possible. In chapter 5, we proposed a Bayesian learning framework which treats the subject as a band-limited communication channel. Based on this learning framework, we first presented a formal mathematical explanation for the observation that shared mode control hastens learning, even though it actually masks subject error. By treating perception as an information-limited band-limited communication process, we demonstrated an explicit link between the difficulty-schedule and the learning rate. We next developed an adaptive algorithm to find the optimal difficulty schedule for performance improvement. In simulation, our adaptive difficulty-control strategy promotes a marked improvement in learning rate.

6.3 Future work

Our analyses suggest a number of new approaches to BCI decoding algorithm design that might prove fruitful. Of all the decoding algorithms we reviewed in chapter 3, none went beyond 2nd order. Given that 2nd order systems appear more usable than 1st order systems, it is interesting to speculate as to whether a 3rd or 4th order system would be even easier to control. These higher-order systems may actually be a closer match to the human arm: in [Liu and Todorov, 2007], Liu and colleagues model the arm as a 3rd order linear physical system and are able to capture many of the emergent features of natural reaching movements. There have been instances in the literature that have included acceleration and higher order terms in their decoding algorithms. For example, Wu and colleagues compared the performance of a Kalman filter decoder with up to 6th order terms, and found that the 3rd order model consisting of position, velocity, and acceleration terms provided the best performance in their off-line trajectory reconstruction [Wu et al., 2006]. However, they used the position-implementation of their decoder, which we have already demonstrated does not correspond to a simple physical system beyond 1st order. It would be interesting to test how physical implementations of higher order control systems might perform on-line.

Another fruitful approach might be the design of state-dependent control systems. The PVA and the OLE are both static systems, i.e., the physical control parameters do not vary with time. The Kalman filter technically has time varying parameters, but in practice the Kalman gain converges to a constant within a few timesteps, and some researchers even initialize it at the converged-values to keep it time-invariant [Malik et al., 2011, Sadtler et al., 2014]. State-dependent control has been used in situations where static systems are no longer appropriate. One such situation is the instability of tuning curves between sessions [Chestek et al., 2009, Rokni et al., 2007]. To compensate for such instability, different adaptive decoding algorithms are proposed where decoding parameters are updated iteratively over time [Kowalski et al., 2013, Li et al., 2011, Orsborn et al., 2014, Shpigelman et al., 2009, Suminski et al., 2013, Zhang and Chase, 2013]. Another kind of state-dependent control is used to assist the subject during training with the BCI system, where the assistance is adjusted according to the subject's performance [Velliste et al., 2008]. Other state-dependent control algorithms include [Shanechi et al., 2013], where in order to simultaneously estimate movement trajectory or target intent, the decoding parameters are adjusted as targets are approached. Such state-dependent control was the goal behind the recently-developed speed dampening Kalman filter [Golub et al., 2014], which effectively manipulated the viscous damping forces as a function of trajectory curvature to allow for more stable transitions from moving to stopping. The physical control system viewpoint we espouse here may be a way of integrating these approaches into a simple, unified framework for robust prosthetic applications.

Finally, we should note that we have focused exclusively in this thesis on kinematic BCI decoders of the type that are commonly used to drive cursors on computer screens. As discussed in section 1.3, another class of decoders attempts to extract kinetic information (forces and torques) from recorded neural activity for direct control of force output [Chhatbar and Francis, 2013, Fagg et al., 2009, Nazarpour et al., 2012, Nishimura et al., 2013, Oby et al., 2013]. It is unclear at present how to best integrate these two approaches to BCI design. One intriguing idea is that the brain represents impedances (relationships between kinetics and kinematics) rather than desired forces or kinematics per se [Hogan, 1985, Hogan and Sternad, 2012, Tin and Poon, 2005]. An interesting, active direction of research is to design decoders that seemlessly transition between free movement and object interaction. It is possible that decoding impedance control signals directly from the brain would enable this transition.

Appendix A

A stabilized dual Kalman filter for adaptive decoding

While my thesis work focused on the notion of optimal BCI design, during my PhD studies I also worked on a method for developing a BCI that would be stable to electrode recording instability and thus require fewer re-calibration sessions. Those results are detailed in this appendix and have been published in [Zhang and Chase, 2013].

A.1 Introduction

Most BCIs require a calibration session to compute decoding parameters that define how recorded neural activity will translate into movement of the device. Calibration is necessary to build models of how neural firing rates are modulated by desired movement. However, these calibrations are not particularly stable: decoding parameters estimated in one session will often not apply even on the next day [Chestek et al., 2009]. This instability could be due to electrode drift, changes in background noise, or changes in the neural tuning curves themselves [Rokni et al., 2007]. Regardless of its source, the daily re-training of the decoding algorithm necessary to achieve optimal performance is a major factor limiting the clinical utility of this technology.

Some approaches have previously been proposed to combat BCI decoder instability. In [Li et al., 2011], Bayesian regression self-training methods were used to update parameters based on estimates from an unscented Kalman filter. In [Shpigelman et al., 2009], a kernelized auto-regressive moving average was employed to change the decoder over time. Here we introduce an alternative method, based on a dual-Kalman filter, as a simple extension to commonly used BCI decoders. Dual-estimation can be sensitive to self-training, which appears to also be a problem in our simulations. However, we find that two simple heuristics can be implemented to substantially improve the estimation stability. We find in both simulation and off-line analysis that our stabilized dual Kalman filter remains robust to parameter drift over long time scales.

A.2 Stabilized dual Kalman filter

Typical BCI decoders are based on the Kalman filter, due to its rigorous theoretical framework and easy implementation [Wu et al., 2003]. In section 2.4, I have reviewed the Kalman filter in detail. While the Kalman filter has successfully been used to achieve proficient closed-loop BCI control [Gilja et al., 2012, Wu et al., 2003], the implicit assumption that the linear model parameters are fixed over time may limit its performance. A common problem in BCI decoding is that neurons are unstable. Micro-movements of the electrode array relative to the brain can cause the amplitude of recorded neurons to change dramatically [Downey et al., 2018, Gilja et al., 2011]. Furthermore, there is evidence that neural tuning curves themselves may change over time, slowly altering the relationship between firing rate and intended movement [Chase et al., 2012, Ganguly and Carmena, 2009, Rokni et al., 2007]. Traditional Kalman filters will not compensate for this variation.

Parameter Tracking Algorithm

To compensate for neural tuning changes, we propose an adaptive decoding model with parameter tracking. In this model, C from Eq. 2.9 are no longer constant over time. To emphasize the time-varying property, here we use C_τ to denote the linear encoding model parameters, where the subscript τ indicates the time scales for the tuning parameter change. Under the new framework, the state evolution model remains the same as Eq. 2.8:

$$\boldsymbol{x}_t^* = A\boldsymbol{x}_{t-1}^* + \boldsymbol{\omega}_t, \tag{A.1}$$

while the linear observation model Eq. 2.9 is replaced by

$$\boldsymbol{y}_t = C_\tau \boldsymbol{x}_t^* + \boldsymbol{\varepsilon}_t. \tag{A.2}$$

Notice time scales for the limb state change, indicated by t and time scales for the tuning parameter change, indicated by τ, are usually different and the later is much slower. If we also apply a linear smooth prior on C_τ, it becomes another linear dynamical system. For this system, the history of neurons' firing rates and estimated limb states can be used to track the evolution of C_τ. Denoting the vector $\boldsymbol{c}_{i,\tau}$ as the transpose of the i-th row of C_τ, then, for the encoding parameters evolution we have

$$\boldsymbol{c}_{i,\tau} = \boldsymbol{c}_{i,\tau-1} + \boldsymbol{\zeta}_{i,\tau}, \tag{A.3}$$

$$\boldsymbol{y}_{i,t|\Pi} = \hat{X}_{t|\Pi}^T \boldsymbol{c}_{i,\tau} + \boldsymbol{\varepsilon}_{i,t|\Pi}, \tag{A.4}$$

where $\boldsymbol{\zeta}_{i,\tau} \sim \mathcal{N}(0, Z_i)$ is the noise on the encoding parameter's dynamics, $\boldsymbol{y}_{i,t|\Pi}$, $\hat{X}_{t|\Pi}$ and $\boldsymbol{\varepsilon}_{i,\tau|\Pi}$ represent the recorded firing rates, the decoded velocity and the noise during the previous Π steps respectively, i.e.,

$$\boldsymbol{y}_{i,t|\Pi} = \left(y_{i,t-\Pi+1}, \ldots, y_{i,t}\right)^T \tag{A.5}$$

$$\hat{X}_{t|\Pi} = \left(\hat{\boldsymbol{x}}_{t-\Pi+1}, \ldots, \hat{\boldsymbol{x}}_t\right) \tag{A.6}$$

$$\boldsymbol{\varepsilon}_{i,\tau|\Pi} = \left(\varepsilon_{i,\tau-\Pi+1}, \ldots, \varepsilon_{i,\tau}\right)^T. \tag{A.7}$$

We assume that parameter change is quite slow compared to the state change, so parameters are updated only every Π steps, and are fixed between updates. To track both the limb state evolution and the encoding parameters evolution, we can apply dual Kalman filter [Nelson, 1976], where two Kalman filters are running in parallel. The Kalman filter tracking the limb state will be updated every step while the Kalman filter tracking the encoding parameters will be updated every Π steps.

In simulation, we have observed that tracking parameters and limb state without any regularization tends to lead to long-term parameter drift. To further constrain the tuning parameters, we propose two heuristics for regularization:

Baseline firing rate estimation

For a sufficiently long window of time, the average firing rate of each neuron will converge to its baseline firing rate $\boldsymbol{r}_\tau$ [1]. This is because the long-term average velocity of the prosthetic limb must be zero. Define $\bar{\boldsymbol{y}}_{t|\Pi} = 1/\Pi\big(\sum_{\pi=0}^{\Pi-1} \boldsymbol{y}_{t-\pi}\big)$, $\bar{\boldsymbol{x}}^*_{t|\Pi} = 1/\Pi\big(\sum_{\pi=0}^{\Pi-1} \boldsymbol{x}^*_{t-\pi}\big)$ and $\bar{\boldsymbol{\varepsilon}}_{t|\Pi} = 1/\Pi\big(\sum_{\pi=0}^{\Pi-1} \boldsymbol{\varepsilon}_{t-\pi}\big)$ as the average firing rate, the average estimated velocity and the average noise between time $t - \Pi + 1$ and t respectively. From the encoding model Eq. A.2, we require that

$$\bar{\boldsymbol{y}}_{t|\Pi} = C_\tau \bar{\boldsymbol{x}}^*_{t|\Pi} + \bar{\boldsymbol{\varepsilon}}_{t|\Pi} \approx \boldsymbol{r}_\tau. \tag{A.8}$$

To implement this in our parameter tracking algorithm, when the decoding parameters are updated we take the average firing rate over the previous Π timesteps as the new baseline firing rate.

Velocity Normalization

The second heuristic we implement to stabilize and improve the parameter tracking algorithm is to normalize the estimated velocity before the parameter update. Essentially, we assume that the subject's velocity over a sufficiently long time window will follow a stable distribution. To implement the velocity normalization, the median of the absolute value of the estimated velocity on each dimension during Π steps, $median\big(\,\big|\hat{v}_{i,t|\Pi}\big|\,\big)$, is required to be the median of the absolute value of the velocity on the corresponding dimension recorded from training data, denoted as $median\big(\big|\hat{v}_{i,train}\big|\big)$. Therefore, instead of using $\hat{V}_{t|\Pi}$ in Eq. A.4, we use $\Lambda\hat{V}_{t|\Pi}$ where Λ is a diagonal matrix with $median\big(\big|\hat{v}_{i,train}\big|\big)/median\big(\big|\hat{v}_{i,t|\Pi}\big|\big)$ on the i-th position of the diagonal.

A.3 Results

A.3.1 Simulation

To test the efficacy of our parameter tracking algorithm, we designed the following 100,000 trial simulation.

[1]Notice here in order to take the drift of baseline firing rate into consideration, the firing rate $\boldsymbol{y}_t$ is *non-centralized*.

Kinematics

Our simulated subject repeatedly draws a Lissajous curve in 2D. The velocity of the trajectory is given by $\boldsymbol{v}_t = \big(s_1 f_1 \sin(f_1 t),\ s_2 f_2 \cos(f_2 t)\big)^T$, where t is the time (in seconds), and $s_1 = 0.2$m, $s_2 = 0.1$m, $f_1 = 2/3\pi$ Hz, $f_2 = 2\pi$ Hz. Here to take the baseline firing rates into consideration, we concatenate the velocity with 1 to get the state vector, i.e., $\boldsymbol{x}_t = (\boldsymbol{v}_t; 1)$.

Each trial is one complete cycle of the curve. We set the time step $\Delta = 3$ms, so each trial consists of 100 steps. Parameters are updated every 20 trials (described below), comprising one session. We run 5,000 sessions in total.

Encoding model

100 neurons are simulated. The initial parameters for neuron i include:

- the preferred direction, $\theta_{i,0}$, randomly sampled from $[0, 2\pi)$;
- the baseline firing rate, $r_{i,0}$, randomly sampled from $[5, 10]$Hz;
- the modular depth, $m_{i,0}$, randomly sampled from $[4, 8]$m/s^{-1}.

We use a vector $\boldsymbol{\beta}_{i,0}$ to represent the initial parameters associated with neuron i:

$$\boldsymbol{\beta}_{i,0} = \begin{pmatrix} m_{i,0} \cos\left(\theta_{i,0}\right) \\ m_{i,0} \sin\left(\theta_{i,0}\right) \\ r_{i,0} \end{pmatrix}. \tag{A.9}$$

The encoding model is assumed to be linear-Gaussian in our simulation. That is, the firing rate of neuron i at time t is a linear function of the state $\boldsymbol{x}_t$,

$$y_{i,t} = \boldsymbol{\beta}_{i,\tau}^T \boldsymbol{x}_t + \epsilon_{i,t}, \tag{A.10}$$

where $\epsilon_{i,t} \sim \mathcal{N}\left(0, \sigma_\epsilon^2\right)$ represents the spiking noise.

Unstable neurons

To model instability in the neural tuning curves, the parameter vector's dynamics are simulated as

$$\boldsymbol{\beta}_{i,\tau+1} = \boldsymbol{\beta}_{i,\tau} + \boldsymbol{\gamma}_{i,\tau}, \tag{A.11}$$

where $\boldsymbol{\gamma}_{i,\tau} \sim \mathcal{N}\big(\mathbf{0}, \sigma_\gamma^2 I\big)$. As stated before, the parameter change is quite slow compared to the velocity change. So in our simulation, the parameter changes every session (20 trials or 2,000 timesteps).

Decoding Algorithms

We compare the following decoding algorithms

- Static Kalman Filter (KF): The traditional Kalman filter with the assumption that the linear model parameters are fixed over time.

- Ground-truth Kalman Filter (KF$_{GT}$): In simulation we always know the exact values of the parameters. Therefore, we also run the ideal Kalman filter where at each step the tuning parameters $\boldsymbol{c}_{i,\tau}$ are set to be the ground truth $\boldsymbol{\beta}_{i,\tau}$. This method has the best decoding performance that can be achieved in the presence of unknown observation noise. We use this as a benchmark on which to test the performance of our parameter tracking algorithm.
- Parameter Tracker without stabilization (PT$_{NS}$): The parameter tracking algorithm without either stabilization heuristics.
- Parameter Tracker with baseline stabilization (PT$_{c_0}$): The parameter tracking algorithm using only the baseline firing rate estimation as the stabilization heuristic.
- Parameter Tracker with velocity normalization (PT$_{VN}$): The parameter tracking algorithm using only the velocity normalization as the stabilization heuristic.
- Stabilized Parameter Tracker (PT): The parameter tracking algorithm using both heuristics for stabilization.

Parameter setting

All algorithms are initialized at time zero with the ground truth parameters, $\boldsymbol{c}_i = \boldsymbol{\beta}_{i,0}$ for the Kalman filter and $\boldsymbol{c}_{i,0} = \boldsymbol{\beta}_{i,0}$ for the parameter tracker. The state-evolution covariance matrix R is estimated from the actual kinematics. We take T to be 2000 time steps. We quantify performance with the mean integrated squared error (MISE) between the decoded velocity and the actual velocity.

We ran several simulations with varying amounts of noise. The standard deviation of the observation noise, σ_ϵ, took on values of e^{-1} and e^{-3} Hz. The standard deviation of the evolution noise, σ_γ, took on values of e^{-1} and e^{-3} (units of Hz(m/s)$^{-1}$ for the first two dimensions and Hz for the last dimension).

Simulation Results

The MISE results are shown in Fig. A.1. KF$_{GT}$ always has the lowest MISE and the performance does not change much as the experiment runs. KF has very low MISE at the beginning, but diverges very fast since it assumes the parameters are fixed. The performance of PT is close to KF$_{GT}$ and demonstrates that PT can track the parameter evolution quite well. From the results we can see the parameter tracker without stabilization doesn't perform very well due to the over-fitting problem. However, the performance improves substantially with either heuristic. An interesting phenomenon is that the MISE of KF$_{GT}$ decays over time under the noise setting $\sigma_\epsilon = e^{-1}$, $\sigma_\gamma = e^{-1}$. This is because under the random-walk model, the magnitude of the tuning parameters can grow over time, effectively increasing the signal to noise ratio. We did not attempt to constrain this parameter in our simulations.

The reconstructed velocity of different methods of the 100,000th trial is shown in Fig. A.2. KF$_{GT}$ represents the best decoding performance we can get with this level of observation noise. The performance of PT at the 100,000th trial is similar to KF$_{GT}$, while the performance of KF is quite biased.

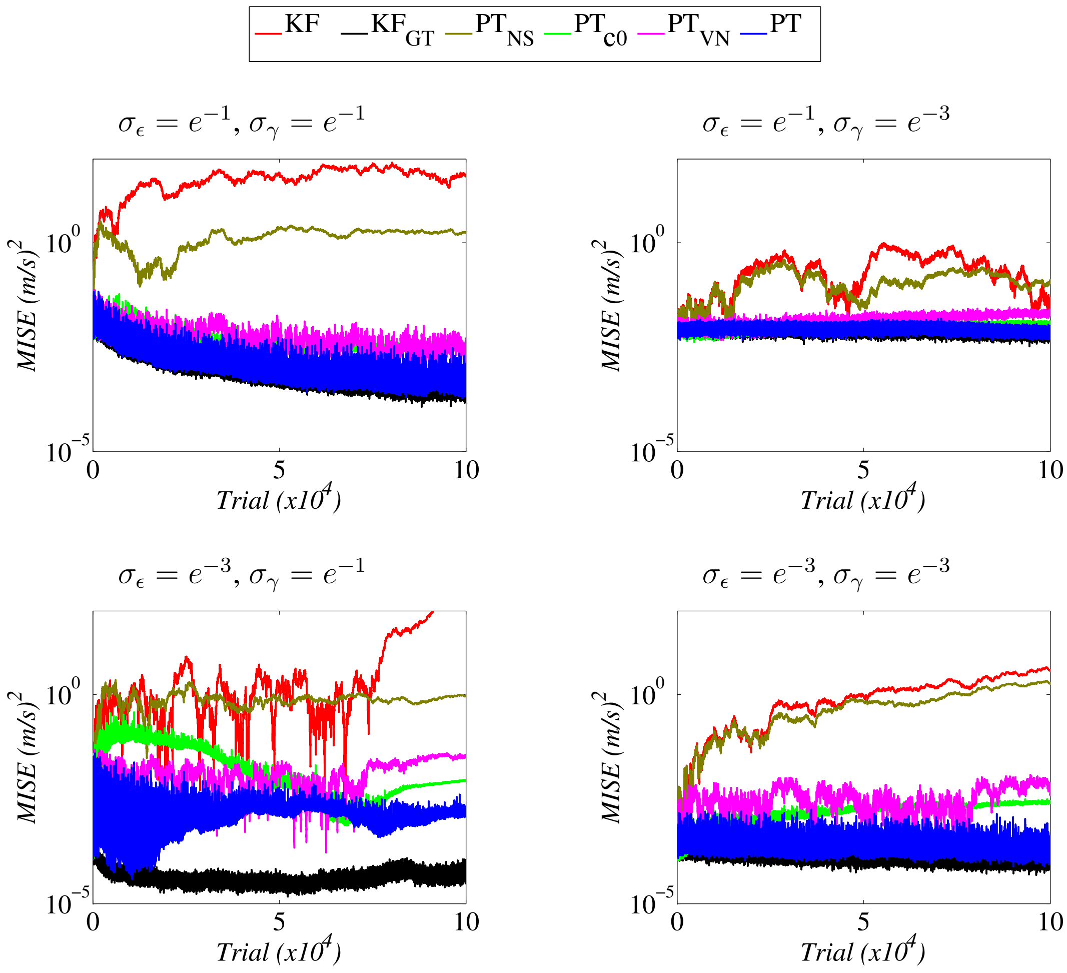


Figure A.1: **Simulation results**: MISE (in log scale) of each of the different methods under various observation noise levels, σ_ϵ, and parameter noise levels, σ_γ.

A.3.2 Offline trajectory reconstruction

We also demonstrate the efficacy of our approach by reconstructing off-line the trajectories of actual arm reaches over 5 consecutive days using simultaneously recorded neural data.

Experimental Details

Briefly, a Rhesus macaque was trained to sit in a primate chair and make center-out and out-center reaches to 26 targets presented in 3D space. Hand position samples were tracked at 30Hz in 3D space using an Optotrak recording system. Spike trains from 53 neurons were recorded with a Utah microelectrode array and tracked over the 5 days of the experiment. Firing rates were computed in 100ms sliding windows sampled at 30Hz. Full experimental details can be found in [Fraser and Schwartz, 2012]. We used a fixed lag of three timesteps between neural activity and predicted kinematics for all neurons.

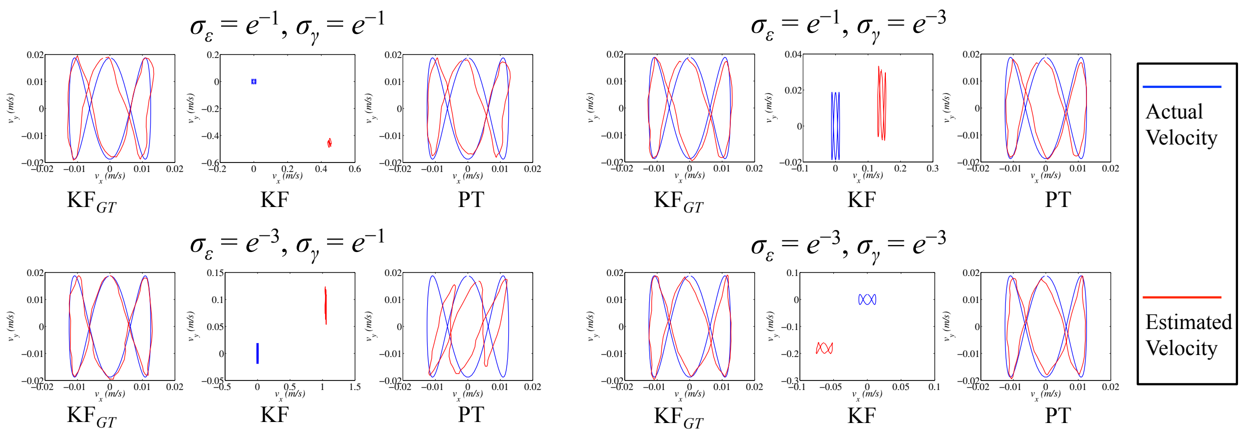


Figure A.2: **Simulation results**: Reconstructed velocity of the 100,000th trial of the simulation resulting from three of the algorithms under various amounts of observation and parameter noise. Note the static KF (middle panels) tends to show an offset, indicative of strongly biased decoding. In contrast, the stabilized dual Kalman filter (right-hand panels) performs nearly as well as the ground-truth Kalman filter (left-hand panels) under most noise contexts.

Reconstruction Methods

We compare the ability of each of the algorithms described above to reconstruct the actual trajectories made by the subject over the 5 consecutive days. However, instead of using a ground truth Kalman filter, as done in simulation, we compare the decoding results to a static Kalman filter calibrated only on day 1 (KF), as well as to a static Kalman filter recalibrated using the first session at the beginning of each day (KF$_{SD}$).

Parameter Estimation

Data recorded from day 1, session 1 are used for training to learn the parameters, including $\boldsymbol{c}_i$ for the Kalman filter, $\boldsymbol{c}_{i,0}$ for the parameter tracker, R and Q. To learn the covariance matrix Z_i for parameter evolution we first notice that there are two kinds of parameter evolutions, one between sessions, denoted as $Z_{i,session}$, and one between days, denoted as $Z_{i,day}$. We use the data from day 1 to learn $Z_{i,session}$ and the data from all 5 days to learn $Z_{i,day}$. We assume the covariance matrixes of parameter evolution are the same for each neuron.

Results

The results are shown in Fig. A.3. On day 1, each of the algorithms performs similarly, as expected. However, on subsequent days the static Kalman filter returns progressively worse trajectory reconstructions. In contrast, the parameter-tracking algorithms perform well on all days, and even outperform a static Kalman filter calibrated on the trials at the beginning of each day.

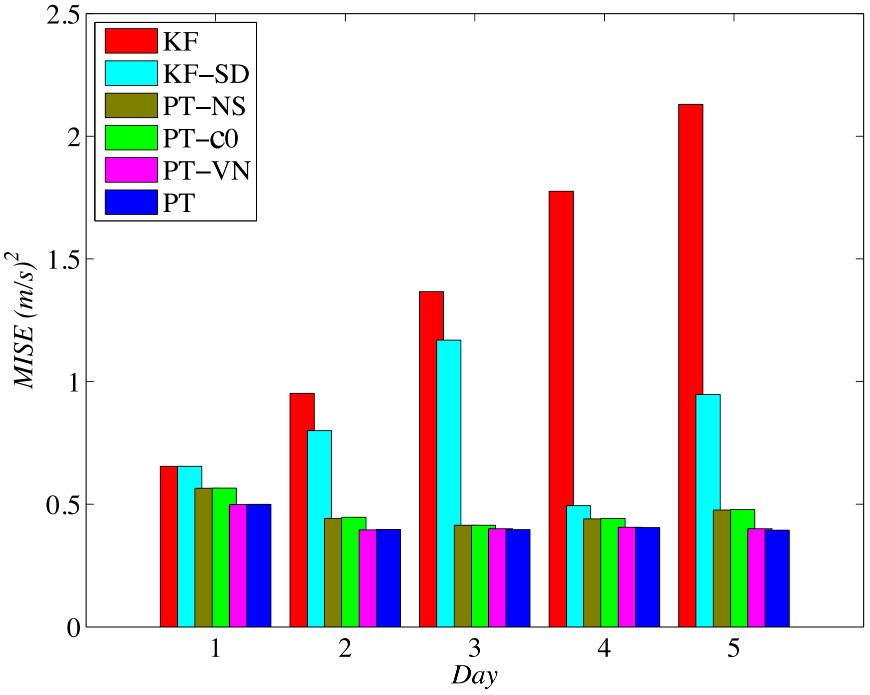


Figure A.3: **Off-line trajectory reconstruction results**: MISE of different methods on the offline data.

A.4 Conclusions

Current BCI systems require daily re-calibration of decoding parameters to optimize performance. These re-calibration sessions can be lengthy, and may limit the clinical utility of neural prosthetic systems. In order to eliminate the need of calibration, we implement a dual Kalman filter to track both the limb states and the parameters, augmented with two stabilizing heuristics: baseline firing rate estimating and velocity normalization. Our stabilized dual Kalman filter performs well in both simulation and in estimating arm movement trajectories off-line.

Here we focused on the instability of the neural tuning parameters while assuming fixed variance. Testing how much improvement may be gained by also tracking the variance remains the subject of future work. We also assumed that the number of neurons we tracked remained fixed. In practice, neurons may drop out or come in to the recording over time. This complicates the implementation of parameter tracking, because the dimensionality of the tuning matrix will change over time [Li et al., 2011]. One way to simply capture this behavior would be to always track a stable “maximum” number of neurons. Those that correspond to blank signals would return tuning coefficients of zero. If the residual variance assigned to those units is constrained to be non-zero, they should not affect the decoding solution. As neurons come in to the recording, the algorithm would start to measure their tuning. Implementation of this extension remains a subject of future work.

Appendix B

Proof for the design of a provably-optimal BCI system

B.1 Derivation of modified Riccati recursion for signal-dependent noise

In order to extend LQR for the signal-dependent noise case, we consider the cost-to-go function at time t, $\mathcal{J}(\boldsymbol{x}_t, \boldsymbol{x}^g, \boldsymbol{z}_{t,\dots,T-1})$, which is defined as the expected future cost if the system is at state $\boldsymbol{x}_t$ and following control signals are $\boldsymbol{z}_t, \dots, \boldsymbol{z}_{T-1}$, i.e.,

$$\begin{aligned}\mathcal{J}(\boldsymbol{x}_t, \boldsymbol{x}^g, \boldsymbol{z}_{t,\dots,T-1}) &= \sum\nolimits_{\tau=t}^{T} \mathbb{E}\left(\boldsymbol{x}_\tau^T Q_\tau \boldsymbol{x}_\tau\right) + \sum\nolimits_{\tau=t}^{T-1} \boldsymbol{z}_\tau^T R_\tau \boldsymbol{z}_\tau \\ &= \boldsymbol{x}_t^T Q_t \boldsymbol{x}_t + \boldsymbol{z}_t^T R_t \boldsymbol{z}_t + \sum\nolimits_{\tau=t+1}^{T} \mathbb{E}\left(\boldsymbol{x}_\tau^T Q_\tau \boldsymbol{x}_\tau\right) + \sum\nolimits_{\tau=t+1}^{T-1} \boldsymbol{z}_\tau^T R_\tau \boldsymbol{z}_\tau \\ &= \boldsymbol{x}_t^T Q_t \boldsymbol{x}_t + \boldsymbol{z}_t^T R_t \boldsymbol{z}_t + \mathbb{E}_{\boldsymbol{x}_{t+1}} \mathcal{J}(\boldsymbol{x}_{t+1}, \boldsymbol{x}^g, \boldsymbol{z}_{t+1,\dots,T-1}).\end{aligned}$$

The first part of the cost-to-go function, $\boldsymbol{x}_t^T Q_t \boldsymbol{x}_t + \boldsymbol{z}_t^T R_t \boldsymbol{z}_t$ is the system's current cost at time t and the second part is the expected cost-to-go function cost at time $t+1$.

Similarly, we have the optimal cost-to-go function as the cost-to-go function under optimal control signals, i.e., $\mathcal{J}^*(\boldsymbol{x}_t, \boldsymbol{x}^g) = \min_{\boldsymbol{z}_{t,\dots,T-1}} \mathcal{J}(\boldsymbol{x}_t, \boldsymbol{x}^g, \boldsymbol{z}_{t,\dots,T-1})$. By mathematical induction, we will now show that the $\mathcal{J}^*(\boldsymbol{x}_t, \boldsymbol{x}^g)$ is a quadratic function on $\boldsymbol{x}_t$ without a linear term, i.e.,

$$\mathcal{J}^*(\boldsymbol{x}_t, \boldsymbol{x}^g) = \boldsymbol{x}_t^T P_t \boldsymbol{x}_t + r_t.$$

For $t = T$, the result is trivial as the control signals at $t = T$ just need to be set to zero. In this case, the optimal cost to go is the current cost, $\mathcal{J}^*(\boldsymbol{x}_T, \boldsymbol{x}^g) = \boldsymbol{x}_T^T Q_T \boldsymbol{x}_T$. Assuming

$\mathcal{J}^*(\boldsymbol{x}_{t+1}, \boldsymbol{x}^g) = \boldsymbol{x}_{t+1}^T P_{t+1} \boldsymbol{x}_{t+1} + r_{t+1}$, we have

$$\begin{aligned}
\mathcal{J}^*(\boldsymbol{x}_t, \boldsymbol{x}^g) &= \min_{\boldsymbol{z}_{t,\dots,T-1}} \mathcal{J}(\boldsymbol{x}_t, \boldsymbol{x}^g, \boldsymbol{z}_{t,\dots,T-1}) \\
&= \min_{\boldsymbol{z}_{t,\dots,T-1}} \left(\boldsymbol{x}_t^T Q_t \boldsymbol{x}_t + \boldsymbol{z}_t^T R_t \boldsymbol{z}_t + \mathbb{E}_{\boldsymbol{x}_{t+1}} \mathcal{J}(\boldsymbol{x}_{t+1}, \boldsymbol{x}^g, \boldsymbol{z}_{t+1,\dots,T-1})\right) \\
&= \min_{\boldsymbol{z}_t} \left(\boldsymbol{x}_t^T Q_t \boldsymbol{x}_t + \boldsymbol{z}_t^T R_t \boldsymbol{z}_t + \mathbb{E}_{\boldsymbol{x}_{t+1}} \min_{\boldsymbol{z}_{t+1,\dots,T-1}} \mathcal{J}(\boldsymbol{x}_{t+1}, \boldsymbol{x}^g, \boldsymbol{z}_{t+1,\dots,T-1})\right) \\
&= \min_{\boldsymbol{z}_t} \left(\boldsymbol{x}_t^T Q_t \boldsymbol{x}_t + \boldsymbol{z}_t^T R_t \boldsymbol{z}_t + \mathbb{E}_{\boldsymbol{x}_{t+1}} \mathcal{J}^*(\boldsymbol{x}_{t+1}, \boldsymbol{x}^g)\right) \\
&= \min_{\boldsymbol{z}_t} \left(\boldsymbol{x}_t^T Q_t \boldsymbol{x}_t + \boldsymbol{z}_t^T R_t \boldsymbol{z}_t + \mathbb{E}_{\boldsymbol{x}_{t+1}} \left(\boldsymbol{x}_{t+1}^T P_{t+1} \boldsymbol{x}_{t+1} + r_{t+1}\right)\right) \\
&= \min_{\boldsymbol{z}_t} \Bigg(\boldsymbol{x}_t^T Q_t \boldsymbol{x}_t + \boldsymbol{z}_t^T R_t \boldsymbol{z}_t + (H\boldsymbol{x}_t + M\boldsymbol{z}_t)^T P_{t+1} (H\boldsymbol{x}_t + M\boldsymbol{z}_t) \\
&\quad + \boldsymbol{z}_t^T \left(\sum_{i=1}^n \kappa_i I_i M^T P_{t+1} M I_i\right) \boldsymbol{z}_t + \mathrm{tr}(P_{t+1} M W M^T) + r_{t+1}\Bigg) \\
&= \min_{\boldsymbol{z}_t} \Big(\boldsymbol{z}_t^T D_t \boldsymbol{z}_t + 2\boldsymbol{x}_t^T (H^T P_{t+1} M) \boldsymbol{z}_t + \boldsymbol{x}_t^T (Q_t + H^T P_{t+1} H) \boldsymbol{x}_t \\
&\quad + \mathrm{tr}(P_{t+1} M W_t M^T) + r_{t+1}\Big).
\end{aligned} \tag{B.1}$$

where $D_t = R_t + M^T P_{t+1} M + \sum_{i=1}^n \kappa_i I_i M^T P_{t+1} M I_i$. By minimizing Eqn. B.1, we have the optimal control signal as

$$\boldsymbol{z}_t^* = -D_t^{-1} M^T P_{t+1} H \boldsymbol{x}_t = L_t \boldsymbol{x}_t,$$

where $L_t = -D_t^{-1} M^T P_{t+1} H$. Therefore, take $\boldsymbol{z}_t^*$ back into Eqn. B.1, the optimal cost-to-go function becomes

$$\begin{aligned}
\mathcal{J}^*(\boldsymbol{x}_t, \boldsymbol{x}^g) &= \boldsymbol{x}_t^T \left(Q_t + H^T P_{t+1}(H + M L_t)\right) \boldsymbol{x}_t + \mathrm{tr}(P_{t+1} M W M^T) + r_{t+1} \\
&= \boldsymbol{x}_t^T P_t \boldsymbol{x}_t + r_t,
\end{aligned}$$

where $P_t = Q_t + H^T P_{t+1}(H + M L_t)$ and $r_t = \mathrm{tr}(P_{t+1} M W M^T) + r_{t+1}$. So we proved the optimal cost-to-go function $\mathcal{J}^*(\boldsymbol{x}_t, \boldsymbol{x}^g)$ is a quadratic function on $\boldsymbol{x}_t$. By keeping track of D_t, L_t and P_t, we can compute the optimal control signal for each state and the minimum cost under optimal policy as

$$\mathcal{J}^*(\boldsymbol{x}_0, \boldsymbol{x}^g) = \boldsymbol{x}_0^T P_0 \boldsymbol{x}_0 + \sum_{t=0}^{T-1} \mathrm{tr}\left(P_{t+1} M W M^T\right).$$

B.2 Changes in modulation depth do not affect the P_t term in Algorithm 1

In this appendix we prove that the quadratic cost-to-go variable, P_t, introduced in Algorithm 1 is unchanged by a change in neural modulation depth. Recall that we capture changes in modulation depth through the diagonal matrix Γ, which scales the mapping between control signal and cursor velocity (i.e., $M_v \to \Gamma M_v$). To study the change of P_t after scaling M_v, we treat P_t as a function of Γ, denoting as $P_t(\Gamma)$. By using mathematical induction, we prove that the magnitude change of M_v will not affect P_t, i.e., $P_t(\Gamma) = P_t(I)$. For notation simplicity, we still use P_t, D_t and L_t to represent $P_t(I)$, $D_t(I)$ and $L_t(I)$ respectively.

At time T, the solution is trivial since $P_T(\Gamma) = Q_T$ and Q_T doesn't depend on Γ. By our induction assumption, $P_{t+1}(\Gamma) = P_{t+1}$. Also, note that since both Γ and I_i are diagonal, $\Gamma I_i = I_i \Gamma$. Therefore, at time t we have,

$$\begin{aligned} D_t(\Gamma) &= R_t(\Gamma) + \Gamma M^T P_{t+1}(\Gamma) M \Gamma + \sum\nolimits_{i=1}^{n} \kappa_i I_i \Gamma M^T P_{t+1}(\Gamma) M \Gamma I_i \\ &= \lambda_u \Gamma M^T M \Gamma + \Gamma M^T P_{t+1} M \Gamma + \sum\nolimits_{i=1}^{n} \kappa_i I_i \Gamma M^T P_{t+1} M \Gamma I_i \\ &= \Gamma \left(\lambda_u M^T M + M^T P_{t+1} M + \sum\nolimits_{i=1}^{n} \kappa_i I_i M^T P_{t+1} M I_i \right) \Gamma \\ &= \Gamma D_t \Gamma, \end{aligned}$$

$$L_t(\Gamma) = -D_t(\Gamma)^{-1} \Gamma M^T P_{t+1}(\Gamma) H = -\Gamma^{-1} D_t^{-1} M^T P_{t+1} H = \Gamma^{-1} L_t,$$

and

$$P_t(\Gamma) = Q_t + H^T P_{t+1}(\Gamma)\big(H + M\Gamma L_t(\Gamma)\big) = Q_t + H^T P_{t+1}(H + M L_t) = P_t$$

Thus, we have proved $P_t(\Gamma) = P_t$ for $0 \leq t \leq T$. Meanwhile, the optimal policy becomes $L_t(\Gamma) = \Gamma^{-1} L_t$ and the optimal cost becomes

$$\begin{aligned} \mathcal{J}^*(\Gamma) &= \operatorname{tr}\big(P_0(\Gamma) X_0\big) + \sum\nolimits_{t=0}^{T-1} \operatorname{tr}\left(P_{t+1}(\Gamma) M \Gamma W \Gamma M^T\right) \\ &= \operatorname{tr}(P_0 X_0) + \sum\nolimits_{t=0}^{T-1} \operatorname{tr}\left(P_{t+1} M \Gamma W \Gamma M^T\right). \end{aligned}$$

B.3 Solving for the optimal distribution of pushing directions

Recall in Eqn. 4.21 we showed that the optimal solution for the distribution of pushing directions could be written as $\min_{M_v} \mathbb{E}_{\boldsymbol{u}^*} \boldsymbol{u}^{*T} (M_v M_v^T)^{-1} \boldsymbol{u}^*$. Here we derive the solution to this problem.

First, realize we can rewrite the objective function as:

$$\begin{aligned} \mathbb{E}_{\boldsymbol{u}^*} \boldsymbol{u}^{*T} (M_v M_v^T)^{-1} \boldsymbol{u}^* =& \mathbb{E}_{\boldsymbol{u}^*} \operatorname{tr}\left(\boldsymbol{u}^* \boldsymbol{u}^{*T} (M_v M_v^T)^{-1}\right) \\ =& \operatorname{tr}\left((\mathbb{E}_{\boldsymbol{u}^*} \boldsymbol{u}^* \boldsymbol{u}^{*T})(M_v M_v^T)^{-1}\right) \\ =& 1/d \operatorname{tr}\left((M_v M_v^T)^{-1}\right). \end{aligned}$$

The last equation is due to the fact that when $\boldsymbol{u}^*$ satisfies uniform spherical distribution, the mean of $\boldsymbol{u}^*$ is $\mathbf{0}$ and the covariance matrix $\mathbb{E}_{\boldsymbol{u}^*} \boldsymbol{u}^* \boldsymbol{u}^{*T}$ is $1/dI$ [Marsaglia, 1972].

Notice that $M_v M_v^T$ is a positive-definite matrix, thus all its eigenvalues are positive. Denote the eigenvalues of $M_v M_v^T$ as $\lambda_1, \ldots, \lambda_d$, then the eigenvalues of $(M_v M_v^T)^{-1}$ are $\lambda_1^{-1}, \ldots, \lambda_d^{-1}$

and $\mathrm{tr}\left((M_vM_v^T)^{-1}\right) = \sum_{j=1}^{d} \lambda_j^{-1}$. Thus the objective function is

$$\begin{aligned} \min_{M_v} \quad & \sum\nolimits_{j=1}^{d} \lambda_j^{-1} \\ \text{s.t.} \quad & \|\boldsymbol{m}_{v,i}\| = 1. \end{aligned}$$

Since $\lambda_i > 0$, from the relationship between the harmonic mean and the arithmetic mean we have $\sum_{j=1}^{d} \lambda_j^{-1} \geq d^2 / \sum_{j=1}^{d} \lambda_j$. Further, since

$$\begin{aligned} \sum\nolimits_{j=1}^{d} \lambda_j &= \mathrm{tr}(M_vM_v^T) \\ &= \mathrm{tr}\left(\sum\nolimits_{i=1}^{n} \boldsymbol{m}_{v,i}\boldsymbol{m}_{v,i}^T\right) \\ &= \sum\nolimits_{i=1}^{n} \mathrm{tr}\left(\boldsymbol{m}_{v,i}\boldsymbol{m}_{v,i}^T\right) \\ &= \sum\nolimits_{i=1}^{n} \mathrm{tr}\left(\boldsymbol{m}_{v,i}^T\boldsymbol{m}_{v,i}\right) \\ &= n, \end{aligned}$$

we get the lower bound for $\sum_{j=1}^{d} \lambda_j^{-1}$ as d^2/n. It reach the lower bound if and only if $\lambda_1 = \lambda_2 = \ldots = \lambda_d = n/d$. It is not difficult to prove that when $n = 2$, the only solution is that $\boldsymbol{m}_{v,1}$ and $\boldsymbol{m}_{v,2}$ are orthogonal, so $\lambda_1 = \lambda_2 = 1$. For $n > 2$, one solution is that $\boldsymbol{m}_{v,i}$ is uniformly distributed on the circle. Therefore, $M_vM_v^T = n/dI$ and $\lambda_1 = \lambda_2 = \ldots = \lambda_d = n/d$.

Bibliography

P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceeding of the 21st International Conference on Machine Learning*, pages 1–8, 2004.

T. Aflalo, S. Kellis, C. Klaes, B. Lee, Y. Shi, K. Pejsa, K. Shanfield, S. Hayes-Jackson, M. Aisen, C. Heck, C. Liu, and R. A. Andersen. Decoding motor imagery from the posterior parietal cortex of a tetraplegic human. *Science*, 348(6237):906–910, 2015.

M. Ahissar and S. Hochstein. Task difficulty and the specificity of perceptual learning. *Nature*, 387(22):401–406, 1997.

A Bolu Ajiboye, Francis R Willett, Daniel R Young, William D Memberg, Brian A Murphy, Jonathan P Miller, Benjamin L Walter, Jennifer A Sweet, Harry A Hoyen, Michael W Keith, et al. Restoration of reaching and grasping movements through brain-controlled muscle stimulation in a person with tetraplegia: a proof-of-concept demonstration. *The Lancet*, 389(10081): 1821–1830, 2017.

B. Amirikian and A. P. Georgopulos. Directional tuning profiles of motor cortical cells. *Neuroscience Research*, 36(1):73–79, 2000.

B. D. O. Anderson and J. B. Moore. *Optimal control: linear quadratic methods*. Prentice-Hall, NJ, 1989.

P.-J. Arduin, Y. Frégnac, D. E. Shulz, and V. Ego-Stengel. "Master" neurons induced by operant conditioning in rat motor cortex during a brain-machine interface task. *Journal of Neuroscience*, 33(19):8308–8320, 2013.

J. Ashe and A. P. Georgopoulos. Movement parameters and neural activity in motor cortex and area 5. *Cerebral Cortex*, 4(6):590–600, 1994.

M. J. Black, E. Bienenstock, J. P. Donoghue, M. Serruya, W. Wu, and Y. Gao. Connecting brains with machines: the neural control of 2d cursor movement. In *Proceedings of the 1st International IEEE EMBS Conference on Neural Engineering*, pages 580–583, 2003.

C. E. Bouton, A. Shaikhouni, N. V. Annetta, M. A. Bockbrader, D. A. Friedenberg, D. M. Nielson, G. Sharma, P. B. Sederberg, B. C. Glenn, W. J. Mysiw, and A. G. Morgan. Restoring cortical control of functional movement in a human with quadriplegia. *Nature*, 533(7602): 247–250, 2016.

A. E. Brockwell, A. Rojas, and R. E. Kass. Recursive bayesian decoding of motor cortical signals by particle filtering. *Journal of Neurophysiology*, 91(4):1899–1907, 2004.

E. N. Brown, L. M. Frank, D. Tang, M. C. Quirk, and M. A. Wilson. A statistical paradigm for neural spike train decoding applied to position prediction from ensemble firing patterns of rat hippocampal place cells. *Journal of Neuroscience*, 18(18):7411–7425, 1998.

E. N. Brown, R. Barbieri, V. Ventura, R. E Kass, and L. M Frank. The time-rescaling theorem and its application to neural spike train data analysis. *Neural Computation*, 14(2):325–346, 2002.

R. Caminiti, P. B. Johnson, and A. Urbano. Making arm movements within different parts of space: dynamic aspects in the primate motor cortex. *Journal of Neuroscience*, 10(7):2039–2058, 1990.

R. Caminiti, P. B. Johnson, C. Galli, S. Ferraina, and Y. Burnod. Making arm movements within different parts of space: the premotor and motor cortical representation of a coordinate system for reaching to visual targets. *Journal of Neuroscience*, 11(5):1182–1197, 1991.

J. M. Carmena, M. A. Lebedev, R. E. Crist, J. E. O'Doherty, D. M. Santucci, D. F. Dimitrov, P. G. Patil, C. S. Henriquez, and M. A. L. Nicolelis. Learning to control a brain-machine interface for reaching and grasping by primates. *PLoS Biology*, 1(2):e42, 2003.

J. K. Chapin, K. A. Moxon, R. S. Markowitz, and M. A. Nicolelis. Real-time control of a robot arm using simultaneously recorded neurons in the motor cortex. *Nature Neuroscience*, 2(7): 664–70, 1999.

S. M. Chase and A. B. Schwartz. Inference from populations: going beyond models. *Progress in Brain Research*, 192:103–112, 2011.

S. M. Chase, A. B. Schwartz, and R. E. Kass. Bias, optimal linear estimation, and the differences between open-loop simulation and closed-loop performance of spiking-based brain-computer interface algorithms. *Neural Networks*, 22(9):1203–1213, 2009.

S. M. Chase, R. E. Kass, and A. B. Schwartz. Behavioral and neural correlates of visuomotor adaptation observed through a brain-computer interface in primary motor cortex. *Journal of Neurophysiology*, 108(2):624–644, 2012.

X. Chen, Y. Wang, M. Nakanishi, X. Gao, T.-P. Jung, and S. Gao. High-speed spelling with a noninvasive brain-computer interface. *Proceedings of the National Academy of Sciences*, 112 (44):E6058–E6067, 2015.

C. A. Chestek, J. P. Cunningham, V. Gilja, P. Nuyujukian, S. I. Ryu, and K. V. Shenoy. Neural prosthetic systems: current problems and future directions. In *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 3369–3375, 2009.

P. Y. Chhatbar and J. T. Francis. Towards a naturalistic brain-machine interface: Hybrid torque and position control allows generalization to novel dynamics. *PloS ONE*, 8(1):e52286, 2013.

Christopher and Dana Reeve Foundation. *One Degree of Separation: Paralysis and Spinal Cord Injury in the United State*. Short Hills, NJ, 2009.

C. K. Chui and G. Chen. *Kalman Filtering: with Real-Time Applications*. Springer, Berlin, 4th edition, 2009.

M. M. Churchland and K. V. Shenoy. Temporal complexity and heterogeneity of single-neuron activity in premotor and motor cortex. *Journal of Neurophysiology*, 97(6):4235–4257, 2007.

M. M. Churchland, B. M. Yu, S. I. Ryu, G. Santhanam, and K. V. Shenoy. Neural variability in premotor cortex provides a signature of motor preparation. *Journal of Neuroscience*, 26(14): 3697–3712, 2006.

M. M. Churchland, J. P. Cunningham, M. T. Kaufman, J. D. Foster, P. Nuyujukian, S. I. Ryu, and K. V. Shenoy. Neural population dynamics during reaching. *Nature*, 487(7405):51–56, 2012.

J. L. Collinger, B. Wodlinger, John E Downey, W. Wang, E. C. Tyler-Kabara, D. J. Weber, A. J.C. McMorland, M. Velliste, M. L. Boninger, and A. B. Schwartz. High-performance neuroprosthetic control by an individual with tetraplegia. *Lancet*, 381(9866):557–564, 2013.

T. B. Crapse and M. A. Sommer. Corollary discharge circuits in the primate brain. *Current Opinion in Neurobiology*, 18(6):552–557, 2008.

S. Dangi, S. Gowda, H. G. Moorman, A. L. Orsborn, K. So, M. Shanechi, and J. M. Carmena. Continuous closed-loop decoder adaptation with a recursive maximum likelihood algorithm allows for rapid performance acquisition in brain-machine interfaces. *Neural Computation*, 26(9):1811–1839, 2014.

A. G. Davidson, V. Chan, R. O'Dell, and M. H. Schieber. Rapid changes in throughput from single motor cortex neurons to muscle activity. *Science*, 318(5858):1934–1937, 2007.

A. de Rugy, G. E. Loeb, and T. J. Carroll. Muscle coordination is habitual rather than optimal. *Journal of Neuroscience*, 32(21):7384–7391, 2012.

J. Dethier, P. Nuyujukian, C. Eliasmith, T. C. Stewart, S. A. Elasaad, K. V. Shenoy, and K. A. Boahen. A brain-machine interface operating with a real-time spiking neural network control algorithm. In *Advances in Neural Information Processing Systems*, pages 2213–2221. 2011.

J. Diedrichsen. Optimal task-dependent changes of bimanual feedback control and adaptation. *Current Biology*, 17(19):1675–1679, 2007.

J. Diedrichsen, R. Shadmehr, and R. B. Ivry. The coordination of movement: optimal feedback control and beyond. *Trends in Cognitive Sciences*, 14(1):31–39, 2010.

A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

J. E. Downey, L. Brane, R. A. Gaunt, Elizabeth T.-K., M. L. Boninger, and J. L. Collinger. Motor cortical activity changes during neuroprosthetic-controlled object interaction. *Scientific Reports*, 7(1):16947, 2017.

J. E. Downey, N. Schwed, S. M. Chase, A. B. Schwartz, and J. L. Collinger. Intracortical recording stability in human brain–computer interface users. *Journal of Neural Engineering*, 15(4): 046016, 2018.

K. Dvijotham and E. Todorov. Inverse optimal control with linearly-solvable mdps. In *Proceedings of the 27th International Conference on Machine Learning*, pages 335–342, 2010.

U. T. Eden, L. M. Frank, R. Barbieri, V. Solo, and E. N. Brown. Dynamic analyses of neural encoding by point process adaptive filtering. 16(5):971–998, 2004.

T. Elbert, B. Rockstroh, W. Lutzenberger, and N. Birbaumer. Biofeedback of slow cortical potentials. i. *Electroencephalography and Clinical Neurophysiology*, 48(3):293–301, 1980.

A. Ergün, R. Barbieri, U. T. Eden, M. A. Wilson, and E. N. Brown. Construction of point process adaptive filter algorithms for neural systems using sequential monte carlo methods. *IEEE Transactions on Biomedical Engineering*, 54(3):419–428, 2007.

C. Ethier, E. R. Oby, M. J. Bauman, and L. E. Miller. Restoration of grasp following paralysis through brain-controlled stimulation of muscles. *Nature*, 485(7398):368, 2012.

E. V. Evarts. Relation of pyramidal tract activity to force exerted during voluntary movement. *Journal of Neurophysiology*, 31(1):14–27, 1968.

A. H. Fagg, A. Shah, and A. G. Barto. A computational model of muscle recruitment for wrist movements. *Journal of Neurophysiology*, 88(6):3348–3358, 2002.

A. H. Fagg, G. W. Ojakangas, L. E. Miller, and N. G. Hatsopoulos. Kinetic trajectory decoding using motor cortical ensembles. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 17(5):487–496, 2009.

J. M. Fan, P. Nuyujukian, J. C. Kao, C. A. Chestek, S. I. Ryu, and K. V. Shenoy. Intention estimation in brain-machine interfaces. *Journal of Neural Engineering*, 11(1):016004, 2014.

L. A. Farwell and E. Donchin. Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and Clinical Neurophysiology*, 70(6):510–523, 1988.

E. E. Fetz. Operant conditioning of cortical unit activity. *Science*, 163(3870):955–958, 1969.

E. E. Fetz and D. V. Finocchio. Operant conditioning of specific patterns of neural and muscular activity. *Science*, 174(4007):431–435, 1971.

S. N. Flesher, J. L. Collinger, S. T. Foldes, J. M. Weiss, J. E. Downey, E. C. Tyler-Kabara, S. J. Bensmaia, A. B. Schwartz, M. L. Boninger, and R. A. Gaunt. Intracortical microstimulation of human somatosensory cortex. *Science Translational Medicine*, 8(361):361ra141–361ra141, 2016.

R. D. Flint, Z. A. Wright, M. R. Scheid, and M. W. Slutzky. Long term, stable brain machine interface performance using local field potentials and multiunit spikes. *Journal of Neural Engineering*, 10(5):056005, 2013.

G. W. Fraser and A. B. Schwartz. Recording from the same neurons chronically in motor cortex. *Journal of Neurophysiology*, 107(7):1970–1978, 2012.

G. W. Fraser, S. M. Chase, A. Whitford, and A. B. Schwartz. Control of a brain-computer interface without spike sorting. *Journal of Neural Engineering*, 6(5):055004, 2009.

Q.-G. Fu, J. I. Suarez, and T. J. Ebner. Neuronal specification of direction and distance during reaching movements in the superior precentral premotor area and primary motor cortex of monkeys. *Journal of Neurophysiology*, 70(5):2097–2116, 1993.

Q.-G. Fu, D. Flament, J. D. Coltz, and T. J. Ebner. Temporal encoding of movement kinematics in the discharge of primate primary motor and premotor neurons. *Journal of Neurophysiology*, 73(2):836–854, 1995.

F. Gandolfo, C. S. Li, B. J. Benda, C. P. Schioppa, and E. Bizzi. Cortical correlates of learning in monkeys adapting to a new dynamical environment. *Proceedings of the National Academy* *of Sciences*, 97(5):2259–2263, 2000.

K. Ganguly and J. M. Carmena. Emergence of a stable cortical map for neuroprosthetic control. *PLoS biology*, 7(7):e1000153, 2009.

K. Ganguly and J. M. Carmena. Neural correlates of skill acquisition with a cortical brain machine interface. *Journal of Motor Behavior*, 42(6):355–360, 2010.

K. Ganguly, D. F. Dimitrov, J. D. Wallis, and J. M. Carmena. Reversible large-scale modification of cortical networks during neuroprosthetic control. *Nature Neuroscience*, 14(5):662–667, 2011.

A. P. Georgopoulos, J. F. Kalaska, R. Caminiti, and J. T. Massey. On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex. *Journal of Neuroscience*, 2(11):1527–1537, 1982.

A. P. Georgopoulos, A. B. Schwartz, and R. E. Kettner. Neuronal population coding of movement direction. *Science*, 233(4771):1416–1419, 1986.

A. P. Georgopoulos, R. E. Kettner, and A. B. Schwartz. Primate motor cortex and free arm movements to visual targets in three-dimensional space. II. Coding of the direction of movement by a neuronal population. *Journal of Neuroscience*, 8(8):2928–2937, 1988.

A. P. Georgopoulos, J. Ashe, N. Smyrnis, and M. Taira. The motor cortex and the coding of force. *Science*, 256(5064):1692–1695, 1992.

V. Gilja, C. A. Chestek, I. Diester, J. M. Henderson, K. Deisseroth, and K. V. Shenoy. Challenges and opportunities for next-generation intracortically based neural prostheses. *IEEE Transactions on Biomedical Engineering*, 58(7):1891–1899, 2011.

V. Gilja, P. Nuyujukian, C. A. Chestek, J. P. Cunningham, B. M. Yu, J. M. Fan, M. M. Churchland, M. T. Kaufman, J. C. Kao, S. I. Ryu, and K. V. Shenoy. A high-performance neural prosthesis enabled by control algorithm design. *Nature Neuroscience*, 15(12):1752–1757, 2012.

V. Gilja, C. Pandarinath, C. H. Blabe, P. Nuyujukian, J. D. Simeral, A. A. Sarma, B. L. Sorice, J. A. Perge, B. Jarosiewicz, L. R. Hochberg, K. V. Shenoy, and J. M. Henderson. Clinical translation of a high-performance neural prosthesis. *Nature Medicine*, 21(10):1142–1145, 2015.

M. J. Giummarra, S. J. Gibson, N. Georgiou-Karistianis, and J. L. Bradshaw. Mechanisms underlying embodiment, disembodiment and loss of embodiment. *Neuroscience and Biobehavioral Reviews*, 32(1):143–160, 2008.

R. Goebel, B. Sorger, J. Kaiser, N. Birbaumer, and N. Weiskopf. Bold brain pong: Self regulation of local brain activity during synchronously scanned, interacting subjects. In *Annual Meeting of the Society for Neuroscience*, pages 23–27, 2004.

R. Goebel, A. Zilverstand, and B. Sorger. Real-time fmri-based brain–computer interfacing for neurofeedback therapy and compensation of lost motor functions. *Imaging*, 2(4):407–415, 2010.

M. D. Golub, S. M. Chase, and B. M. Yu. Internal models engaged by brain-computer interface control. In *Proceedings of the 35th Annual International Conference of the IEEE Engineering* *in Medicine and Biology Society*, pages 1327–1330, 2012.

M. D. Golub, S. M. Chase, and B. M. Yu. Learning an internal dynamics model from control demonstration. In *Proceedings of the 30th International Conference on Machine Learning*, pages 606–614, 2013.

M. D. Golub, B. M. Yu, A. B. Schwartz, and S. M. Chase. Motor cortical control of movement speed with implications for brain-machine interface control. *Journal of Neurophysiology*, 112 (2):411–429, 2014.

M. D. Golub, M. Y. Byron, and S. M. Chase. Internal models for interpreting neural population activity during sensorimotor control. *Elife*, 4:e10015, 2015.

M. D. Golub, P. T. Sadtler, E. R. Oby, K. M. Quick, S. I. Ryu, E. C. Tyler-Kabara, A. P. Batista, S. M. Chase, and B. M. Yu. Learning by neural reassociation. *Nature Neuroscience*, 21: 607–616, 2018.

S. Gowda, A. L. Orsborn, S. A. Overduin, H. G. Moorman, and J. M. Carmena. Designing dynamical properties of brain-machine interfaces to optimize task-specific performance. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 22(5):911–920, 2014.

M. A. Guadagnoli and T. D. Lee. Challenge point: A framework for conceptualizing the effects of various practice conditions in motor learning. *Journal of Motor Behavior*, 36(2):212–224, 2004.

R. Gupta and J. Ashe. Offline decoding of end-point forces using neural ensembles: application to a brain–machine interface. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 17(3):254–262, 2009.

M. Haruno and D. M. Wolpert. Optimal control of redundant muscles in step-tracking wrist movements. *Journal of Neurophysiology*, 94(6):4244–4255, 2005.

N. Hatsopoulos, J. Joshi, and J. G. O'Leary. Decoding continuous and discrete motor behaviors using motor and premotor cortical ensembles. *Journal of Neurophysiology*, 92(2):1165–1174, 2004.

N. G. Hatsopoulos and J. P. Donoghue. The science of neural interface systems. *Annual Review of Neuroscience*, 32:249–266, 2009.

R. Héliot, K. Ganguly, J. Jimenez, and J. M. Carmena. Learning in closed-loop brain–machine interfaces: modeling and experimental validation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 40(5):1387–1397, 2010.

J. A. Hennig, M. D. Golub, P. J. Lund, P. T. Sadtler, K. M. Quick, S. I. Ryu, E. C. Tyler-Kabara, A. P. Batista, S. M. Chase, and B. M. Yu. Predicting neural activity in behaviorally-irrelevant dimensions. In *Computational and Systems Neuroscience*, 2017. Poster no. II-14.

M. C. Hepp-Reymond, U. R. Wyss, and R. Anner. Neuronal coding of static force in the primate motor cortex. *Journal De Physiologie*, 74(3):287, 1978.

M. C. Hepp-Reymond, M. Kirkpatrick-Tanner, L. Gabernet, H. X. Qi, and B. Weber. Context-dependent force coding in motor and premotor cortical areas. *Experimental Brain Research*, 128(1-2):123–133, 1999.

L. R. Hochberg, M. D. Serruya, G. M. Friehs, J. A. Mukand, M. Saleh, A. H. Caplan, A. Branner, D. Chen, R. D. Penn, and J. P. Donoghue. Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature*, 442(7099):164–171, 2006.

L. R. Hochberg, D. Bacher, B. Jarosiewicz, N. Y. Masse, J. D. Simeral, J. Vogel, S. Haddadin, J. Liu, P. van der Smagt, and J. P. Donoghue. Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature*, 485(7398):372–375, 2012.

N. Hogan. The mechanics of multi-joint posture and movement control. *Biological Cybernetics*, 52(5):315–331, 1985.

N. Hogan and D. Sternad. Dynamic primitives of motor behavior. *Biological Cybernetics*, 106 (11–12):727–739, 2012.

M. L. Homer, M. T. Harrison, M. J. Black, J. A. Perge, S. S. Cash, G. Friehs, and L. R. Hochberg. Mixing decoded cursor velocity and position from an offline Kalman filter improves cursor control in people with tetraplegia. In *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 715–718, 2013.

D. R. Humphrey, E. M. Schmidt, and W. D. Thompson. Predicting measures of motor performance from multiple cortical spike trains. *Science*, 170(3959):758–762, 1970.

E. J. Hwang and R. A. Andersen. The utility of multichannel local field potentials for brain-machine interfaces. *Journal of Neural Engineering*, 10(4):046005, 2013.

B. Jarosiewicz, S. M. Chase, G. W. Fraser, M. Velliste, R. E. Kass, and A. B. Schwartz. Functional network reorganization during learning in a brain-computer interface paradigm. *Proceedings of the National Academy of Sciences*, 105(49):19486–19491, 2008.

J. F. Kalaska, D. A. Cohen, M. L. Hyde, and M. Prud'homme. A comparison of movement direction-related versus load direction-related activity in primate motor cortex, using a two-dimensional reaching task. *Journal of Neuroscience*, 9(6):2080–2102, 1989.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1):35–45, 1960.

J. C. Kao, P. Nuyujukian, S. I. Ryu, M. M. Churchland, J. P. Cunningham, and K. V. Shenoy. Single-trial dynamics of motor cortex and their applications to brain-machine interfaces. *Nature Communications*, 6, 2015.

R. E. Kass and V. Ventura. A spike-train probability model. *Neural Computation*, 13(8):1713–1720, 2001.

R. E. Kass, V. Ventura, and E. N. Brown. Statistical issues in the analysis of neuronal data. *Journal of Neurophysiology*, 94(1):8–25, 2005.

M. Kawato. Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology*, 9(6):718–727, 1999.

H. K. Kim, J. M. Carmena, S. J. Biggs, T. L. Hanson, M. A. L. Nicolelis, and M. A. Srinivasan. The muscle activation method: an approach to impedance control of brain-machine interfaces through a musculoskeletal model of the arm. *IEEE Transactions on Biomedical Engineering*, 54(8):1520–1529, 2007.

S. Kim, T. Callier, G. A. Tabot, R. A. Gaunt, F. V. Tenore, and S. J. Bensmaia. Behavioral assessment of sensitivity to intracortical microstimulation of primate somatosensory cortex. *Proceedings of the National Academy of Sciences*, 112(49):15202–15207, 2015.

S.-P. Kim, J. D. Simeral, L. R. Hochberg, J. P. Donoghue, and M. J. Black. Neural control of computer cursor velocity by decoding motor cortical spiking activity in humans with tetraplegia. *Journal of Neural Engineering*, 5(4):455–476, 2008.

G. Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

K. C. Kowalski, B. D. He, and L. Srinivasan. Dynamic analysis of naive adaptive brain-machine interfaces. *Neural Computation*, 25(9):2373–2420, 2013.

S. Koyama, S. M. Chase, A. S. Whitford, M. Velliste, A. B. Schwartz, and R. E. Kass. Comparison of brain-computer interface decoding algorithms in open-loop and closed-loop control. *Journal of Computational Neuroscience*, 29(1–2):73–87, 2010a.

S. Koyama, L. C. Pérez-Bolde, C. R. Shalizi, , and R. E. Kass. Approximate methods for state-space models. *Journal of the American Statistical Association*, 105(489):170–180, 2010b.

J. W. Krakauer, Z. M. Pine, M. F. Ghilardi, and C. Ghez. Learning of visuomotor transformations for vectorial planning of reaching trajectories. *Journal of Neuroscience*, 20(23):8916–8924, 2000.

H. Kwakernaak and R. Sivan. *Linear Optimal Control Systems*. Wiley-Interscience, New York, 1st edition, 1972.

M. A. Lebedev and M. Nicolelis. Brain-machine interfaces: past, present and future. *Trends in Neurosciences*, 29(9):536–546, 2006.

M. A. Lebedev and M. A. L. Nicolelis. Brain-machine interfaces: From basic science to neuroprostheses and neurorehabilitation. *Physiological Reviews*, 97(2):767–837, 2017.

J.-H. Lee, J. Ryu, F. A. Jolesz, Z.-H. Cho, and S.-S. Yoo. Brain–machine interface via real-time fmri: preliminary study on thought-controlled robotic arm. *Neuroscience Letters*, 450(1):1–6, 2009.

R. Legenstein, S. M. Chase, A. B. Schwartz, and W. Maass. A reward-modulated hebbian learning rule can explain experimentally observed network reorganization in a brain control task. *Journal of Neuroscience*, 30(25):8400–8410, 2010.

C.-S. R. Li, C. Padoa-Schioppa, and E. Bizzi. Neuronal correlates of motor performance and motor learning in the primary motor cortex of monkeys adapting to an external force field. *Neuron*, 30(2):593–607, 2001.

Z. Li, J. E. O'Doherty, T. L. Hanson, M. A. Lebedev, C. S Henriquez, and M. A.L. Nicolelis. Unscented Kalman filter for brain-machine interfaces. *PloS ONE*, 4(7):e6243, 2009.

Z. Li, J. E. O'Doherty, M. A. Lebedev, and M. A. L. Nicolelis. Adaptive decoding for brain-machine interfaces through bayesian parameter updates. *Neural Computation*, 23(12):3162–3204, 2011.

M. D. Linderman, G. Santhanam, C. T. Kemere, V. Gilja, S. O'Driscoll, B. M. Yu, A. Afshar,

S. I. Ryu, K. V. Shenoy, and T. H. Meng. Signal processing challenges for neural prostheses. *IEEE Signal Processing Magazine*, 25(1):18–28, 2008.

D. Liu and E. Todorov. Evidence for the flexible sensorimotor strategies predicted by optimal feedback control. *Journal of Neuroscience*, 27(35):9354–9368, 2007.

X. Liu and R. A. Scheidt. Activity of the same motor cortex neurons during repeated experience with perturbed movement dynamics. *Journal of Neurophysiology*, 99(5):2546–2557, 2008.

X. Liu, K. M. Mosier, F. A. Mussa-Ivaldi, M. Casadio, and R. A. Scheidt. Reorganization of finger coordination patterns during adaptation to rotation and scaling of a newly learned sensorimotor transformation. *Journal of Neurophysiology*, 105(1):454–473, 2011.

G. E. Loeb. Optimal isn't good enough. *Biological Cybernetics*, 106(11):757–765, 2012.

N. K. Logothetis, J. Pauls, M. Augath, T. Trinath, and A. Oeltermann. Neurophysiological investigation of the basis of the fmri signal. *Nature*, 412(6843):150, 2001.

Z. L. Lu and B. A. Dosher. External noise distinguishes attention mechanisms. *Vision Research*, 38(9):1183–1198, 1998.

W. Q. Malik, W. Truccolo, E. N. Brown, and L. R. Hochberg. Efficient decoding with steady-state Kalman filter in neural interface systems. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 19(1):25–34, 2011.

Y. Mandelblat-Cerf, R. Paz, and E. Vaadia. Trial-to-trial variability of single cells in motor cortices is dynamically modified during visuomotor adaptation. *Journal of Neuroscience*, 29 (48):15053–15062, 2009.

Y. Mandelblat-Cerf, I. Novick, R. Paz, Y. Link, S. Freeman, and E. Vaadia. The neuronal basis of long-term sensorimotor learning. *Journal of Neuroscience*, 31(1):300–313, 2011.

A. R. Marathe and D. M. Taylor. Decoding position, velocity, or goal: Does it matter for brain–machine interfaces? *Journal of Neural Engineering*, 8(2):025016, 2011.

G. Marsaglia. Choosing a point from the surface of a sphere. *Annals of Mathematical Statistics*, 43(2):645–646, 1972.

B. T. Marsh, V. S. A. Tarigoppula, C. Chen, and J. T. Francis. Toward an autonomous brain machine interface: integrating sensorimotor reward modulation and reinforcement learning. *Journal of Neuroscience*, 35(19):7374–7387, 2015.

P. McCullagh and J. Nelder. *Generalized linear models*. Chapman and Hall, London, 1989.

D. B. McNiel, J. S. Choi, J. P. Hessburg, and J. T. Francis. Reward value is encoded in primary somatosensory cortex and can be decoded from neural activity during performance of a psychophysical task. In *Proceedings of the 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 3064–3067, 2016.

J. Merel, D. M. Pianto, J. P. Cunningham, and L. Paninski. Encoder-decoder optimization for brain-computer interfaces. *PLoS Computational Biology*, 11(6):e1004288, 2015.

J. Merel, D. Carlson, L. Paninski, and J. P. Cunningham. Neuroprosthetic decoder training as imitation learning. *PLoS Computational Biology*, 12(5):e1004948, 2016.

D. W. Moran and A. B. Schwartz. Motor cortical representation of speed and direction during reaching. *Journal of Neurophysiology*, 82(5):2676–2692, 1999a.

D. W. Moran and A. B. Schwartz. Motor cortical activity during drawing movements: population representation during spiral tracing. *Journal of Neurophysiology*, 82(5):2693–2704, 1999b.

C. T. Moritz and E. E. Fetz. Volitional control of single cortical neurons in a brainmachine interface. *Journal of Neural Engineering*, 8(2):025017, 2011.

C. T. Moritz, S. I. Perlmutter, and E. E. Fetz. Direct control of paralysed muscles by cortical neurons. *Nature*, 456:639–642, 2008.

K. M. Mosier, R. A. Scheidt, S. Acosta, and F. A. Mussa-Ivaldi. Remapping hand movements in a novel geometrical environment. *Journal of Neurophysiology*, 94(6):4362–4372, 2005.

G. H. Mulliken, S. Musallam, and R. A. Andersen. Decoding trajectories from posterior parietal cortex ensembles. *Journal of Neuroscience*, 28(48):12913–12926, 2008.

A. J. Nagengast, D. A. Braun, and D. M. Wolpert. Optimal control predicts human performance on objects with internal degrees of freedom. *PLoS Computational Biology*, 5(6):e1000419, 2009.

J. Y. Nashed, F. Crevecoeur, and S. H. Scott. Influence of the behavioral goal and environmental obstacles on rapid feedback responses. *Journal of Neurophysiology*, 108(4):999–1009, 2012.

K. Nazarpour, C. Ethier, L. Paninski, J. M. Rebesco, R. C. Miall, and L. E. Miller. Emg prediction from motor cortical recordings via a nonnegative point-process filter. *IEEE Transactions on Biomedical Engineering*, 59(7):1829–1838, 2012.

R. M. Neely, A. C. Koralek, V. R. Athalye, R. M. Costa, and J. M. Carmena. Volitional modulation of primary visual cortex activity requires the basal ganglia. *Neuron*, 97(6):1356–1368, 2018.

L. Nelson. The simultaneous on-line estimation of parameters and states in linear systems. *IEEE Transactions on Automatic Control*, 21(1):94–98, 1976.

A. Y. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *Proceeding of the 17th International Conference on Machine Learning*, pages 663–670, 2000.

Y. Nishimura, S. I. Perlmutter, and E. E. Fetz. Restoration of upper limb movement via artificial corticospinal and musculospinal connections in a monkey with spinal cord injury. *Frontiers in Neural Circuits*, 7, 2013.

E. R. Oby, C. Ethier, and L. E. Miller. Movement representation in the primary motor cortex and its contribution to generalizable emg predictions. *Journal of Neurophysiology*, 109(3): 666–678, 2013.

E. R. Oby, A. D. Degenhart, E. C. Tyler-Kabara, B. M. Yu, and A. Batista. Network constraints dictate the timescale of learning new brain-computer interfaces. In *Annual Meeting of the Society for Neuroscience*, 2015.

J. E. O'Doherty, M. A. Lebedev, P. J. Ifft, K. Z. Zhuang, S. Shokur, H. Bleuler, and M. A. L. Nicolelis. Active tactile exploration using a brain–machine–brain interface. *Nature*, 479 (7372):228, 2011.

B. P. Olson, J. Si, J. Hu, and J. He. Closed-loop cortical control of direction using support vector machines. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 13 (1):72–80, 2005.

M. Omrani, J. Diedrichsen, and S. H. Scott. Rapid feedback corrections during a bimanual postural task. *Journal of Neurophysiology*, 109(1):147–161, 2013.

A. L. Orsborn and B. Pesaran. Parsing learning in networks using brain–machine interfaces. *Current Opinion in Neurobiology*, 46:76–83, 2017.

A. L. Orsborn, H. G. Moorman, S. A. Overduin, M. M. Shanechi, and D. F. Dimitrov J. M. Carmena. Closed-loop decoder adaptation shapes neural plasticity for skillful neuroprosthetic control. *Neuron*, 82(6):1380–1393, 2014.

R. Paz and E. Vaadia. Learning-induced improvement in encoding and decoding of specific movement directions by neurons in the primary motor cortex. *PLoS Biology*, 2(2):e45, 2004.

R. Paz, T. Boraud, C. Natan, H. Bergman, and E. Vaadia. Preparatory activity in motor cortex reflects learning of local visuomotor skills. *Nature Neuroscience*, 6(8):882–890, 2003.

R. Paz, C. Nathan, T. Boraud, H. Bergman, and E. Vaadia. Acquisition and generalization of visuomotor transformations by nonhuman primates. *Experimental Brain Research*, 161(2): 209–219, 2005.

J. A. Pruszynski and S. H. Scott. Optimal feedback control and the long-latency stretch response. *Experimental Brain Research*, 218(3):341–359, 2012.

S. M. Radhakrishnan, S. N. Baker, and A. Jackson. Learning a novel myoelectric-controlled interface task. *Journal of Neurophysiology*, 100(4):2397–2408, 2008.

R. G. Rasmussen, A. B. Schwartz, and S. M. Chase. Dynamic range adaptation in primary motor cortical populations. *Elife*, 6, 2017.

N. D. Ratliff, J. A. Bagnell, and M. Zinkevich. Maximum margin planning. In *Proceeding of the 23rd International Conference on Machine Learning*, pages 729–736, 2006.

G. A. Reina, D. W. Moran, and A. B. Schwartz. On the relationship between joint angular velocity and motor cortical discharge during reaching. *Journal of Neurophysiology*, 85(6): 2576–2589, 2001.

A. G. Richardson, T. Borghi, and E. Bizzi. Activity of the same motor cortex neurons during repeated experience with perturbed movement dynamics. *Journal of Neurophysiology*, 107 (11):3144–3154, 2012.

U. Rokni, A. G. Richardson, E. Bizzi, and H. S. Seung. Motor learning with unstable neural representations. *Neuron*, 54(4):653–666, 2007.

P. T. Sadtler, K. M. Quick, M. D. Golub, S. M. Chase, S. I. Ryu, E. C. Tyler-Kabara, B. M. Yu, and A. P. Batista. Neural constraints on learning. *Nature*, 512:423–426, 2014.

E. Salinas and L. F. Abbott. Vector reconstruction from firing rates. *Journal of Computational Neuroscience*, 1(1–2):89–107, 1994.

J. C. Sanchez, S.-P Kim, D. Erdogmus, Y. Rao, J. C. Principe, J. Wessberg, and M. Nicolelis. Input-output mapping performance of linear and nonlinear models for estimating hand trajectories from cortical neuronal firing patterns. In *Proceedings of the 12th IEEE Workshop on*

Neural Networks for Signal Processing, pages 139–148, 2002.

J. C. Sanchez, D. Erdogmus, Y. Rao, S.-P Kim, M. Nicolelis, J. Wessberg, and J. C. Principe. Interpreting neural activity through linear and nonlinear models for brain machine interfaces. In *Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 2160–2163, 2003.

J. C. Sanchez, J. C. Principe, J. M. Carmena, M. A. Lebedev, and M. Nicolelis. Simultaneus prediction of four kinematic variables for a brain-machine interface using a single recurrent neural network. In *Engineering in Medicine and Biology Society, 2004. IEMBS'04. 26th Annual International Conference of the IEEE*, volume 2, pages 5321–5324, 2004.

M. H. Schieber. Dissociating motor cortex from the motor. *Journal of physiology*, 589(23): 5613–5624, 2011.

A. B. Schwartz. Motor cortical activity during drawing movements: single-unit activity during sinusoid tracing. *Journal of Neurophysiology*, 68(2):528–541, 1992.

A. B. Schwartz. Motor cortical activity during drawing movements: population representation during sinusoid tracing. *Journal of Neurophysiology*, 70(1):28–36, 1993.

A. B. Schwartz. Direct cortical representation of drawing. *Science*, 265(5171):540–542, 1994.

A. B. Schwartz and D. W. Moran. Motor cortical activity during drawing movements: population representation during lemniscate tracing. *Journal of Neurophysiology*, 82(5):2705–2718, 1999.

A. B. Schwartz, R. E. Kettner, and A. P. Georgopoulos. Primate motor cortex and free arm movements to visual targets in three-dimensional space. I. Relations between single cell discharge and direction of movement. *Journal of Neuroscience*, 8(8):2913–2927, 1988.

A. B. Schwartz, D. W. Moran, and G. A. Reina. Differential representation of perception and action in the frontal cortex. *Science*, 303(5656):380–383, 2004.

A. B. Schwartz, X. T. Cui, D. J. Weber, and D. W. Moran. Brain-controlled interfaces: movement restoration with neural prosthetics. *Neuron*, 52:205–220, 2006.

S. H. Scott. The computational and neural basis of voluntary motor control and planning. *Trends in Cognitive Sciences*, 16(11):541–549, 2012.

M. D. Serruya, N. G. Hatsopoulos, L. Paninski, M. R. Fellows, and J. P. Donoghue. Instant neural control of a movement signal. *Nature*, 416(6877):141–142, 2002.

R. Shadmehr and J. W. Krakauer. A computational neuroanatomy for motor control. *Experimental Brain Research*, 185(3):359–381, 2008.

R. Shadmehr, M. A. Smith, and J. W. Krakauer. Error correction, sensory prediction, and adaptation in motor control. *Annual Review of Neuroscience*, 33:89–108, 2010.

R. Shadmehr, H. J. Huang, and A. A. Ahmed. A representation of effort in decision-making and motor control. *Current Biology*, 26(14):1929–1934, 2016.

M. M. Shanechi, Z. M. Williams, G. W. Wornell, R. C. Hu, M. Powers, and E. N. Brown. A real-time brain-machine interface combining motor target and trajectory intent using an optimal feedback control design. *PloS ONE*, 8(4):e59049, 2013.

M. M. Shanechi, A. L. Orsborn, and J. M. Carmena. Robust brain-machine interface design using optimal feedback control modeling and adaptive point process filtering. *PLoS Computational Biology*, 12(4):e1004730, 2016.

K. V. Shenoy and J. M. Carmena. Combining decoder design and neural adaptation in brain-machine interfaces. *Neuron*, 84(4):665–680, 2014.

K. V. Shenoy, M. Sahani, and M. M. Churchland. Cortical control of arm movements: a dynamical systems perspective. *Annual Review of Neuroscience*, 36:337–359, 2013.

L. Shpigelman, H. Lalazar, and E. Vaadia. Kernel-ARMA for hand tracking and Brain-machine interfacing during 3d motor control. In *Advances in Neural Information Processing Systems*, pages 1489–1496, 2009.

R. Sitaram, A. Caria, R. Veit, T. Gaber, G. Rota, A. Kuebler, and N. Birbaumer. Fmri brain-computer interface: a tool for neuroscientific research and treatment. *Computational Intelligence and Neuroscience*, 2007:1–10, 2007.

M. A. Sommer and R. H. Wurtz. A pathway in primate brain for internal monitoring of movements. *Science*, 296(5572):1480–1482, 2002.

A. J. Suminski, D. C. Tkach, A. H. Fagg, and N. G. Hatsopoulos. Incorporating feedback from multiple sensory modalities enhances brain–machine interface control. *Journal of Neuroscience*, 30(50):16777–16787, 2010.

A. J. Suminski, A. H. Fagg, F. R. Willett, M. Bodenhamer, and N. G. Hatsopoulos. Online adaptive decoding of intended movements with a hybrid kinetic and kinematic brain machine interface. In *Proceedings of the 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1583–1586. 2013.

D. M. Taylor, S. I. Tillery, and A. B. Schwartz. Direct cortical control of 3D neuroprosthetic devices. *Science*, 296:1829–1832, 2002.

S. I. H. Tillery and D. M. Taylor. Signal acquisition and analysis for cortical control of neuroprosthetics. *Current Opinion in Neurobiology*, 14(6):758–762, 2004.

C. Tin and C. S. Poon. Internal models in sensorimotor integration: perspectives from adaptive control theory. *Journal of Neural Engineering*, 2(3):147–163, 2005.

E. Todorov. Optimality principles in sensorimotor control. *Nature Neuroscience*, 7(9):907–915, 2004.

E. Todorov. Stochastic optimal control and estimation methods adapted to the noise characteristics of the sensorimotor system. *Neural Computation*, 17(5):1084–1108, 2005.

E. Todorov and M. I. Jordan. Optimal feedback control as a theory of motor coordination. *Nature Neuroscience*, 5(11):1226–1235, 2002.

D. J. Tolhurst, J. A. Movshon, and A.F. Dean. The statistical reliability of signals in single neurons in cat and monkey visual cortex. *Vision Research*, 23(8):775–785, 1983.

J. Trommershäuser, L. T. Maloney, and M. S. Landy. Statistical decision theory and trade-offs in the control of motor response. *Spatial Vision*, 16(3):255–275, 2003.

W. Truccolo, U. T. Eden, M. R. Fellows, J. P. Donoghue, and E. N. Brown. A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. *Journal of Neurophysiology*, 93(2):1074–1089, 2005.

R. J. van Beers, E. Brenner, and J. B. J. Smeets. Random walk of motor planning in task-irrelevant dimensions. *Journal of Neurophysiology*, 109(4):969–977, 2013.

M. Velliste, S. Perel, M. C. Spalding, A. S. Whitford, and A. B. Schwartz. Cortical control of a prosthetic arm for self-feeding. *Nature*, 453:1098–1101, 2008.

J. J. Vidal. Toward direct brain-computer communication. *Annual Review of Biophysics and Bioengineering*, 2(1):157–180, 1973.

J. J. Vidal. Real-time detection of brain events in eeg. *Proceedings of the IEEE*, 65(5):633–641, 1977.

R. Wahnoun, J. He, and S. I. H. Tillery. Selection and parameterization of cortical neurons for neuroprosthetic control. *Journal of Neural Engineering*, 3(2):162, 2006.

W. Wang, S. S. Chan, D. A. Heldman, and D. W. Moran. Motor cortical representation of position and velocity during reaching. *Journal of Neurophysiology*, 97(6):4258–4270, 2007.

J. Wessberg, C. R. Stambaugh, J. D. Kralik, P. D. Beck, M. Laubach, J. K. Chapin, J. Kim, S. J. Biggs, M. A. Srinivasan, and M. A. L. Nicolelis. Real-time prediction of hand trajectory by ensembles of cortical neurons in primates. *Nature*, 408(6810):361–365, 2000.

M. C. Wetzel. Operant conditioning in motor and neural integration. *Neuroscience and Biobehavioral Reviews*, 10(4):387–429, 1986.

F. R. Willett, A. J. Suminski, A. H. Fagg, and N. G. Hatsopoulos. Improving brain-machine interface performance by decoding intended future movements. *Journal of Neural Engineering*, 10(2):026011, 2013.

S. P. Wise, S. L. Moody, K. J. Blomstrom, and A. R. Mitz. Changes in motor cortical activity during visuomotor adaptation. *Experimental Brain Research*, 121(3):285–299, 1998.

B Wodlinger, JE Downey, EC Tyler-Kabara, AB Schwartz, ML Boninger, and JL Collinger. Ten-dimensional anthropomorphic arm control in a human brain-machine interface: difficulties, solutions, and limitations. *Journal of Neural Engineering*, 12(1):016011, 2014.

J. R. Wolpaw and D. J. McFarland. Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans. *Proceedings of the National Academy of Sciences of the United States of America*, 101(51):17849–17854, 2004.

J. R. Wolpaw, D. J. McFarland, G. W. Neat, and C. A. Forneris. An eeg-based brain-computer interface for cursor control. *Electroencephalography and Clinical Neurophysiology*, 78(3): 252–259, 1991.

D. M. Wolpert, Z. Ghahramani, and M. I. Jordan. An internal model for sensorimotor integration. *Science*, 269(5232):1880–1882, 1995.

H. G. Wu and M. A. Smith. The generalization of visuomotor learning to untrained movements and movement sequences based on movement vector and goal location remapping. *Journal of Neuroscience*, 33(26):10772–10789, 2013.

W. Wu and N. G. Hatsopoulos. Real-time decoding of nonstationary neural activity in motor cortex. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 16(3):213–222, 2008.

W. Wu, M. J. Black, Y. Gao, E. Bienenstock, M. Serruya, A. Shaikhouni, and J. P. Donoghue. Neural decoding of cursor motion using a Kalman filter. In *Advances in Neural Information Processing Systems*, pages 133–140. 2003.

W. Wu, Y. Gao, , E. Bienenstock, J. P. Donoghue, and M. J. Black. Bayesian population decoding of motor cortical activity using a Kalman filter. *Neural Computation*, 18(1):80–118, 2006.

S.-S. Yoo, T. Fairneny, N.-K. Chen, S.-E. Choo, L. P. Panych, H. Park, S.-Y. Lee, and F. A. Jolesz. Brain–computer interface using fmri: spatial navigation by thoughts. *Neuroreport*, 15 (10):1591–1595, 2004.

B. M. Yu, J. P. Cunningham, G. Santhanam, S. I. Ryu, K. V. Shenoy, and M. Sahani. Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. *Journal of Neurophysiology*, 102(1):614–635, 2009.

N. Zach, D. Inbar, Y. Grinvald, H. Bergman, and E. Vaadia. Emergence of novel representations in primary motor cortex and premotor neurons during associative learning. *Journal of Neuroscience*, 28(38):9545–9556, 2008.

Y. Zhang and S. M. Chase. A stabilized dual Kalman filter for adaptive tracking of brain-computer interface decoding parameters. In *Proceedings of the 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 7100–7103, 2013.

Y. Zhang and S. M. Chase. Recasting brain-machine interface design from a physical control system perspective. *Journal of Computational Neuroscience*, 39(2):107–118, 2015.

Y. Zhang and S. M. Chase. A control-theoretic approach to brain-computer interface design. In *American Control Conference*, pages 5765–5771, 2016.

Y. Zhang and S. M. Chase. Optimizing the usability of brain computer interfaces. *Neural Computation*, 30(5):1323–1358, 2018.

Y. Zhang, A. B. Schwartz, S. M. Chase, and R. E. Kass. Bayesian learning in assisted brain-computer interface tasks. In *Proceedings of the 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 2740–2743, 2012.

X. Zhou, R. Tien, and S. Chase. Distinct timescales of cortical reorganization in a long-term learning task. In *Annual Meeting of the Society for Neuroscience*, 2015.

B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Proceeding of the 23rd AAAI Conference on Artificial Intelligence*, pages 1433–1438, 2008.