Design of control algorithms for redundant neuroprosthetic brain-machine interfaces

Suraj Gowda

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2015-182
http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-182.html

August 10, 2015

Design of control algorithms for redundant neuroprosthetic brain-machine interfaces

by

Suraj Raju Gowda

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Jose Carmena, Chair
Professor Claire Tomlin
Professor Masayoshi Tomizuka

Summer 2015

Design of control algorithms for redundant neuroprosthetic brain-machine interfaces

Abstract

Design of control algorithms for redundant neuroprosthetic brain-machine interfaces

by

Suraj Gowda

Doctor of Philosophy in Engineering–Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Jose M. Carmena, Chair

Dexterous control of upper arm movements is integral to activities of daily living. This important ability may be lost due to neurological injury or disease, yet brain-machine interfaces (BMIs) or neuroprosthetics may offer the ability to restore motor function by bypassing damaged neural circuitry. Though tremendous progress has been made in BMI performance and robustness, additional gains are needed to improve the clinical utility of BMI systems. Of particular importance is the need to scale BMI performance beyond the simple 2-D reaching paradigm typically studied and extend to the multi-degree-of-freedom (multi-DOF) case. We explored methods and design considerations for redundant multi-DOF BMIs and examine the neuroprosthetic control mechanisms engaged by subjects.

In this work, healthy non-human primate subjects controlled a variety of virtual and physical actuators using neural signals recorded from electrode arrays implanted in the motor cortex. We analyzed 2-D natural arm movements and developed a fast method to extract sparse components of hand kinematics during point-to-point reaching. We then analyzed neuroprosthetic control of a computer cursor in which neural decoding is performed using a Kalman filter (KF). We discovered several links between KF parameters and control performance of the overall BMI system, including the ability to manipulate the speed-accuracy tradeoff of neural cursors. Building from hypotheses of model based control in natural motor control, we examine BMI cursor control for evidence of model-based and feedforward control.

We then proceeded to develop an architecture for BMI control of a redundant actuator. Subjects learned BMI control of a redundant four-link virtual "arm" which was confined to move in a two-dimensional plane. When controlling this redundant arm, performance improved when subjects made use of the redundant control dimensions. Furthermore, redundant control signals were critical to escaping manipulator singularities, similar to how redundant control signals are used in robotic systems. These results suggest the utility of using neural signals in conjunction with standard robotic path planning methods to control endpoint-redundant aspects of a robotic prosthesis. Altogether, the work of this dissertation contributes progress toward BMI control of highly redundant prosthetic systems for clinical use.

Acknowledgements

When I started graduate school, my goal was to find a project that was important and impactful but which was also, to put it simply, awesome. I wanted to work on something I thought was important, but also something I simply would not have been able to do except in an academic environment. Neuroprosthetics fit the bill perfectly, and I can't thank Jose Carmena enough for the chance to work on such amazing and important areas of research and for advising me throughout my PhD. I'm lucky to have worked with someone who was so committed giving his students the freedom to choose what to work on. Jose's enthusiasm never let me forget the big picture, especially when progress was difficult to come by.

A substantial part of the work in this thesis involved experimental work with primates. When people ask me what primate work is like, I explain that it's like trying to work on science with an incredibly strong toddler. As such, I would have failed in the experimental work if not for the help of my labmates Amy Orsborn, Simon Overduin, Alejandra Dominguez and Helene Moorman. In particular I'd like to thank Simon for teaching me how to handle the primates and Helene for being such a fantastic collaborator on a huge fraction of this work.

In retrospect, it's not particularly easy for me to differentiate between my undergraduate time at Berkeley and my graduate time. In truth, it feels more like one long Berkeley journey, and so I would be remiss if I did not also thank those who set me up for success. I deeply appreciate the chance that Dan Werthimer gave me to get into research early on in my undergraduate years. The experience of working with Dan solidified my interests in research and graduate school in general and signal processing applications in particular. Similarly, I had numerous amazing professors without whom I never would have gotten excited about signal processing.

I would also like to thank Michel Maharbiz, Claire Tomlin and Masayoshi Tomizuka for serving on my quals/thesis committees and for their invaluable feedback on the details of my work. The Tomlin and Tomizuka labs have also been great collaborators as we ambitiously attempted to build an upper arm exoskeleton controlled by brain signals. I hope this thesis serves as a small but substantial foundation stone as that project continues.

My years at Berkeley would have been much less enjoyable without the company and support of many coworkers/labmates, friends and roommates. Thank you all for helping me take my mind off work when needed (and sometimes, when not needed). The Carmena lab has simply been a fantastic group to be a part of. Amy, Helene, Simon, Sid, Preeya and Sam: thanks for struggling through the primate work alongside me. Chris, Ryan, Yu-Xuan, Jared, Sid, Shervin, Julian and Kevin: thanks for being such awesome roommates over the years–life in Berkeley wouldn't have been the same otherwise. Thanks Arjun, Sid, Julian, Kevin, Dan, Cameron, as well as other random strangers, for basketball-based diversions from academic work. Samarth, Anjali, Brian, Mike, Sarah, Ryan, Catherine, Sid, Dan, Cameron, Kevin: thanks for being an awesome group of people to hang out with outside of work; I look forward to additional extracurricular activities in the years to come. Sid, as my roommate and labmate for years and friend since the early days, no part of my (long) time

at Berkeley would have been the same without you. I feel that I'm omitting so much in this paragraph, but I can only hope that those of you who supported my research dreams knew my appreciation long before reading this page.

There were many times when completing my PhD felt impossible. Luckily for me, in those times I needed to look no further than my parents and grandparents for fantastic examples of what hard work can accomplish. In comparison to their work, the effort I put into dissertation is negligible. Jokingly, my parents claim that my accomplishments are actually their accomplishments since they did all the setup work. It's hard to fault their logic.

Contents

Chapter 1

Introduction

Neuroprosthetics enable cortical control of an external actuator using a direct brain-machine interface (BMI). This remarkable technology may hold the potential to restore communication and motor action for patients with severe neurological disorders or injuries, including stroke, amyotrophic lateral sclerosis (ALS) and spinal cord injury. A number of compelling proof-of-concept demonstrations have also been shown in both monkeys [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11] and humans [12, 13, 14]. Cortically controlled prosthetics have become a realistic possibility as advances in chronic, multi-site, multi-electrode arrays have demonstrated that neural signals can be recorded and decoded for years from both monkeys and humans.

In addition to their potential clinical value, BMIs also provide a unique opportunity to explore learning and control of novel systems. Despite the abstract nature of BMI control in a virtual environment, the acquisition of BMI skill has been shown to resemble the acquisition of natural motor skill. Over the course of several days, retuning of the BMI network during BMI control has stability resembling a memory engram [8], can rapidly context-switch [8], isolate the output layers of the network [15, 16], and involves deeper brain structures in the basal ganglia [17]. Sleep has also been shown to play an important role in the consolidation of BMI skill [18], as it does during learning of natural motor skill. Studies of short-term learning (single session) have also shown that adaptation to perturbation involves both global (entire network) and cell-specific adaptation strategies [19, 20], and that learning is faster when it does not require changing the spontaneous correlational structure of the network [21].

Despite the compelling proof-of-concept of many BMI demonstrations, much remains to be desired in terms of the robustness, flexibility, and ease-of-use of current BMI systems. Improvement is needed both on the physical neural interface components as well as the algorithms used to infer control signals for the actual prosthetic devices. On the implant side, a wireless, fully implantable, long-lasting, more biocompatible system remains to be developed, though much progress is being made on the neurotechnology front. This work instead focuses on the development of control algorithms for BMI systems.

A central part of any BMI system is the *decoder* which maps from neural control signals to actuator control signals (e.g., velocity set points, torque commands, etc.). Feedback interaction between the brain and the decoder is a crucial determinant of performance.

Intuitively, errors in the decoding algorithm resulting in BMI trajectories inconsistent with the subject's intention will result in a reaction from the subject to modify their motor plan. This simple intuition suggests that we ought tune the parameters of our decoder while the subject is interacting with the decoder. This approach, which we term closed-loop decoder adaptation (CLDA), is a critical component of the methods used to develop the BMI systems in this work.

In typical control problems, we may have a system which is not fully observable or fully modeled, but full freedom to design the control law. In BMI applications, the inverse is true: the control law is specified by the subject and we have little design freedom over it, but we do have the ability to alter the properties of the system, the "plant", to try and enhance control. Although the decoder has been identified as important component of the closed-loop interaction between brain and machine, relatively little is known about what type of decoder is optimal for a neural controller. What coordinate system should the decoder operate in? What should be the rate at which the decoder operates? Should an individual BMI cell contribute to all aspects of the control signal, or should the cell population be more compartmentalized, e.g., separate populations of cells to control the shoulder and to control the elbow? These questions and others remain to be answered. Importantly, they remain somewhat mysterious in natural movement as well; for example, the firing patterns of motor cortical neurons has been implicated in both kinetic [22] as well as kinematic control [23].

In this work, we seek to analyze the neural control of movement, both natural and neuroprosthetic, to inform how future generations of BMIs should be designed. Chapter 2 provides an overview of the experimental and mathematical methods used in this work. Chapter 3 demonstrates for the particular case of the Kalman filter how a control systems analysis can reveal properties which are not easily evident from the statistical modeling framework. We discover pathological behavior of the decoding algorithm which emerges due to calibration error and devise parameter estimators to correct this error, and discover simple methods to tune the speed-accuracy tradeoff during BMI control. Chapter 4 explores the hypothesis that natural motor control is intermittent and uses sparse feedforward control inputs in addition to a continuous feedback controller. We devise accelerated methods to extract this sparsity from natural movements. In Chapter 5, we examine the possibility of model-based control of BMI systems. Finally, in Chapter 6, we demonstrate experimentally BMI control of a redundant virtual actuator and show that subjects are able to exercise control over the redundant elements in a way that enhances their control over the system.

Chapter 2

Experimental and mathematical methods

This chapter outlines the experimental as well as mathematical methods used in this work.

2.1 Electrophysiology

Six adult male rhesus macaques (*Macaca mulatta*; monkeys P, R, S, J, C and G) were used in this work. Not all monkeys participated in all aspects of each experiment. All procedures were conducted in compliance with the National Institute of Health Guide for Care and Use of Laboratory Animals and were approved by the University of California, Berkeley Institutional Animal Care and Use Committee.

The monkeys were all implanted with chronic microwire electrode arrays for neural recording (35 micron diameter, 500 micron wire spacing; Innovative Neurophysiology, Durham, NC). Arrays targeted the arm areas of primary motor cortex (M1) and dorsal premotor cortex (PMd) based on stereotactic coordinates. Monkeys S and J were implanted with one 8x16 array in each hemisphere targeting M1 and PMd. Monkey C also received the same arrays as Monkeys S and J, but was also implanted with two 64 channel arrays targeting the arm area of ventral premotor cortex (PMv), one in each hemisphere. Monkey G was implanted in the left hemisphere with an 8x16 array and in the right hemisphere with two 8x8 arrays.

For monkeys S and J, single and multi-unit activity was recorded using a 128-channel MAP system and sorted online using Sort Client (Plexon, Inc., Dallas, TX). For monkeys C and G, a 256-channel Omniplex system (Plexon, Inc., Dallas, TX) was used instead. The BMI control in this work focused primarily on control using putative single- and multi-unit action potentials. In some experiments, typically when recording quality had degraded, the neural signal was local field potentials (LFPs).

2.2 Behavioral Tasks

All the behavioral tasks used in this work are variants of a 2D point-to-point reach and hold task in which they were required to move a cursor to a highlighted target. Monkeys were first trained on the abstract computer task by associating the cursor with the motion of their hand, either by moving their arm using a 2-D planar exoskeleton (BKIN Technologies, Kingston, ON, Canada) or by moving a manipulandum (joystick). The monkeys initiated trials by moving the cursor to the visually-instructed origin and holding for 400-1500ms. Upon entering the origin, the terminus appeared. The monkeys were then required to reach to the specified terminus within a specified time-limit and hold for 400-1500ms to receive a liquid reward. More specific details are provided in each chapter, as the task details often varied by the scientific questions being asked. In some experiments, after they learned the rules of the task they were switched to controlling the cursor using cortical neural activity and did not directly benefit from overt arm movement.

2.3 Decoding algorithms

The decoding algorithm or "decoder" of a BMI forms the bridge from neural activity (e.g., spike rate, field potential power, etc.) to commands to an actuator (e.g., virtual cursor, virtual arm, exoskeleton, etc.). This section provides a partial overview of the methods used in this work and is primarily background material. Novel methods are introduced in context in later chapters.

The core of current state-of-the-art decoding algorithms involves inference on a hidden Markov model graph, as shown in Figure 2.1. The hidden state $\tilde{x}_t$ represents the subject's intended BMI state at time t, and the observations y_t represent neural activity features observable from the electrode array. Inference algorithms generate an estimate of the intended state, x_t, which is then used to drive the actuator. For example, $\tilde{x}_t$ could be used to represent the subject's intended cursor position ($\tilde{p}$) and velocity ($\tilde{v}$) at time t:

$$\tilde{x}_t = \begin{bmatrix} \tilde{p}_t^{horiz} & \tilde{p}_t^{vert} & \tilde{v}_t^{horiz} & \tilde{v}_t^{vert} & 1 \end{bmatrix}^T.$$

Symbols and notation

$$\begin{aligned} x_t &= \text{hidden state, e.g. intended BMI kinematics} \\ \Delta &= \text{discrete-time update rate of the filter} \\ y_t &= \text{Spikes observed during interval } (t, t+\Delta] \text{ for all } N\text{neurons} \\ y_{[k,t]} &= \text{Spikes observed during interval } (t, t+\Delta] \text{ for neuron } k \\ y_{t_0}^{t_1} &= \{y_{t_0}, \cdots, y_{t_1}\} \\ y^t = y_0^t &= \text{History of spike observations up to time } t \end{aligned}$$

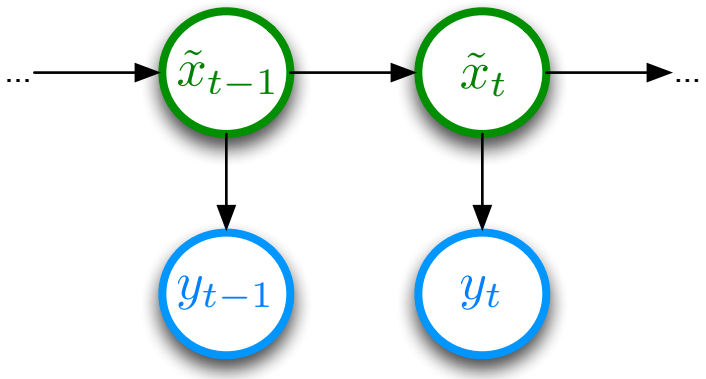

Figure 2.1: Hidden Markov model for BMI decoding.

We make use of two special types of matrices, the $M \times P$ matrix of all zeros $0_{M\times P}$ and the $N \times N$ identity matrix I_N. When possible, we follow the standard mathematical convention of writing 0 in place of $0_{M\times P}$ when the size of the matrix is unambiguous. When indexing sub-matrices, we follow the MATLAB convention, e.g. $z_{[1:2,:]}$ refers to the first two rows of a matrix z.

State-space model

The **state-space model** (SSM) is our model of how the intended trajectory would evolve over time if we could observe it directly. In mathematical terms, the state-space model defines the conditional probability distribution $p(\tilde{x}_t|\tilde{x}_{t-1})$. In nearly all applications where the BMI state is continuous-valued and the subject exerts continuous control over the prosthesis, a **Gaussian random walk** SSM is used:

$$\tilde{x}_{t+1} = A\tilde{x}_t + w_t; \quad w_t \sim \mathcal{N}(0, W).$$

This model is typically selected not because it is particularly accurate, but rather because it simplifies inference on the model. In this model, w_t represents our estimate of the control input to the system. Indeed, comparison to a standard linear system

$$\tilde{x}_{t+1} = A\tilde{x}_t + Bu_t,$$

where Bu_t represents the subject's input control signal to the system, reveals that the process noise w_t is just a rather simple zeroth-order model of what would be the control term in a linear system. We will return to this relationship in later chapters.

Kalman filter and Gaussian observation model

Variants of the KF have been used in several online BMI experiments [11, 24, 25, 26, 27]. Under the KF model, the observations y_t represent the spikes observed from a neural ensemble

in the past 100 ms. The KF models $\tilde{x}_t$ and y_t as jointly Gaussian with the relationship

$$y_t = C\tilde{x}_t + q_t; \quad q_t \sim \mathcal{N}(0, Q).$$

This model of neural firing assumes that spikes are linearly related to cursor kinematics. Neural noise is modeled by q_t, which represents neural firing covariance not captured by the linear model.

Note that due to conditional independence properties (Markov properties) of the KF model, $p(y_t|x_t, y^{t-1}) = p(y_t|x_t)$. In words, the current spike observations are conditionally independent of past spike observations when the BMI state is known. Parameters A and W define the "state space" model while parameters C and Q constitute the neural firing model. Together, the parameter set $\theta = A, W, C, Q$ completely specifies the KF model.

The minimum mean squared error estimate of $\tilde{x}_t$, x_t, is both linear and recursive under the KF model:

$$x_t = (I - K_t C)Ax_{t-1} + K_t y_t. \tag{2.3.1}$$

The KF algorithm recursively estimates $\tilde{x}_t$ using the previous estimate x_{t-1} and the most recent observation y_t. The Kalman gain K_t is an optimal linear mapping that combines the observation vector with the previous state estimate. The decoder was updated as frequently as new observations of neural activity could be made (every Δ=100 ms).

The Kalman gain can be computed using common properties of jointly-gaussian distributions. The Kalman gain can be found using basic properties of jointly Gaussian distributions. We begin by partitioning the set y^t:

$$p\left(x_t \middle| y^t\right) = p\left(x_t \middle| y_t, y^{t-1}\right).$$

The latter expression has a simple form because the variables are jointly Gaussian. To derive the distribution $p(x_t|y^t)$, we first write the joint distribution $p(x_t, y_t|y^{t-1})$:

$$\left[\begin{array}{c} x_t \\ y_t \end{array}\right] \middle| y^{t-1} \sim \mathcal{N}\left(\left[\begin{array}{c} \hat{x}_{t|t-1} \\ C\hat{x}_{t|t-1} \end{array}\right], \left[\begin{array}{cc} cov\left(x_t \middle| y^{t-1}\right) & cov\left(x_t, y_t \middle| y^{t-1}\right) \\ cov\left(y_t, x_t \middle| y^{t-1}\right) & cov\left(y_t \middle| y^{t-1}\right) \end{array}\right]\right).$$

We fill in blocks of the joint covariance matrix with the covariance calculations:

$$\begin{aligned}
cov\left(x_t \middle| y^{t-1}\right) &\triangleq P_{t|t-1} \\
cov\left(x_t, y_t \middle| y^{t-1}\right) &= E\left[x_t y_t^T \middle| y^{t-1}\right] = E\left[x_t (Cx_t)^T + x_t q_t^T \middle| y^{t-1}\right] \\
&= E\left[x_t x_t^T \middle| y^{t-1}\right] C^T = P_{t|t-1} C^T \\
cov\left(y_t, x_t \middle| y^{t-1}\right) &= cov\left(x_t, y_t \middle| y^{t-1}\right)^T = CP_{t|t-1}^T = CP_{t|t-1} \\
cov\left(y_t \middle| y^{t-1}\right) &= cov\left(Cx_t \middle| y^{t-1}\right) + cov\left(q_t \middle| y^{t-1}\right) \\
&= C \cdot cov\left(x_t \middle| y^{t-1}\right) C^T + cov(q_t) = CP_{t|t-1}C^T + Q.
\end{aligned}$$

The distribution becomes

$$\left[\begin{array}{c} x_t \\ y_t \end{array}\right] \middle| y^{t-1} \sim \mathcal{N}\left(\left[\begin{array}{c} \hat{x}_{t|t-1} \\ C\hat{x}_{t|t-1} \end{array}\right], \left[\begin{array}{cc} P_{t|t-1} & P_{t|t-1}C^T \\ CP_{t|t-1} & CP_{t|t-1}C^T + Q \end{array}\right]\right)$$

Multivariate Gaussian theory provides a simple way to write the conditional distribution $p\left(x_t|\, y^t\right)$ from the joint distribution $p(x_t, y^t)$:

$$\begin{aligned}
x_t|\, y^t &\sim \mathcal{N}\left(\hat{x}_{t|t}, P_{t|t}\right) \\
\hat{x}_{t|t} &= \hat{x}_{t|t-1} + cov\left(x_t, y_t|\, y^{t-1}\right) cov\left(y_t|\, y^{t-1}\right)^{-1}\left(y_t - C\hat{x}_{t|t-1}\right) \\
&= \left(I - K_t C\right)\hat{x}_{t|t-1} + K_t y_t;\ K_t = P_{t|t-1}C^T\left(CP_{t|t-1}C^T + Q\right)^{-1} \\
P_{t|t} &= cov\left(x_t, y_t|\, y^{t-1}\right) cov\left(y_t|\, y^{t-1}\right)^{-1} cov\left(y_t, x_t|\, y^{t-1}\right) \\
&= P_{t|t-1}C^T\left(CP_{t|t-1}C^T + Q\right)^{-1} CP_{t|t-1},
\end{aligned}$$

where K_t as defined above is the Kalman gain. Finally, to complete the recursion, we determine the distribution $p\left(x_t|y^{t-1}\right)$:

$$\begin{aligned}
x_t|\, y^{t-1} &\sim \mathcal{N}\left(\hat{x}_{t|t-1}, P_{t|t-1}\right) \\
\hat{x}_{t|t-1} &= E\left[x_t|\, y^{t-1}\right] = E\left[Ax_{t-1} + w_t|\, y^{t-1}\right] \\
&= AE\left[x_t|\, y^{t-1}\right] \\
&= A\hat{x}_{t|t-1} \\
P_{t|t-1} &= cov\left(x_t|\, y^{t-1}\right) = cov\left(Ax_{t-1} + w_t|\, y^{t-1}\right) \\
&= cov\left(Ax_{t-1}|\, y^{t-1}\right) + cov\left(w_t|\, y^{t-1}\right) \\
&= A \cdot cov\left(x_{t-1}|\, y^{t-1}\right) A^T + W \\
&= AP_{t-1|t-1}A^T + W.
\end{aligned}$$

The quantities $P_{t|t-1}$ and K_t are closely related to each other and will be important for any modifications to the KF parameter estimation procedure.

Point-process observation model

Though the Kalman filter has been successfully applied to the decoding of action potentials by approximating integer spike counts as Gaussian, spike processes are in reality better described as *point-processes*, where events occur with some rate λ, where λ may in general depend on some (unobservable) covariates or the past event history. For sufficiently small Δ, the point process observation model for unit k is

$$\begin{aligned}
p\left(y_{[k,t]}|x_t, y^{t-1}\right) &\approx \exp\left\{y_{[k,t]} \log\left[\lambda_k\left(t|x_t, y^t\right)\Delta\right] - \lambda_k\left(t|x_t, y^t\right)\Delta\right\} \\
&= \left[\lambda_k\left(t|x_t, y^t\right)\Delta\right]^{y_{[k,t]}} \exp\left\{\lambda_k\left(t|x_t, y^t\right)\Delta\right\}
\end{aligned}$$

For derivation of the point-process filter (PPF), it's sufficient to think about the observation model as modeling the probability of observing 1 spike in a time period of length Δ under a Poisson distribution. Each unit is assumed to be conditionally independent when the state

x_t and process history y^{t-1} are known, so

$$p\left(y_t|x_t, y^{t-1}\right) = \prod_{k=1}^{N} p\left(y_{[k,t]}|x_t, y^{t-1}\right)$$

Generalized linear models are often used to capture the relationship between relevant covariants (kinematic parameters, spiking history, etc.) and the spike rates λ_k (e.g., [28]). In the general case (for any model of λ), the inference equation becomes

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + P_{t|t}^{-1}\left(\frac{\partial}{\partial x_t}\log\lambda_t\right)\left[y_t - \Delta\lambda_t\right].$$

A full derivation of the relationship above is provided in A.2, as well as the equation for calculating the posterior estimator covariance, $P_{t|t}$.

The simplest BMI-relevant model of neural tuning is

$$\log\lambda_k\left(t|\, x_t\right) = C_{[k,:]}x_t$$

This simple model has conditionally independent cells, and does not take into account the self or population history of neural spiking. In this case, evaluating $P_{t|t}$ becomes quite simple and the PPF equations greatly resemble the KF equations:

$$Q^{-1} := \begin{bmatrix} \Delta\exp\left(C_{[0,:]}x_t\right) & & \\ & \ddots & \\ & & \Delta\exp\left(C_{[N,:]}x_t\right) \end{bmatrix}$$

$$P_{t|t} = P_{t|t-1} - P_{t|t-1}C^T\left(Q + CP_{t|t-1}C^T\right)^{-1}CP_{t|t-1}$$

Note that (A.2.13) and (A.2.6) are identical with the appropriate definition of Q^{-1}. For completeness, we show the use of the matrix inversion lemma, but (A.2.14) is not necessarily numerically stable since it requires calculating the inverse of a matrix with many small diagonal elements, since typically the number of expected spikes in a bin will be predicted to be close to 0. In the KF, the Q matrix is a static parameter meant to represent "unexplained covariance" or "noise" in the observations. In the PPF, the Q matrix equivalent changes dynamically with the observation. In essence, the more spikes you observe, the more your uncertainty about the state decreases

2.4 Closed-loop decoder adaptation (CLDA)

Closed-loop decoder adaptation (CLDA) is an emerging paradigm for rapidly improving closed-loop BMI performance by re-estimating decoder parameters while the subject continues to operate the BMI [2, 29, 30, 31, 11, 26].

SmoothBatch

Parameter estimation procedures were described in our previous work [26]. Briefly, A and W represent a linear model of hand endpoint kinematics over time, which can be estimated independently of C and Q due to the structure of the KF model. Linear regression was used to estimate A and W from arm movement kinematics discretized at the update rate of the BMI. Here we used SmoothBatch [26], a CLDA algorithm that updates decoder parameters on an intermediate time-scale with a sliding average of parameters estimated on small (1-2 min) batches of data. Parameter settings were updated by the equations

$$C^{i+1} = \alpha C^i + (1-\alpha)\,\hat{C} \tag{2.4.1}$$

$$Q^{i+1} = \alpha Q^i + (1-\alpha)\,\hat{Q} \tag{2.4.2}$$

In (2.4.1), $\hat{C}$ is an estimate of C from the newest "batch" of data (e.g., the previous minute), C^i and C^{i+1} are the previous and updated parameter settings, respectively, and α determines the rate at which parameters change. Maximum-likelihood estimators (MLEs) are used to calculate $\hat{C}$ and $\hat{Q}$ by constructing a linear model between our estimate of the subject's intended kinematics and neural spiking [26].

2.5 Assistive (shared) control

During the CLDA calibration procedure described in Section 2.4, the parameter initialization may result in very poor performance in the early stages of CLDA. To keep the subjects engaged and to ensure that the training data set spans the entire workspace roughly evenly, we may share control between a machine feedback controller and the decoder in the initial calibration procedure. The assistive input methods we use are typically in two flavors

Directed random walk assist

Section 2.3 introduced the random walk model which we typically used for decoding algorithms. This section introduces a slight variant, the directed random walk model:

$$x_{t+1} = Ax_t + c_t + w_t; \quad w_t \sim \mathcal{N}(0, W) \tag{2.5.1}$$

The new term c_t is what specifies the "directed" component of the model, as it alters the state in a way which is not zero-mean. This captures a "control input term" to a linear system, which is absent from the random walk model. In essence, an appropriate choice of c_t inputs makes this a noisy, goal-directed model. In our applications, this term may be specified in three ways:

1. Specify c_t directly.

2. Specify u_t and compute $c_t = Bu_t$ for a static control input matrix B.

3. Specify F and x^* and compute $u_t = -F(x_t - x^*)$ for a static control input matrix B.

In the cases where c_t is specified without reference to the current state x_t (cases 1 and 2 above), the model control input is regarded as an "oracle" signal that appears magically and does not affect the covariance of x_{t+1}. In the third case, the random walk model becomes

$$\begin{aligned} x_{t+1} &= Ax_t - BF(x_t - x^*) + w_t \\ &= (A - BF)\, x_t + BFx^* + w_t. \end{aligned}$$

In this case, the covariance of x_{t+1} is affected. To fit this case into our original formalism (2.5.1), we make the substitutions

$$\begin{aligned} A &\leftarrow A - BF \\ c_t &\leftarrow BFx^*, \end{aligned}$$

where in the modified equivalent system, c_t becomes a constant if x^* is constant.

Weighted average assist

One drawback with using the directed random walk assist is that the control term is simply added onto the decoder. Thus the decoder may be able to "overpower" the assist and in cases where the parameter initialization is extremely bad, this may result in bizarre performance. This ability of the decoder to overpower the assist comes from the w_t term, where the possibility of process noise admits neural control inputs into the system. This overpowering can be squashed by changing the magnitude of the W matrix temporarily. However, the units of the W matrix may be difficult to interpret and hence difficult to manipulate in a way that carefully impacts performance (see Section 3.3 for a discussion on this point using real experimental data). An equivalent but much simpler parameterization simply involves a weighted assist:

$$x_{t+1} = (1 - \alpha)x_{t+1}^{decoder} + \alpha x_{t+1}^{assist}$$

Bounding of configuration space

In addition to generating goal-directed trajectories during the parameter calibration period, similar methods may be used to apply soft constraints to the state of the prosthesis, e.g., to keep the position of a cursor in range of an "equilibrium" point or to keep a robot near a particular part of configuration space. In this case, applying the methods of Section 2.5 with x^* set to a constant, experimenter-specified equilibrium state yields a system which softly constrains the state of the BMI. Just as with the assist, the decoded neural commands can

override the model-generated command to drive the system toward the equilibrium state. We apply this idea to the control of a redundant kinematic chain in Chapter 6 and find in Chapter 3 that similar behavior emerges by accident when certain neural models are applied to KF control of a cursor.

Chapter 3

Designing dynamical properties of BMIs to optimize task-specific performance

This chapter analyzes a BMI cursor system from a linear systems perspective and analyzes empirically and theoretically how changes in the decoder parameters affect task performance. This work was in collaboration with Amy Orsborn, Simon Overduin, Helene Moorman and Jose Carmena and has been published [32].

3.1 Introduction

An understanding of the dynamical properties of closed-loop BMI systems may inform the design of future BMI decoding algorithms. In this paper, we demonstrate that closed-loop dynamical properties of a BMI vary due to parameter recalibration, affect BMI performance and can be tailored to a particular task to maximize performance.

In closed-loop BMI, subjects attempt to control the state of the neuroprosthesis using real-time feedback. Discrepancies between the subject's motor intent and the output of the decoding algorithm, or "decoder", likely alter the subject's future control plan. Given this inherent closed-loop interaction between subject and decoder, the dynamical properties of the decoder are of great relevance to BMI performance. For instance, performance is influenced by the selection of kinematic variables to be decoded. BMIs using a stochastic position model were outperformed by BMIs where position was computed deterministically by integrating velocity [3, 11, 24]. In addition, closed-loop BMI experiments have demonstrated how the smoothness of the decoded output [**?**] as well as the total delay of the decoder [33] can affect performance. BMI subjects can also improve performance with learning [8], adapt to rotation perturbations [19, 34, 20], and even learn to control arbitrary linear systems [7, 8]. However, experimental evidence also suggests that BMI subjects may not be able to learn to produce arbitrary activity patterns with the neurons selected for BMI [20, 35], meaning that there may be decoders with properties that cannot be optimally controlled. Understanding the properties and performance impacts of various BMI subsystems can enable the design of

more effective BMIs.

To better understand the range of possible closed-loop BMI system properties and how they influence performance, we trained monkeys to make center-out reaches using a Kalman filter (KF) BMI that decoded cursor position and velocity [25]. The position-velocity KF (PVKF) has readily interpretable parameters that can produce a broad range of interesting dynamical features. We analyzed two system properties in detail. First, nearly all parameter settings for the PVKF create attractor points that the subject's control input to the system must counteract. Attractor points can change unpredictably when the PVKF parameters change, e.g. when the decoder is recalibrated, but they are eliminated entirely in systems where only velocity is decoded. Second, we observed an intuitive tradeoff between speed of the cursor and proficiency in holding at the target. This tradeoff is highly correlated with changes in system memory, which can also change with recalibration but can be easily controlled by the BMI designer with proper parameter selection.

3.2 Methods

Task

Monkey S (J) was trained to perform a self-paced delayed 2D center-out reaching task to 8 circular targets of 1.7 (1.2) cm radius uniformly spaced about a 14 (13) cm-diameter circle. After being trained to perform the task with arm movements, the monkeys learned to perform the BMI task. In this task, the monkeys controlled the cursor directly with neural activity via a Kalman filter decoder and in the absence of overt arm movements. Figure 3.1 illustrates the experimental setup. The monkeys initiated trials by moving the cursor to the center and holding for 400 (250) ms. The monkeys had an unlimited amount of time to enter the center to initiate a trial. Upon entering the center, the peripheral target appeared. After the center-hold period ended, the monkeys were cued (the center changed color) to initiate the reach. Then they were required to move the cursor to the peripheral target within a given 3-7 second time-limit and hold for 400 (250) ms to receive a liquid reward. Failure to hold at the center, hold at the peripheral target, or reach the peripheral target within the time limit restarted the trial to the same target without reward. Center hold errors at the start of the trial occurred frequently as the penalty to the monkey was minimal. Failure to hold at the target was far more common than failure to reach the target within the time limit. Targets were block-randomized to evenly distribute trials to each target in pseudorandom order.

Performance Measures

We quantified BMI task performance as well as the accuracy with which the monkeys performed the task. Upon successfully initiating a trial, there were two possible task-errors: failure to reach the peripheral target in time, and failure to hold at the peripheral target. We only analyzed performance of "target-in" trials, in which the monkeys were able to reach

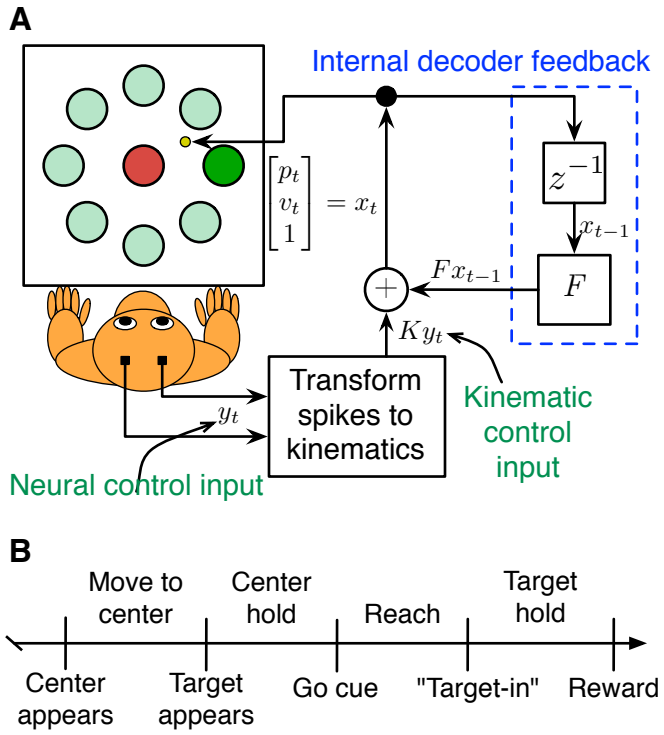

Figure 3.1: (A) BMI task. While controlling a 2D BMI cursor, the monkeys sat in primate chairs with their arms outside of the cursor workspace. Monkeys were required to operate the BMI in a center-out task. Visual feedback of the cursor's decoded position was presented to the monkeys to enable closed-loop BMI operation. Recursive decoders, like the KF, use internal decoder feedback to generate new decoder outputs based partially on previous decoder outputs. The BMI is represented as a linear dynamical system in this diagram, which can represent many commonly used BMI algorithms including the population vector algorithm, Wiener filter and Kalman filter. (B) Time-line of task events.

the peripheral target, including both successful trials and trials that ended in target hold errors (see Figure 3.1B). BMI performance was assessed using three metrics:

1. Hold error rate: the number of target hold errors per successful trial, e.g. a hold error rate of 2 corresponds to 2 hold errors per successful trial. Target hold errors had a stricter penalty than center hold errors, as the monkey was required to return to the center before re-attempting the target hold. Hold errors have been identified by numerous studies as particularly problematic for BMIs [3, 11, 13, 24].

2. Cursor speed: the displacement of the cursor per second, calculated continuously (i.e., every new decoder output yielded a new speed measurement) during the time-limited "reaching" epoch (see Figure 3.1B).

3. Movement error (ME): the average distance between the cursor's position and the "task axis", the straight line between the center and peripheral target, per trial [24]. We discarded a small percentage of highly inaccurate trials in which the cursor touched one of the other targets.

Lower values correspond to better performance for all of these performance measures. The number of initiated trials also varied across sessions because the task was self-paced and the duration of experimental blocks varied. In statistical tests across sessions with different numbers of trials, each target-in trial was given equal statistical weight (e.g. weighted regression analysis).

Simulating BMI control using linear-quadratic control

Simulations of optimal control provide a useful mechanism for qualitatively understanding how a fixed control strategy behaves when the properties of the system being controlled changes. We simulated center-out reaches by generating control inputs to the BMI system using an optimal linear feedback controller with no control noise. In mathematical terms, the BMI system is simulated as

$$\begin{bmatrix} p_t \\ v_t \\ 1 \end{bmatrix} = F \begin{bmatrix} p_{t-1} \\ v_{t-1} \\ 1 \end{bmatrix} + \begin{bmatrix} 0_{2\times 1} \\ u_t \\ 0 \end{bmatrix}. \tag{3.2.1}$$

The two-dimensional velocity control signal u_t was designed to drive the cursor from an initial rest position x_0 to a final rest position x_{targ} while minimizing the linear-quadratic cost function $\sum_{t=1}^{T}(x_t - x_{targ})^T Z(x_t - x_{targ}) + u_t^T R u_t$. The optimal feedback controller L is given by the linear-quadratic regulator (LQR) solution for the system in (3.2.1) if Z, R, and a model of F are known [36]. Thus, the model of F used for controller design, F_{model}, is analogous to the BMI subject's "internal model" of cursor dynamics. Both Z and R were set to 1 to avoid extreme control inputs (e.g. on-off/bang-bang controllers). The feedback controller generated the control input sequence $u_t = Lx_t$ in both open- and closed-loop. The source of x_t used to generate the input u_t differentiated open- versus closed-loop execution. Open-loop execution used x_t^{OL} pre-computed using the internal model:

$$x_t^{OL} = F_{model}^t x_0 + \sum_{k=1}^{t-1} F_{model}^{t-k} \cdot u_k; \quad u_t^{OL} = L x_t^{OL}.$$

In general, $x_t \neq x_t^{OL}$ due to control noise and/or mismatch between the internal model and the true system. In contrast, the closed-loop controller uses the true x_t, i.e. $u_t^{CL} = Lx_t$.

For our simulations, the internal model was set to be "physically realistic", assuming the cursor position to be set by integrating independent horizontal and vertical velocity

components:

$$F_{model} = \begin{bmatrix} I_2 & 0.055I_2 & 0_{2\times1} \\ 0_{2\times2} & 0.6I_2 & 0_{2\times1} \\ 0_{1\times2} & 0_{1\times2} & 1 \end{bmatrix}. \tag{3.2.2}$$

We picked the values 0.055 and 0.6 from the range of experimentally observed instances of F. In Section 3.3, we explored how the controller designed with F_{model} behaved when there was a one-parameter mismatch between the true F and F_{model}.

3.3 Results

Comparisons of dynamical systems underlying linear decoding algorithms

BMI systems that use linear decoders, including the KF, produce the estimate of intended kinematics $\tilde{x}_t$ using the previous estimate x_{t-1} and the current spike observations y_t, as shown in (2.3.1):

$$x_t = (I - K_tC)Ax_{t-1} + K_ty_t = F_tx_{t-1} + K_ty_t. \tag{3.3.1}$$

We refer to (3.3.1) as the BMI system equation. F_t is the closed-loop state transition matrix of the system, and K_t is the control input matrix. In the KF, F_t and K_t are determined by the initial cursor state and all of the model parameters (not just the state-space model).

In stable KFs with fixed model parameters, F_t and K_t converge to F and K, respectively. This convergence occurs within seconds at the update rate of BMIs. This "steady-state" Kalman filter has been used in offline predictions in place of the standard KF to reduce computational overhead [37]. Rapid convergence enables us to analyze KF decoders as time-invariant linear dynamical systems and exposes decoder properties that directly impact BMI control. Figure 3.1A shows this system in block-diagram form.

For 2D linear position/velocity BMIs, we can further sub-divide F and K into components related to position and velocity:

$$\begin{bmatrix} p_t \\ v_t \\ 1 \end{bmatrix} = \begin{bmatrix} T & S & \bar{p} \\ M & N & \bar{v} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{t-1} \\ v_{t-1} \\ 1 \end{bmatrix} + \begin{bmatrix} K_{pos} \\ K_{vel} \\ 1 \end{bmatrix} y_t \tag{3.3.2}$$

In (3.3.2), T and M determine position-dependent dynamics while S and N determine velocity-dependent dynamics. In this context, dynamics refers generally to properties of a dynamical system. The control input sub-matrices $K_{pos} = K_{[1:2,:]}$ and $K_{vel} = K_{[3:4,:]}$ map spike observations to control inputs for position and velocity, respectively.

The duration of time that a velocity control input $K_{vel}y_t$ influences the system state x_t depends on N: $N = 0$ allows a control input to be active for exactly one system iteration, $||N||_2 \geq 1$ creates an unstable system in which control inputs affect the system state forever, and $0 < ||N||_2 < 1$ causes the influence of a control input to smoothly decay over multiple

time steps. In this last case, which represents all KF decoders used in practice, the time duration a control input remains relevant to the system state depends on the magnitude of N. Consequently, we will say that the system has a control memory quantified by $||N||_2$. In contrast, position control inputs $K_{pos}y_t$ generate "instantaneous" position displacement, which is characteristically different from the multiple time steps required for a velocity control input to enact a position displacement.

The linear systems perspective enables us to easily identify common features of different linear decoding algorithms. For example, the VKF state $x_t = \begin{bmatrix} v_t^{horiz} & v_t^{vert} & 1 \end{bmatrix}^T$ contains only the velocity components. Position estimates are generated by "integrating" the decoded velocity: $p_t = \Delta v_t + p_{t-1}$, where Δ is the rate at which new decoder outputs are generated. The VKF system equation reduces to

$$\begin{bmatrix} p_t \\ v_t \\ 1 \end{bmatrix} = \begin{bmatrix} I_2 & \Delta I_2 & 0 \\ 0 & N & \bar{v} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{t-1} \\ v_{t-1} \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ K_{vel} \\ 1 \end{bmatrix} y_t.$$

The steady-state VKF is a special case of the steady-state PVKF, governed by a similar but more structured system driven only by the velocity terms.

The ReFIT-KF [11] retains some, but not all, properties of the PVKF. The ReFIT-KF uses the same neural observation model as the PVKF, but decodes position deterministically, like the VKF [11]. The ReFIT-KF thus has a decoder structure in between the PVKF and the VKF:

$$\begin{bmatrix} p_t \\ v_t \\ 1 \end{bmatrix} = \begin{bmatrix} I_2 & \Delta I_2 & 0 \\ M & N & \bar{v} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{t-1} \\ v_{t-1} \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ K_{vel} \\ 1 \end{bmatrix} y_t. \tag{3.3.3}$$

The closed-loop implications of allowing $M \neq 0$, the major difference in decoder structure between the VKF and the ReFIT-KF, are discussed in Section 3.3.

To acquire the standard population vector algorithm (PVA), we only need to modify the VKF to make $N = 0$, which updates the cursor based only on the instantaneously decoded velocity. However, the velocity decoded by the PVA is often smoothed before integration (e.g. [34] uses a boxcar filter). Any linear filter can be absorbed into (3.3.1). For example, when we apply the parameter constraints

$$A = \begin{bmatrix} I_2 & \Delta I_2 & 0 \\ 0 & aI_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}, W = \begin{bmatrix} 0 & 0 & 0 \\ 0 & wI_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, C^T Q^{-1} C = \begin{bmatrix} 0 & 0 & \Sigma_p \\ 0 & dI_2 & \Sigma_v \\ \Sigma_p^T & \Sigma_v^T & \Sigma_1 \end{bmatrix}, \tag{3.3.4}$$

to the PVKF, the system becomes

$$\begin{bmatrix} p_t \\ v_t \\ 1 \end{bmatrix} = \begin{bmatrix} I_2 & sI_2 & \bar{p} \\ 0 & nI_2 & \bar{v} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{t-1} \\ v_{t-1} \\ 1 \end{bmatrix} + \begin{bmatrix} \gamma K_{vel} \\ K_{vel} \\ 1 \end{bmatrix} y_t. \tag{3.3.5}$$

(see Appendix for details). This system is equivalent to a PVA decoder with a first-order infinite impulse response low-pass filter. The first two constraints force the KF state-space

model to treat horizontal and vertical velocity independently. The third constraint asserts that the vertical tuning models should be orthogonal. Because A and W are never recalibrated in our system, only one degree of freedom remains to determine both n and s. We will refer to all three constraints collectively as the independent velocity control (IVC) constraints. Under these constraints, the control memory $||N||_2$ simplifies to $||N||_2 = n$.

More generally, y_t can be replaced with spike inputs from multiple time-steps, e.g., $\tilde{y}_t = \begin{bmatrix} y_t^T & \cdots & y_{t-J}^T \end{bmatrix}^T$ to accommodate the multi-timestep history often decoded using Wiener filter decoders, also referred to as optimal linear estimators (e.g. [1, 3, 8, 9, 12]). The Wiener filter can be written simply as $x_t = K_{WF}\tilde{y}_t$. Just as for the PVA, F can be extended to capture any static filtering applied to the decoded output.

A high degree of similarity exists between the various types of linear decoders used in BMI demonstrations. Because the PVKF permits the most possible dynamical properties in F, the remainder of the Results analyzes how various PVKF properties may affect BMI performance, both theoretically (Section 3.3) and in BMI experimental data (Sections 3.3-3.3).

Closed-loop properties of linear decoders

In this section, we demonstrate possible closed-loop dynamical properties of linear position-velocity decoders using simple optimal control simulations described in Section 3.2. The feedback control strategy is based on an "internal model" of the true behavior of the system being controlled, in this case the BMI. For the system in (3.2.1), the internal model used was physically realistic as described previously in Section 3.2. This physically realistic structure was not established in the PVKF when using standard MLEs to estimate parameters, as we show at the end of this section. The presence of non-physical system properties may affect performance due to mismatch between the control strategy and the true system properties. In the following simulations, the true F differed from F_model in (3.2.2) by exactly one element. Each simulation perturbed a different parameter and we simulated open- and

closed-loop control in each condition:

$$F_{\Delta T_{[1,1]}} = \begin{bmatrix} \mathbf{0.98} & 0 & 0.055 & 0 & 0 \\ 0 & 1 & 0 & 0.055 & 0 \\ 0 & 0 & 0.6 & 0 & 0 \\ 0 & 0 & 0 & 0.6 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$F_{\Delta M_{[1,1]}} = \begin{bmatrix} 1 & 0 & 0.055 & 0 & 0 \\ 0 & 1 & 0 & 0.055 & 0 \\ \mathbf{0.03} & 0 & 0.6 & 0 & 0 \\ 0 & 0 & 0 & 0.6 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$F_{\Delta N_{[1,2]}} = \begin{bmatrix} 1 & 0 & 0.055 & 0 & 0 \\ 0 & 1 & 0 & 0.055 & 0 \\ 0 & 0 & 0.6 & \mathbf{-0.1} & 0 \\ 0 & 0 & 0 & 0.6 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$F_{\Delta(\Delta n)} = \begin{bmatrix} 1 & 0 & 0.055 & 0 & 0 \\ 0 & 1 & 0 & 0.055 & 0 \\ 0 & 0 & 0.6 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{0.5} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Each of the selected parameters, $T_{[1,1]}$, $M_{[1,1]}$, $N_{[1,2]}$ and $\Delta n = |N_{[0,0]} - N_{[1,1]}|$, was replaced with its average experimental value to form the "perturbed" system and differs from the controller's internal model (3.2.2) by the single selected parameter. The feedback controller was not updated after the system was altered, so the internal model used for planning and control did not match the true system properties.

Mathematical analysis can predict the effect of each perturbation. The system $F_{\Delta T_{[1,1]}}$ has a property that drives $|p_x|$, the absolute value of the cursor's horizontal position, to 0. The system $F_{\Delta M_{[1,1]}}$ has a property which causes vertical velocity to increase as the position of the cursor moves towards the top of the workspace, creating non-uniform velocity offsets in the vertical dimension of the workspace. The system $F_{\Delta N_{[1,2]}}$ has a property that causes horizontal velocity to decrease as vertical velocity increases, resulting in counterclockwise curl dynamics. Finally, the system $F_{\Delta(\Delta n)}$ has a property that makes the velocity control memory asymmetric, causing the cursor to respond differently to velocity control inputs in the horizontal and vertical dimensions.

The LQR controller planned an open-loop control sequence using its internal model of system properties. Figure 3.1A shows the example planned reach, which was identical for open- and closed-loop control simulations. This control sequence was executed with each of the four perturbed F matrices in turn. Figure 3.1B-E shows this open-loop control executed after modifying $T_{[1,1]}$, $M_{[1,1]}$, $N_{[1,2]}$ and Δn, respectively. Figure 3.1B shows open-loop control

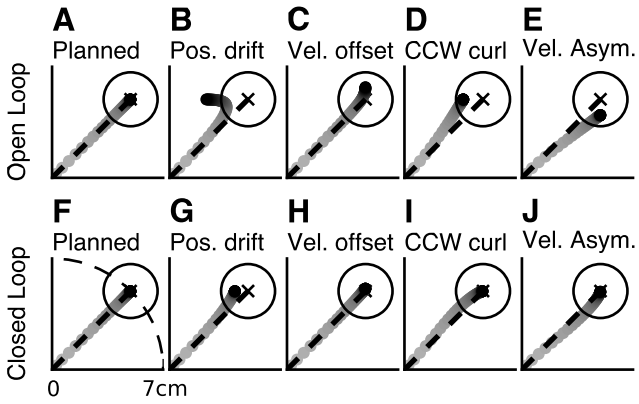

Figure 3.1: Simulations of open-loop and closed-loop optimal control using a fixed internal model. (A) Open-loop planned reach from the center to the peripheral target, where the trajectory begins as gray and transitions to black at the end of the reach. The open-loop control sequence was executed on a system that was slightly different from the internal model used for planning. Depending on the difference between the internal model and the true system, the open-loop trajectory was affected by (B) position drift, (C) non-uniform velocity offsets, (D) counterclockwise curl and (E) undershooting. (F) Same reach planned as in (A), but control inputs were generated using closed-loop feedback control instead of open-loop feedforward control. The closed-loop trajectory was distorted by (G) position drift, (H) non-uniform velocity offsets, (I) counterclockwise curl and (J) clockwise curl due to closed-loop control of an asymmetric system.

of $F_{\Delta T_{[1,1]}}$, where at the end of the reach, the cursor drifts left toward the vertical axis where $|p_x| = 0$. Figure 3.1C illustrates open-loop control of $F_{\Delta M_{[1,1]}}$, where the cursor overshoots the planned vertical endpoint due to the non-zero vertical velocity at the endpoint. Figure 3.1D depicts open-loop control of $F_{\Delta N_{[1,2]}}$, where the cursor trajectory curls off to the left. Intuitively, $F_{\Delta N_{[1,2]}}$ does not allow the cursor to move along the horizontal axis by controlling only the horizontal velocity. Lastly, Figure 3.1E demonstrates open-loop control of $F_{\Delta(\Delta n)}$, where the trajectory was still a straight line reach but short of the target in the vertical dimension. Simply put, open-loop control cannot compensate for any of the perturbations because they were unknown at the time of trajectory planning.

Switching to the closed-loop controller produces better but suboptimal trajectories. Figure 3.1F-J shows closed-loop control executed after modifying $T_{[1,1]}$, $M_{[1,1]}$, $N_{[1,2]}$ and Δn, respectively. For modifications to $T_{[1,1]}$, $M_{[1,1]}$, $N_{[1,2]}$, closed-loop control reduced the impact of dynamical effects that were problematic for open-loop control, but did not eliminate them entirely. Interestingly, Figure 3.1J shows that when controlling $F_{\Delta(\Delta n)}$, a curl effect was present in closed-loop but not open-loop control. In this case, the control policy recognized that the cursor was vertically short of the target, but because it was designed for a system with symmetric velocity dynamics, it made the horizontal velocity larger even though the

original horizontal control was correct. Therefore, unexpected decoder dynamics can have a deleterious effect on BMI performance, even when control inputs are generated in closed-loop and without control noise. These theoretical results imply that non-physical system properties may affect BMI control if the subject plans BMI movements using a mismatched internal model.

In experimental PVKF decoders, all of these non-physical "perturbations" are active simultaneously and interact with each other. For example, we can examine how the static offsets $\bar{p}$ and $\bar{v}$, which were zero for illustration purposes earlier, interact with T and M. The net position offset will be zero under certain conditions, i.e. the effects of T cancel $\bar{p}$ when $(T - I_2)\, p_t + \bar{p} = 0$ and the effects of M cancel with $\bar{v}$ when $Mp_t + \bar{v} = 0$. In experimental decoders, both M and $T - I_2$ were always invertible, establishing $p_T^* = -(T - I_2)^{-1}\bar{p}$ and $p_M^* = M^{-1}\bar{v}$ as cursor attractor points. As shown in (3.5.4), the static offsets $\bar{p}$ and $\bar{v}$ changed based on the neural firing model. Any training method for C and Q must be data-dependent, meaning that the attractor points are also training data-dependent.

Based on this analysis of the dynamical subsystems of the position/velocity decoder, we split F into two components

$$F = \underbrace{\begin{bmatrix} I_2 & sI_2 & 0 \\ 0 & nI_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{F^{vel}} + \underbrace{\begin{bmatrix} T - I_2 & S - sI_2 & \bar{p} \\ M & N - nI_2 & \bar{v} \\ 0 & 0 & 0 \end{bmatrix}}_{F^{offset}}. \tag{3.3.6}$$

The system F^{vel} moves the cursor by propagating forward independent horizontal and vertical velocity states, including the control memory discussed in Section 3.2. The addition of F^{offset} incorporates all of the "offset" properties discussed in this section. The scalar n is set to the average diagonal value of N to minimize $||N - nI_2||_F$ (Frobenius norm); s is defined similarly. With these subsystem splits, the equations governing the motion of the cursor become

$$p_{t+1} = p_t + sv_t + F_{pos}^{offset} x_t + K_{pos} y_t \tag{3.3.7}$$

$$v_{t+1} = nv_t + F_{vel}^{offset} x_t + K_{vel} y_t, \tag{3.3.8}$$

where $F_{pos}^{offset} = F_{[1:2,:]}^{offset}$ and $F_{vel}^{offset} = F_{[3:4,:]}^{offset}$. The offset effects had sizable contribution to the cursor kinematics. Across all PVKF sessions (monkey S: 114, monkey J: 8), F_{pos}^{offset} contributed 0.35±0.15 cm (mean±sd) for monkey S and 0.30±0.07 cm for monkey J, a sizable contribution when compared to $K_{pos}y_t$ (monkey S: 0.47±0.13 cm, monkey J: 0.61±0.13 cm) or when compared with sv_t (monkey S: 0.24±0.02 cm, monkey J: 0.23±0.02 cm). Similarly for the velocity, F_{vel}^{offset} contributed 1.53±0.66 cm/sec for monkey S and 0.92±0.27 cm/sec for monkey J, which was again sizable compared to $K_{vel}y_t$ (monkey S: 2.92±0.33 cm/sec, monkey J: 2.35±0.23 cm/sec) and compared to nv_t (monkey S: 1.48±0.24 cm/sec, monkey J: 2.14±0.19 cm/sec). The offset effects clearly make a non-trivial contribution to cursor motion and we explore their performance impacts further in Section 3.3.

Stochastic position decoding generates workspace attractor points

In Section 3.3, we described two mechanisms by which the inclusion of F^{offset} could generate workspace attractor points, denoted p_T^* and p_M^* based on their respective underlying mathematical causes. We confirm here that the influence of these attractor points is indeed visible in experimental data. In this analysis, we analyzed BMI control of 114 sessions for monkey S and 8 PVKF sessions from monkey J.

The directional variance of $F_{pos}^{offset}x_t$ decreased substantially after accounting for the attractor point p_T^*. For a two-dimensional vector $x = (x_1, x_2)$, we define its direction as $\angle x = \tan^{-1}(x_2/x_1)$; the directional variance of x is the circular variance of $\angle x$ [38]. In the experimental data, the circular variance of $[\angle F_{pos}^{offset}x_t - \angle(p_T^* - p_t)]$ is much less than that of $\angle F_{pos}^{offset}x_t$: 89.9±12.2% (mean±std) less for monkey S and 93.5±2.7% less for monkey J. Figure 3.2A shows an example of the influence of F_{pos}^{offset} as well as the theoretical attractor point for a typical experimental session. Note that theoretically the path along which the attractor point pulled the cursor could have been a curved path but the effects of F_{pos}^{offset} were typically well explained as a straight pull toward p_T^*. The attractor points due to p_T^* in other sessions varied across the workspace unsystematically and changed unpredictably. Thus, for both monkeys, one of the main drivers of cursor motion was driving the cursor toward a fixed position.

The effects of F_{vel}^{offset} were less clearly defined than those of F_{pos}^{offset}. For monkey J, the directional variance of $F_{vel}^{offset}x_t$ decreased 61.7±28.1% after accounting for p_M^*. Monkey S contributed many more sessions than monkey J, and the results were more complex due to the greater diversity of dynamical effects. For 62 sessions, the standard deviation of the direction of $F_{vel}^{offset}x_t$ was less than 15 degrees and thus well explained as a pull in a particular direction (rather than towards a point). For 64 sessions, the directional variance of $F_{vel}^{offset}x_t$ decreased at least 25% after accounting for the location of the attractor point; the variance reduction for this subset of the data was 62.2±17.4%. 32 sessions were well described by both models (e.g. if p_M^* were far enough from the workspace that $p_M^* - p_t$ was approximately constant across the workspace); thus 94 sessions (64+62-32) are well described by one of the two models. For the 20 remaining sessions, these models were insufficient to describe F_{vel}^{offset}, indicating more complicated interaction between the offset effects. Figure 3.2B shows an example of $F_{vel}^{offset}x_t$ during a representative experimental session, and its behavior is clearly more complex than that of $F_{pos}^{offset}x_t$ in the same session, shown in Figure 3.2A.

Simulations and experimental data indicate that the attractor points may be due to parameter estimation artifacts rather than the monkey's control strategy. To investigate how properties of the MLEs can affect decoder properties, we simulated the neural activity of 25 velocity-tuned neurons with uniformly distributed preferred directions and modulation depth of 0.7 spikes per cm [39] in response to artificial center-out trajectories. Attractor points in these simulation conditions must be artifacts because the true neural tuning was invariant to the cursor position; a positional dependence could only emerge due biases in the training data, e.g. finite data effects or the correlation between velocity and position in

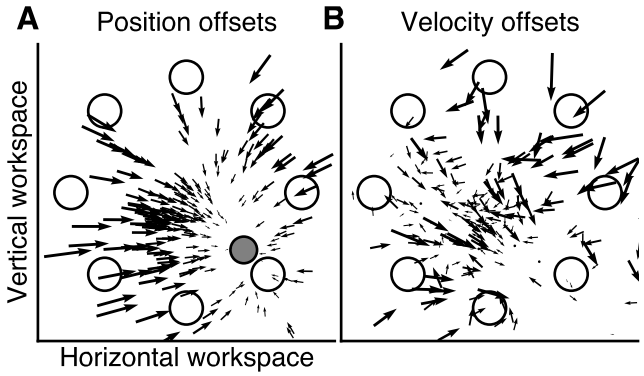

Figure 3.2: Experimental examples of PVKF "offset" properties. The evolution of the cursor's position and velocity, which were updated every Δ=100 ms, can be mapped back to 3 different sub-systems of the KF: the direct "control input", the integration of the previous velocity, and lumped "offset" characteristics of the system, as shown in (12) and (13). (A) Examples of the position offset properties in an experimental position-velocity decoder, which we show in Section 3.3 can be explained primarily as movement toward an attractor point (gray dot). (B) Velocity offset properties for the same experimental session as in (A), which generally cannot be described as compactly as the position offset properties. As we show in Section 3.3, both position and velocity offsets have non-trivial contributions to the kinematics of the BMI.

center-out trajectories. However, across 100 simulations, p_T^* again accounted for 91.8±9.1% of the directional variance of $F_{pos}^{offset}x_t$, similar to the experimental data, and the attractor points found were spread across the workspace, even though trajectories and neural tuning properties were identical in all simulations. Moreover, for subject J, application of the IVC constraints resulted in decreases to both the median $||\bar{p}||$ and the median $||\bar{v}||$ compared to PVKF estimates of $\bar{p}$ and $\bar{v}$: $||\bar{p}||$ decreased by a factor of 30 (Kruskal-Wallis ANOVA, p<0.001) and $||\bar{v}||$ decreased by a factor of 9 (Kruskal-Wallis ANOVA, p<0.001). That is, when position-dependent offset properties were eliminated, the static offset properties diminished without constraint. This implies that the necessity of the static offsets depends on the presence of position-dependent offset properties within the system, and the emergent attractor points in unconstrained PVKFs could be modeling artifacts rather than features of the monkey's neural control strategy.

Our results offer an explanation for other studies that have described a negative impact of stochastic position models on BMI performance. As we described in Section 3.3, the attractor points cannot exist if $T = I_2$ and $M = 0$. Other decoding algorithms automatically impose such constraints by removing stochastic position-related components, including the VKF, ReFIT-KF, PVA and PVKF under IVC constraints, and experiments of [11] and [24] both show performance improvements if $T = I_2$. Consistent with those results, monkey S's hold error rate improved as $||F_{pos}^{offset}x_t||$ decreased (r=0.34, $p < 0.0005$) and monkey J's hold

error rate when using the PVKF was improved by nearly a factor of 4 with IVC constraints (Kruskal-Wallis test, $p < 0.0005$). In other words, hold performance improved when p_T^* decreased in relevance or was eliminated entirely. Additionally, our results indicate that for monkey S, hold performance improved as $||F_{vel}^{offset} x_t||$ decreased (r=0.20, p<0.05) and when $||N - nI_2||$ decreased (r=0.19, p<0.05). Correlations with cursor speed and movement error were not significant, suggesting that $||F_{pos}^{offset} x_t||$ and $||F_{vel}^{offset} x_t||$ have their greatest impact on performance during the precision holds. However, the effects of F_{vel}^{offset} were complex, and making strong conclusions about its performance impact may require future experiments. From (3.3.3) it can be observed that the ReFIT-KF allows one attractor point but not the other (p_M^* but not p_T^*). The experiments of Shenoy and colleagues show that ReFIT-KF outperforms a VKF trained on arm movements and their work discusses the benefits of modeling the neural response to the cursor's position [11]. Our analysis here shows that these position models can be used without creating any attractor points if F is appropriately structured. Many options may be possible to achieve such structure, e.g. by intervening in the operation of the KF as is performed by Shenoy and colleagues [11]. Eliminating static attractor points while retaining a position-dependent neural model might result in further performance improvements if neural firing is significantly related to cursor position, but further experimentation is required to test this hypothesis.

Model-driven tradeoff between accuracy and speed

In Section 3.3, we noted that $||N||_2$ quantifies the system's control memory which determines how long a velocity control input $K_{vel} y_t$ influences the state of the BMI. Intuitively, a cursor with a longer control memory should be faster but also harder to slow down. To verify this intuition with experimental data, we used the same 114 PVKF sessions from monkey S and 8 PVKF sessions from monkey J used in Section 3.3, as well as 17 PVKF sessions with IVC parameter constraints from monkey J. For both monkeys, the cursor speed increased as $||N||_2$ increased ($r = 0.482$ for monkey S and $r = 0.871$ for monkey J, $p < 0.001$). Furthermore, we also found that the hold error rate increased as $||N||_2$ increased ($r = 0.583$ for monkey S, $r = 0.825$ for monkey J, $p < 0.001$). Figure 3.3 depicts these results graphically, and a similar tradeoff is observed for the movement error. This tradeoff is also observed for monkey J when limiting the data only to experiment sessions where IVC parameter constraints were applied. Again, larger $||N||_2$ correlated with increases in cursor speed ($r = 0.945$, $p < 10^{-7}$) as well as the hold error rate ($r = 0.63$, $p = 0.006$). As discussed in Section 3.3, the control memory is the only free parameter in F after the IVC parameter constraints are applied. Therefore, it is likely that the control memory is responsible for an intuitive tradeoff between speed and accuracy.

In addition to controlling performance tradeoffs, properly constraining the control memory may enable faster decoder calibration. The average trajectories for $||N||_2$ during CLDA were similar across sessions and monkeys. In Figure 3.4A-B, we show the average trajectories of $||N||_2$ over the course of SB adaptation for the most common training time of 25 min-

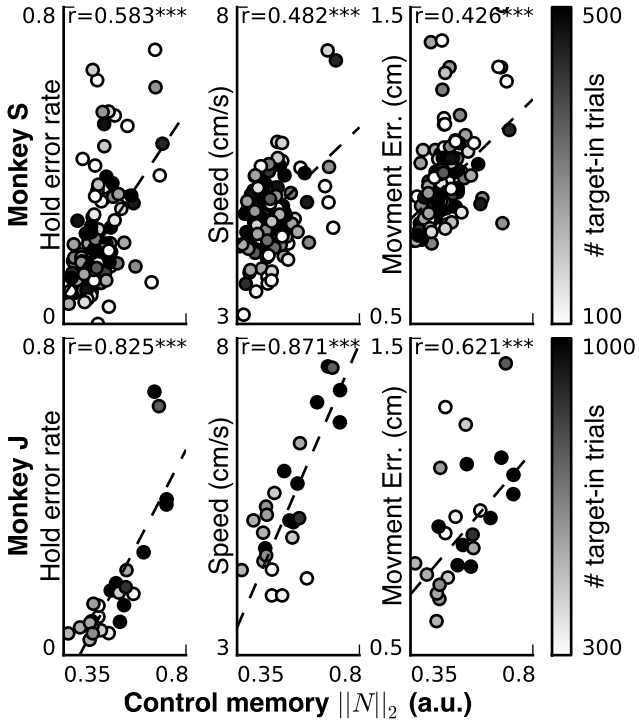

Figure 3.3: BMI "control memory" influences performance. The "control memory" of the linear decoder, quantified by $||N||_2$, is plotted against three experimental performance measures, the hold error rate, mean cursor speed and mean movement error. Each point in the scatter-plot represents one set of decoder parameters and the color of each point represents the number of trials for which that decoder parameter set was used in experimental sessions. For both monkeys, as $||N||_2$ increased, mean cursor speed increased, but so did the hold error rate and movement error. Manipulating the system's control memory setting may enable the BMI designer to manipulate the tradeoff between cursor speed and hold performance.

utes. The small standard error indicates that $||N||_2$ followed a consistent trajectory across sessions during the course of CLDA calibration; trends during other calibration blocks of different lengths were similar. This data suggests that calibrating the control memory once per monkey and always leaving it fixed will facilitate faster parameter recalibration as well as more consistent performance. In the KF model, the control memory changes based on our confidence in the state-space model, W, as well as our confidence in the observation model, $C^T Q^{-1} C$. These relationships between the KF model parameters and the control memory are observed most simply when the IVC constraints are applied as both $C^T Q^{-1} C$ and W can be represented as scalar values (see Appendix). Figure 3.4C shows how changing $||C^T Q^{-1} C||_2$ can alter $||N||_2$ and thus the system's control memory; a similar relationship can be shown for $||W||_2$. Thus, by crafting the covariance matrices in the KF model, the tradeoff between speed and hold performance can be manipulated.

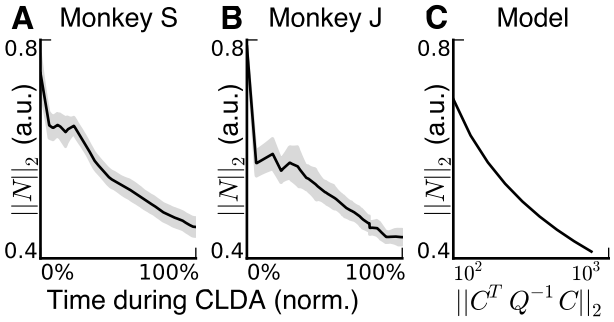

Figure 3.4: The BMI's control memory changes over the course of CLDA: (A-B) Average trajectory and standard error for the BMI's "control memory", quantified by $||N||_2$, during closed-loop calibration of decoder parameter settings. The settings learned for $||N||_2$ are visually consistent across session and across monkey. This suggests that keeping the control memory fixed at a known high-performance value may enable faster parameter calibration using CLDA. (C) For the KF, the control memory changes based on our certainty about the neural tuning model, quantified by $||C^T Q^{-1} C||_2$ (a.u.), as decreasing certainty about the neural tuning model results in a larger control memory and a higher degree of smoothing applied by the decoder. By adjusting relevant covariance matrices in the KF model, the experimenter may be able to manipulate the system's control memory to alter performance tradeoffs described in Section 3.3.

3.4 Discussion

Small changes in closed-loop BMI dynamical properties can have significant and predictable effects on performance. Using the PVKF as a specific algorithm to study, we demonstrated the presence of unintended attractor points which pulled the cursor to arbitrary workspace positions, as well as a calibration-dependent control memory property which parameterized the tradeoff between speed and hold performance. Importantly, both properties were not only present in experimental data but also their mathematical causes could be traced back to the mechanics of the closed-loop decoder. Understanding the basis of these and other closed-loop BMI system properties may be crucial to increasing system reliability by minimizing the noisiness decoder calibration procedures, including CLDA.

None of the PVKF model parameters overtly describe an attractor point, despite its significant effect on the BMI. It is important to clarify that our results do not suggest much, if anything, about the parameters M1 neurons encode in natural movement or even in BMI. Rather, the main takeaway regarding the attractor points should be that the behavior of the linear model was unexpected. Furthermore, depending on the intended purpose of the BMI, the presence of an attractor point is not necessarily problematic. For instance, if an attractor point was intentionally placed at the center of the workspace, then the experimenter could embed a "return to center" property into the decoder. Such a property could speed up the number of center-out trials initiated at the cost of hold performance (which we observed) or

generalization to new target layouts. However, in recalibrations of the PVKF, the attractor point moved across the workspace unpredictably, even in simulation conditions where no attractor point should have existed at all, suggesting that the attractor points were modeling artifacts. Unintended attractor points can be removed through simple changes to the decoding algorithm (e.g. decoding velocity only) or to the structure of the decoder parameters (e.g. applying the IVC constraints). The shortcomings of position decoders are not fundamental limitations. In a linear system such as the PVKF cursor, an LQ controller with knowledge of the system matrices would have little trouble tailoring the control law to match the specific plant properties. This mismatch between experiment and theory suggests that a position strategy might be outside the "manifold" of intuitive theories [21]. For the KF, and generally for recursive Bayesian decoders, the closed-loop properties induced by interaction between the state-space model and the neural firing model can produce unintentional dynamical artifacts.

We demonstrated a tradeoff between speed and hold performance which likely is driven by the fluctuation in the closed-loop control memory of the decoder. The control memory alters smoothing properties, which are known to contribute to performance [**?**]. The optimal tradeoff between speed and hold performance must again depend on the BMI's purpose. For example, in a communication prosthesis requiring only short holds, one may wish to design the control memory to explicitly prioritize cursor speed. However, like the attractor point, control memory and smoothing properties of the KF changed when parameters were recalibrated. We showed that the control memory for the KF changed depending on the neural noise variance parameter estimated during CLDA. The estimated covariance of neural noise could change across sessions due to a plethora of possible causes such as insufficient training data. Indeed during the calibration period we observed consistent decreases in control memory as CLDA received more training data. This highlights a possible disadvantage of attempting to learn all of the parameters from limited training data. For example, we showed that under appropriate mathematical constraints, the PVKF becomes structurally identical to a PVA decoder combined with a smoothing filter. However, the experimenter defines the smoothing filter for the PVA whereas CLDA defines the smoothing filter for the PVKF and is therefore subject to recalibration noise. If it is desirable to tightly control the tradeoff between speed and hold performance to improve performance consistency, then continuously attempting to recalibrate smoothing parameters may not be desirable.

The presence of the PVKF attractor point can be interpreted as CLDA learning an "equilibrium point" for the cursor's position. This is similar to BMI architectures that propose to decode force field parameters [40], with the difference that once CLDA is finished, the neurons can only change the location of the attractor point by altering the baseline firing rates of the ensemble. Future BMIs might benefit from an attractor point that the subject can move across the workspace, enabling hierarchical control where the attractor point defines the coarse location and the cursor's velocity can also be controlled to specify precise location. Exploring this decoder architecture is the subject of future work.

Our previous work [8] shows numerous benefits of keeping both the decoder and the neural ensemble fixed across sessions, including performance improvements and the formation of a

prosthetic "motor memory". In practice, however, it may be necessary to alter the BMI neural ensemble if a cell is lost (due to electrode array shifts, cell death, etc.), and CLDA is an effective tool to calibrate a decoder even from poor initial performance [26]. As we showed here, CLDA can change system properties that we might prefer to keep fixed if we wish to keep the system as consistent as possible before and after recalibration. Our results suggest that the least complex dynamical system may be the easiest to learn (e.g. without attractor points), particularly if the subject tries to learn to improve performance by learning an internal model of the decoder [41]. Regardless, [8] suggests that we ought to keep the decoder as constant as possible, including the F matrix, before and after recalibration. For the KF, this requires restricting the observation model to a subspace (i.e., keeping $C^TQ^{-1}C$ fixed). Doing so may facilitate learning across days even with a changing ensemble, but these hypotheses require further experiments.

In summary, we analyzed experimental data from two monkeys performing closed-loop BMI to gain a better understanding of BMI system properties and their influence on task performance. We demonstrated by simulation that mismatches between the true BMI dynamics and the model used for control can cause predictable errors in the planned trajectory. In particular, this analysis revealed that attractor points intrinsic to the position-velocity Kalman filter had a sizable and deleterious effect in experimental data. Furthermore, we identified dynamical features such as system memory that should be kept static during any recalibration periods so as to improve the consistency of decoder properties across sessions. The performance impact of the dynamical properties we examined may inform the design of future BMI algorithms more compatible with neural control strategies.

3.5 Appendix: KF with Independent Velocity Control (IVC)

Recall from (2.3.1) that the BMI system equation governing Kalman filter (KF) decoders is $x_{t+1} = (I - K_{t+1}C)\,Ax_t + K_{t+1}y_{t+1}$. The Kalman gain K_{t+1} in (2.3.1) is given by

$$K_{t+1} = P_{t+1}C^T\left(CP_{t+1}C^T + Q\right)^{-1},$$

in which $P_{t+1} = var\left(x_{t+1}|\, x_1, \cdots, x_t\right)$ represents a recursively calculated prediction covariance [25]:

$$P_{t+1} = AP_tA^T - AP_tC^T\left(CP_tC^T + Q\right)^{-1}CP_tA^T + W. \tag{3.5.1}$$

The base condition for (3.5.1) depends on the initial uncertainty about the state of the cursor; in our system, $P_1 = W$. We expand $C^T(CP_tC^T + Q)^{-1}C$ with the matrix inversion lemma:

$$\begin{aligned} C^T(CP_tC^T + Q)^{-1}C &= C^T\left[Q^{-1} - Q^{-1}CP_t\left(I + C^TQ^{-1}CP_t\right)^{-1}C^TQ^{-1}\right]C \\ &= \left[I - C^TQ^{-1}CP_t\left(I + C^TQ^{-1}CP_t\right)^{-1}\right]C^TQ^{-1}C \end{aligned} \tag{3.5.2}$$

If

$$P_t = \begin{bmatrix} eI_2 & fI_2 & 0 \\ fI_2 & gI_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{3.5.3}$$

then we can calculate F_t defined in (3.3.1), using (3.5.2) above and the constraints in (3.3.4):

$$F_t = \begin{bmatrix} I_2 & (\Delta - \beta adf)\, I_2 & \bar{p}_t \\ 0 & \beta a I_2 & \bar{v}_t \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.5.4}$$

In (3.5.4), $\beta = 1/(1+dg)$, $\bar{p}_t = (\beta df^2 - e)\,\Sigma_p - \beta f \Sigma_v$, and $\bar{v}_t = -\beta\,(f\Sigma_p + g\Sigma_v)$ which satisfies the desired structure for F_t in (3.3.5). (Σ_p, Σ_v, and Σ_1 are unconstrained). To complete the proof, induction can be used to show that under the initial condition for P_1, P_t has the structure of (3.5.3) for all t. Both the base case (P_2) and inductive case (P_{t+1}) can be verified using (3.5.1) and (3.5.2).

It may be of interest to control the steady-state form F_t. P_t converges in the limit as $t \to \infty$

$$\lim_{t\to\infty} P_t = \lim_{t\to\infty} P_{t+1}.$$

Equating P_{t+1} and P_t above yields

$$\begin{aligned} g &= \frac{a^2 g + wgd + w}{1+gd} = \frac{a^2 g}{1+gd} + w \\ f &= \frac{(f+\Delta g)\, a}{1+gd} \end{aligned} \tag{3.5.5}$$

We observe that f and g are fully determined given a, w and d. Interestingly, under the constraints, e in 3.5.3 is not convergent, but this does not affect the convergence of the Kalman gain. From (3.5.4), we can see that

$$n = \frac{a}{1+gd} \Rightarrow \frac{1}{1+gd} = \frac{n}{a}.$$

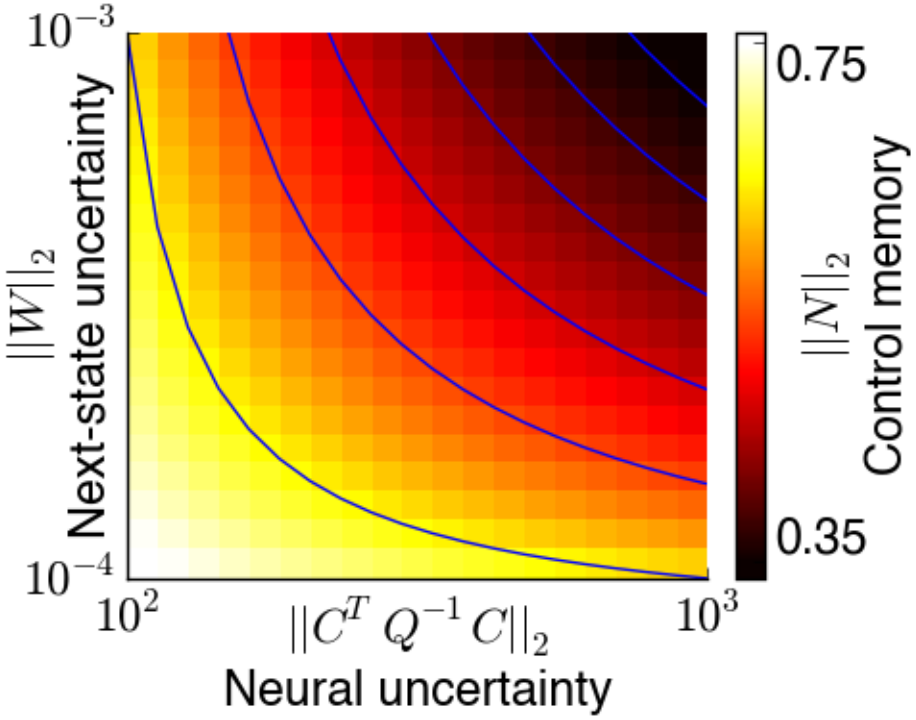

Figure 3.1: Control memory as a function of the covariance matrices. Control memory is redundantly controlled by the next state uncertainty and the neural uncertainty.

We substitute this into (3.5.5)

$$\begin{aligned} g &= \frac{n}{a} \cdot a^2 g + w \\ &= \frac{w}{1-an} \\ n &= \frac{a}{1+d\left(\frac{w}{1-an}\right)} \\ &= \frac{a}{\frac{1-an+dw}{1-an}} \\ &= \frac{a\,(1-an)}{1-an+dw} \\ (1-an+dw)\,n &= a\,(1-an) \\ dw &= \frac{(1-an)\,(a-n)}{n}. \end{aligned}$$

Chapter 4

Extracting sparsity from natural movements

This chapter details the development of a method to quickly extract sparse components from complex kinematic trajectories during natural movements. This work was in collaboration with Simon Overduin, Mo Chen, Young-Hwan Chang, Claire Tomlin and Jose Carmena. A version of this work has been published [42].

4.1 Introduction

Movement kinematics and smoothness are analyzed when probing the nature of neurodegenerative diseases, including Huntington's disease [43], multiple sclerosis [44] and Alzheimer's disease [45]. One commonly employed technique in movement smoothness analysis is to decompose movements into one or more discrete canonical shapes, or "submovements". Superpositions of a few simple submovements, e.g., bell-shaped velocity profiles, can generate complex kinematic trajectories. Submovement decomposition aims to recover the presumed underlying submovement components from the observation of the overall trajectory. An example of such a decomposition is shown in Figure 4.1, where submovement decomposition enables us to break up the movement into different components which may reflect the points at which differing motor plans began execution.

The neural origins of submovements remain only partially understood. One might theorize based on the apparently sparse nature of submovement initiation that motor control may involve intermittent execution of new feedforward submovements in response to movement error [46]. Electroencephalographical correlates of this intermittent control hypothesis have been observed in humans [47]. Alternatively, movement intermittencies may be related to cortical oscillatory dynamics [48], or they may simply reflect a continuous controller interacting with the musculoskeletal system [49]. Nevertheless, quantitatively precise submovement algorithms have many clinical and engineering applications. For instance, analysis has shown that submovements become more overlapping or "blended" as patients recover from stroke

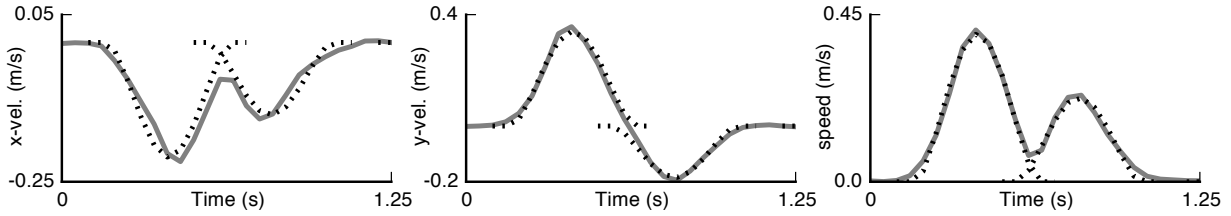

Figure 4.1: An example of submovement decomposition applied to a movement under target perturbation conditions in which the instructed target location changes in the middle of the movement (see Section 4.2 for behavioral details). These perturbation conditions are likely to necessitate a large secondary corrective movement following the perturbation that manifests as a distinct segment in the velocity profile. In this movement (gray line), two separate overlapping submovements are extracted (dotted black lines). More than 98% of the velocity variance can be explained by two appropriately shaped and scaled "minimum-jerk" curves (see Methods).

[50].

Several methods of submovement analysis do not require assumptions about the functional form of the submovement basis functions. These methods may simply split movements into their "primary" and "corrective" components [51], or analyze movement acceleration, jerk, and higher-order derivatives for zero crossings or local extrema to find velocity inflection points [52]. However, higher-order derivatives of movement can be hard to acquire using sensors or to compute numerically without substantial noise. Alternatively, the use of an explicit prototype submovement function (minimum-jerk [53], lognormal [54], etc.) allows for the use of optimization-based methods to extract submovements from position or velocity time-series data. This approach involves curve-fitting by optimizing a non-convex error function. Hogan and colleagues demonstrated that the best results arise from optimizing over all submovement parameters jointly rather than sequentially [55]. For clinical applications where many movements must be analyzed or decompositions must operate in real-time, finding an exact solution requires an unsuitable amount of computation time. Instead, random-restart "scattershot" search retains advantages of joint optimization with orders of magnitude less computation time [56].

Additional reductions in computation time may help enable otherwise infeasible real-time applications of submovement decomposition. In this work, we extend the scattershot method [56] to accelerate computation time. First, we define *shape pursuit,* a method of iteratively fitting submovements when the basis function has shape parameters distinct from amplitude parameters. This condition applies to many submovement function prototypes [54]. Second, we introduce the idea of *error concentration*, which allows for tunable concentration of the iterative search procedure in regions of high error. This heuristic makes it possible to trade exploration of solution space for computation speed as demanded by the application. Both of these innovations yield substantial gains in computation time and improve the suitability

of submovement decomposition for real-time applications.

4.2 Methods

Submovement decomposition as an optimization problem

Suppose that we have a velocity trajectory $v(t)$ that we wish to decompose into N submovements, and that the submovement function can be parameterized as $f(t;\theta,A)$, in which t is the time variable and θ and A are fixed function parameters defining shape (defined precisely in Section 4.2) and amplitude, respectively. For example, the minimum-jerk submovement velocity prototype [53] has 3 parameters, the onset time t_0, the duration D and the amplitude A:

$$f(t;\theta,A) = A \cdot \frac{1}{D} \begin{cases} 30t_n^4 - 60t_n^3 + 30t_n^2 & t \in [t_0, t_0 + D] \\ 0 & \text{otherwise} \end{cases} \tag{4.2.1}$$

in which $\theta = (t_0, D)$ and $t_n = \frac{t-t_0}{D}$ is time normalized to the duration of the movement. A may in general be a column vector (movements may extend through multiple spatial dimensions), so $f(t;\theta,A)$ maps a scalar t to a vector of the same dimension as A. Examples of this type of bell-shaped speed profile are shown in Figure 4.1 (dashed lines). This particular submovement function was previously discovered to be a good characterization of human hand velocity in multi-joint arm movements [53]. Numerous other functions also may produce similar bell shapes [54]. The optimization problem to solve is

$$\min_{\{\theta_i,A_i\}_{i=1}^N} \sum_{t=1}^{K} \left\| v(t) - \sum_{i=1}^{N} f(t;\theta_i,A_i) \right\|_2^2, \tag{4.2.2}$$

in which K is the number of sample points in the trajectory and $||\cdot||_2$ is the ℓ_2-norm of a vector. For a set of submovement parameters $\{\theta_i, A_i\}_{i=1}^N$, let $r(t) = \sum_{i=1}^N f(t;\theta_i,A_i)$ be the *reconstructed movement* and let $c(t) = \|v(t) - r(t)\|_2^2$ be the reconstruction error at time sample t. This optimization problem has multiple local optima (and is thus not convex) and finding a globally optimal solution can be computationally expensive. In the scatter-shot method, local minima of the cost function are found from different initializations of the parameters $\{\theta_i, A_i\}_{i=1}^N$ [56]. With multiple random initializations, the probability of finding the true global optimum increases with the number of initializations. The number of submovements N is found iteratively, by incrementing N and re-solving the entire optimization problem [55]. In contrast, a "greedier" algorithm might attempt to increment N by solving the problem

$$\min_{\theta_N,A_N} \sum_{t=1}^{K} \left\| v(t) - \sum_{i=1}^{N-1} f(t;\theta_i,A_i) - f(t;\theta_N,A_N) \right\|_2^2. \tag{4.2.3}$$

Rather than fitting N submovements to the velocity curve $v(t)$, (4.2.3) fits one submovement to the residual error from fitting $N-1$ submovements, $v(t) - \sum_{i=1}^{N-1} f(t;\theta_i,A_i)$. In this

work, we always jointly optimize over all submovement parameters, including the parameters resulting from previous optimizations, as joint optimization has been shown to be less prone to being trapped in local optima [55].

Note there is no restriction on the dimensionality of the velocity data, as each A can be of dimension $k \times 1$ allowing the optimization to be carried out over multiple kinematic dimensions jointly. In this study, we decompose 2D movements and use

$$v(t) = \begin{bmatrix} v_x(t) & v_y(t) & \sqrt{v_x^2(t) + v_y^2(t)} \end{bmatrix}^T$$
$$A_i = \begin{bmatrix} A_x & A_y & \sqrt{A_x^2 + A_y^2} \end{bmatrix}^T,$$

in which the third element of both vectors is the tangential speed. The inclusion of the tangential speed term acts as a regularization term. Concretely, suppose that we have a movement where the true $v_x(t) = f(t; \theta_{ex}, 1)$ and $v_y(t) = 0$. Then the tangential speed of this movement is $v_s(t) = v_x(t)$. If we happen to choose $r(t) = f(t; \theta_{ex}, [2, 0, 2]^T) + f(t; \theta_{ex}, [-1, 0, 1]^T)$, then $r(t)$ perfectly fits $v_x(t)$ and $v_y(t)$ since the two submovements partially cancel each other. However, this is clearly an example of overfitting and this particular type of overfitting results in a bad fit for $v_s(t)$. Since the submovements cannot cancel each other in the speed dimension, $v_s(t)$ is fit with an amplitude of 3 when it should have had an amplitude of 1. Thus, the inclusion of the tangential speed dimension into the optimization penalizes fits of overlapping submovements with opposite signs. Importantly, it does not preclude such destructive cancellation; an example is shown in Figure 4.1 in the y-velocity, where the two submovements of opposite signs slightly overlap. This type of destructive cancellation may reflect a new motor plan being vectorially combined with the ongoing motor plan, e.g., to abruptly change reach direction [57]. Thus we aim to penalize such movements to avoid overfitting but not eliminate them altogether.

Although we selected an ℓ_2-norm cost function in (4.2.2), there is no inherent restriction to do so. Indeed, the choice of cost function may be motivated by computational rather than physiological considerations. For example, [55] uses a branch-and-bound algorithm with an ℓ_1-norm cost function for computational speed. Our choice of an ℓ_2-norm cost function was also motivated by computational considerations, as described in Section 4.2.

For a given movement segment, we incremented the number of submovements fit until 1) success, assessed when the residual error fell below a mean squared error (MSE) threshold (e.g., 2%), 2) insufficient progress was being made, i.e., when incrementing the number of submovements resulted in an error reduction less than a threshold (e.g., 0.1% MSE), or 3) the maximum number of submovements was hit, where the maximum was set based on a restriction that the minimum time between submovement start times be 100ms (a conservative minimum reaction time to generate a new submovement as a visual feedback correction).

Innovation 1: Shape pursuit (SP)

In (4.2.1), the shape parameters θ alter the submovement function independently of the amplitude parameter A: $f(t;\theta,A) = A \cdot f(t;\theta,1)$. In this case, which describes many submovement functions [54], (4.2.2) becomes

$$\min_{\{\theta_i,A_i\}_{i=1}^N} \sum_{t=1}^{K} \left\| v(t) - A_i \sum_{i=1}^{N} f\left(t;\theta_i,1\right) \right\|_2^2. \tag{4.2.4}$$

Let

$$V = \begin{bmatrix} v(1) & \cdots & v(K) \end{bmatrix}^T$$
$$F = \begin{bmatrix} f\left(1;\theta_1,1\right) & \cdots & f\left(1;\theta_N,1\right) \\ \vdots & \vdots & \vdots \\ f\left(K;\theta_1,1\right) & \cdots & f\left(K;\theta_N,1\right) \end{bmatrix}$$
$$M = \begin{bmatrix} A_1 & \cdots & A_N \end{bmatrix}^T.$$

The optimization cost function can be rewritten in matrix form as $\|V - FM\|_F^2$, in which $\|\cdot\|_F$ is the Frobenius norm. Once the shape parameters (F) are fixed, we can optimize for the amplitude parameters (M). Let V_x, V_y, and V_s be the first, second and third columns of V, and define M_x, M_y, and M_s similarly. We seek to solve the problem

$$\begin{aligned} \min \quad & \|V_x - FM_x\|^2 + \|V_y - FM_y\|^2 + \|V_s - FM_s\|^2 \\ \text{subject to} \quad & M_{x(i)}^2 + M_{y(i)}^2 - M_{s(i)}^2 = 0 \quad i \in \{1,\cdots,N\}. \end{aligned}$$

The constraints are necessary to ensure that the last column is consistent with its interpretation as the tangential speed scaling factor. A straightforward application of Lagrange multipliers yields a system of highly coupled equations. A similar variable coupling occurs in ℓ_1 regularized regressions [58]. For computational reasons, we attempt two relaxations. First, we notice that the N constraints above imply that $M_x^T M_x + M_y^T M_y - M_s^T M_s = 0$, though the converse is not true. Hence we can relax the problem by replacing the N original constraints with the single condition $M_x^T M_x + M_y^T M_y - M_s^T M_s = 0$ and then apply the Lagrange multiplier method:

$$\begin{aligned} \mathcal{L}(M,r) &= \|V_x - FM_x\|^2 + \|V_y - FM_y\|^2 + \|V_s - FM_s\|^2 \\ &\quad + r\left(M_x^T M_x + M_y^T M_y - M_s^T M_s\right) \\ \frac{\partial}{\partial M_x}\mathcal{L} &= -2F^T\left(V_x - FM_x\right) + 2rM_x = 0 \\ M_x &= \left(F^T F + rI\right)^{-1} F^T V_x. \end{aligned} \tag{4.2.5}$$

Similar equations follow for M_x and M_s. The Lagrange multiplier method provides a method of selecting the optimal r, but it again requires solving a complicated equation. However,

we note that (4.2.5) is a ridge regression in which the ridge parameter r acts to shrink the amplitudes of each movement [59]. We selected $r = 0.5$, making an implicit prior assumption that the submovement amplitudes (in meters) are twice as variable as the observed velocity (in meters/sec). In reality, we expect that the velocity variance will be greater than the submovement amplitude, since the peak velocity of a movement (in meters/sec) is typically greater than its total displacement (in meters). By assuming submovement amplitudes more variable than observed velocity, we apply a conservative shrinkage penalty to the submovement amplitudes. We refer the reader to [59] for an extended discussion of this Bayesian interpretation of ridge regression. Importantly, although we have made two simplifications to the optimization problem to calculate the amplitude parameters, the cost function we use to evaluate the quality of solutions remained the same across all methods.

A simple 1D illustration of the shape pursuit (SP) concept is shown in Figure 4.1A, where we attempt to fit one submovement to the black curve. For an estimate of the submovement shape parameters $\hat{t}_0$ and $\hat{D}$, the cost varies quadratically as a function of the amplitude (Figure 4.1A, bottom). For these fixed shape parameters, we can jump directly to the optimal amplitude rather than seeking the optimal amplitude by an iterative method. On the next iteration, the shape parameters are refined and subsequently the amplitude parameters are again found using (4.2.5). Our original optimization problem thus becomes an alternating minimization problem in which we alternatively fit shape and then amplitude parameters. Shape pursuit solves the problem

$$\min_{\{\theta_i\}_{i=1}^N} \left\| V - FV(F^TF + rI)^{-1}F^T \right\|_F, \tag{4.2.6}$$

and makes the search space smaller by reducing the number of variables to search over. For other types of cost functions, e.g., ℓ_1-optimization, the same partitioning of the parameter space can be applied, but different methods must be used to find M.

Importantly, the scale-shape independence property which we exploit is applicable to submovement functions other than the minimum-jerk function considered above. For instance, Gaussian functions also produce a bell shape, and we could use

$$g(t; t_0, D, A) = A \cdot \exp\left\{ -\frac{1}{D^2}\left[t - \left(t_0 + \frac{D}{2} \right) \right]^2 \right\}$$

as the submovement function. In this case, the movement is no longer finite length; $t_0 + \frac{D}{2}$ represents the midpoint of the movement (peak velocity) and D represents the duration (smaller D corresponds to more concentration around the midpoint). However, we still retain the property that $g(t; t_0, D, A) = A \cdot g(t; t_0, D, 1)$, meaning that the shape pursuit method still applies. Since the focus of this work is on the computational methods and not the function type, we only present results for the minimum-jerk function.

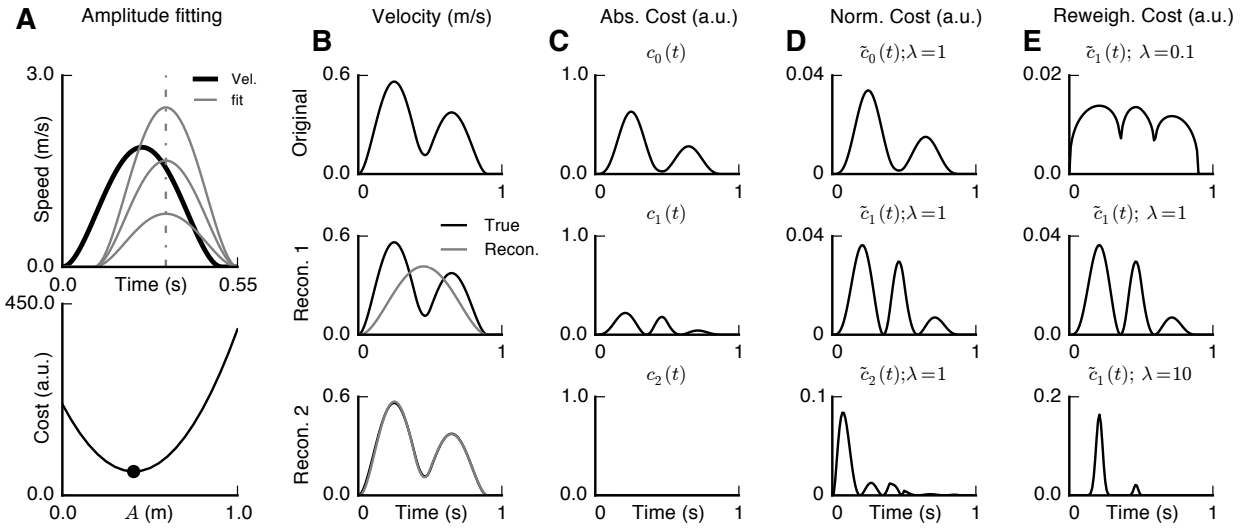

Figure 4.1: Submovement decomposition innovations. *(A)* Top: An example of a simulated bell-shaped speed profile. For the (incorrect) estimates of the shape parameters $\hat{t}_0$ and $\hat{D}$ shown, the submovement of "unit" norm and two amplitude-scaled submovements, including the submovement of best fit (i.e., best scale parameter for the given shape parameters), are shown. Bottom: The cost function for the same fixed $\hat{t}_0$ and $\hat{D}$ as a function of the movement amplitude A. The cost function is quadratic (and therefore convex) in A, enabling it to be found quickly once the shape parameters are fixed. *(B)* Simulated velocity profiles as well as the results of fitting one submovement and fitting two submovements. *(C)* The point-by-point cost functions $c_0(t)$, $c_1(t)$, and $c_2(t)$ from fitting zero, one and two submovements. *(D)* Cost functions from *(C)* are normalized to sum to one to form $\tilde{c}_0(t)$, $\tilde{c}_1(t)$, $\tilde{c}_2(t)$ with no reweighting ($\lambda = 1$, see Section (4.2)). *(E)* Different reweighting factors λ can be used to make the normalized cost more flat (top) or peaked (bottom), enabling control of the "greediness" of the search concentration.

Innovation 2: Error concentration (EC)

Recall from Section 4.2 that we solve the optimization problem (4.2.2) several times from various initialization points for a fixed N. One approach to parameter initialization is to sample independently from uniform distributions for each parameter on each restart. This is the approach taken by scattershot and the approach that we apply to SP. However, the number of restarts must be small in order to accrue computational savings. As the number of submovements being fit increases, a small number of random samples in an exponentially growing search space has increasingly poor coverage. To combat this, error concentration (EC) initializes parameters greedily. In particular, if we have already found $N-1$ submovement parameters but now we want to fit N submovements, we recycle the old parameters $\{\theta_i\}_{i=1}^{N-1}$ and append new randomly sampled parameters θ_N. Furthermore, we concentrate θ_N to represent a region of relatively high error by picking a submovement shape which spans

two adjacent local minima in the *reweighted normalized cost* $\tilde{c}(t) = c(t)^\lambda / \int c(\tau)^\lambda d\tau$ in which τ spans the time samples. The exponent λ is a reweighting factor [60]: $\lambda > 1$ makes the function closer to a "delta" (more "peaked"), $\lambda < 1$ makes the function more uniform, and $\lambda = 1$ results in no reweighting (only normalization). Figure 4.1C-E shows simulation examples of the cost, normalized cost and reweighted cost. As λ increases, there is a reduction in the number of local minima from which to initialize θ_N; thus the greediness of the parameter initialization can be quantified and controlled.

While we initialize parameters greedily, we still perform joint optimization over all parameters. In other words, EC only deals with parameter initialization; the actual optimization routine runs the same as SP. The cost residual $c(t)$ must be calculated and new parameters must be sampled from some distribution with or without EC. The predominant computational expense of EC is to exponentiate the residual $c(t)$, which is an expense similar to a single evaluation of the cost function. Furthermore, this minor expense is only incurred when the number of submovements is incremented. Hence EC is in practice free when compared to the expense of the actual optimization routine. In our results, we used $\lambda = N$ so that the greediness of the initialization scaled roughly with the expansion of the search space.

Optimization methods

We primarily used MATLAB's *fmincon* to find locally optimal solutions using sequential quadratic programming (SQP) to solve both (4.2.2) and (4.2.5). The complexity of evaluating the cost function scales linearly as the number of submovements being fit increases because (4.2.1) must be evaluated once for each submovement i and each time-point t. In standard big-O notation, the complexity of evaluating (4.2.2) is $O(NK)$, where N is the number of submovements and K is the number of time samples in the movement segment. Averaged over 10,000 evaluations, the average computation time to evaluate the cost function scaled linearly with the number of submovements ($R^2 > 0.99$, $p < 10^{-19}$, evaluated up to 20 submovements). Evaluating the SP cost function required strictly more time than evaluating the original cost function (4.2.2) in our implementation. We first found the amplitude parameters by evaluating (4.2.5) and then proceeded to evaluate the original cost function (4.2.2).

We supplied an analytical gradient when using SQP. For the minimum-jerk prototype function, the complexity of evaluating the cost function and computing the gradient also scaled linearly with the number of submovements ($R^2 > 0.99$, $p < 10^{-19}$, evaluated up to 20 submovements). Computing both the gradient and cost required about $1.6\times$ more time that computing the cost alone, and did not change significantly as the number of submovements increased ($R^2 = 0.13$, $p = 0.11$). The same gradient was used for solving both (4.2.2) and (4.2.5). In the case of SP, it is only an approximate gradient since the amplitudes have been removed from the optimization variables.

We compared the performance of 1) the scattershot (SS) algorithm [56], 2) scattershot combined with shape pursuit (SP), and 3) scattershot combined with shape pursuit and error concentration (SPEC). We also compared the performance of the gradient-based SQP to two non-gradient metaheuristics. Specifically, we assessed the performance of covariance matrix

adaptation evolution strategy (CMA-ES) [61] and simulated annealing (SA). For CMA-ES, we used the implementation corresponding to [61]. For SA, we used the implementation built into MATLAB's optimization toolbox. These methods do not utilize gradient information and instead rely on random search for exploration. We applied CMA-ES and SA to the original optimization problem (4.2.2), which we refer to as CMA-ES-SS and SA-SS, respectively. We also applied them to the SP equivalent problem (4.2.6), which we refer to as CMA-ES-SP and SA-SP, respectively. A pseudocode outline of parameter initialization and optimization is provided in Algorithm 1.

Behavioral data

Three adult male macaques (*Macaca mulatta*), J, P and R, were used in this study. All procedures were conducted in compliance with the National Institutes of Health Guide for Care and Use of Laboratory Animals and were approved by the University of California, Berkeley Institutional Animal Care and Use Committee.

All monkeys performed variants of a 2D point-to-point reach and hold task. The monkeys initiated trials by moving their right hand inside a KINARM exoskeleton (BKIN Technologies, Kingston, ON, Canada) to the visually-instructed origin and holding for 400-1500ms. Upon entering the origin, the terminus appeared. The monkeys were then required to reach to the specified terminus within a 3-7 second time-limit and hold for 400-1500ms (constant within each dataset) to receive a liquid reward. Depending on the dataset, the terminus was either 1 of 8 targets spaced uniformly about a circle or 1 of 18 targets at the corners of two concentric hexagons. During the trial, only the origin, terminus, and cursor representing the subject's hand were visible; the remaining targets were hidden. The next trial then began with the origin appearing at the center of the workspace (center-out) or the previous terminus (point-to-point). Targets were block-randomized to evenly distribute trials to each target in pseudorandom order. In all data, trial initiation was self-paced. Monkeys were well-trained on the task before the datasets analyzed here were collected. Before decomposing submovements, we applied a Savitzky-Golay filter to remove high-frequency recording noise (5th order filter, 151ms window, similar to [43]) and decimated to 100 Hz. Dataset-specific parameters (duration, perturbation conditions, etc.) for the five datasets we examined in this work are listed in Table 4.1.

Datasets was not decomposed all at once. Instead, a thresholding procedure was applied to the tangential speed to segment the velocity profile. A movement segment started whenever tangential speed increased above a 0.005 m/s threshold, and ended as soon as the speed fell below the same threshold. The average segment length ranged from 0.7s to 1.1s over the five datasets.

Algorithm 1: Submovement decomposition pseudocode

input : $v(t)$, Δ
output: Submovement parameters Θ_N
$K = \#$ of time samples in $v(t)$, $N_{max} = \frac{K\Delta}{0.1 \text{ sec}}$
$D_{min} = 0.150$ sec., $D_{max} = 1$ sec., $t_0^{max} = K\Delta - D_{min}$
while *True* **do**
  for k *in* $\{1, \cdots 10\}$ **do**
    // Initialize submovement parameters
    for i *in* $\{1, \cdots, N\}$ **do**
      if $N > 1$ *and method is SPEC and* $i < N$ **:** Get $t_0^{(i)}$, $D^{(i)}$ from Θ_{N-1}
      elif *method is SPEC* **:** Sample $t_0^{(i)}$ and $D^{(i)}$ as described in Section II.C
      elif *method is SP* **:** $t_0^{(i)} \sim \text{uni}[0, t_0^{max}]$, $D^{(i)} \sim \text{uni}[D_{min}, D_{max}]$
      elif *method is Scattershot* **:**
        $t_0^{(i)} \sim \text{uni}[0, t_0^{max}]$, $D^{(i)} \sim \text{uni}[D_{min}, D_{max}]$,
        $A_x^{(i)} \sim \text{uni}[min(v_x), max(v_x)] \cdot D_{max}$, $A_y^{(i)} \sim \text{uni}[min(v_y), max(v_y)] \cdot D_{max}$
      $\theta_i = (t_0^{(i)}, D^{(i)})$
    //
    // Jointly optimize over all parameters $\{\theta_i, A_i\}_{i=1}^N$
    // using SQP/CMA-ES/SA
    if *method is Scattershot* **:**
    Solve (4.2.6) to find Θ_N^k
    elif *method is SP or method is SPEC* **:**
    Solve (4.2.2) to find Θ_N^k
  $\Theta_N = \min_k error(\Theta_N^k)$
  // Determine the relative MSE of the current best fit
  $error(\Theta_N) = \frac{\|V - F(\Theta_N)M(\Theta_N)\|_F^2}{\|V\|_F^2}$
  $\Delta MSE = error(\Theta_{N-1}) - error(\Theta_N)$
  // Check termination conditions
  if $error(\Theta_N) < 2\%$ *MSE or* $\Delta MSE < 0.1\%$ *MSE or* $N > N_{max}$ **:** break
  else: $N = N + 1$

4.3 Results

For each of the five datasets, we ran each algorithm 5 times in alternating order. Averaging performance statistics from multiple runs over the same data was necessary due to the

Table 4.1: Summary of datasets and experimental parameters.

Subj.	Data	Length	Task type	Target configuration
P	P1	843 s	Center-out	8 points on circle
J	J1	502 s	Point-to-point	18 corners of two hexagons
R	R1	611 s	Center-out	8 points on circle
P	P2	967 s	Center-out & curl force field	8 points on circle
J	J2	892 s	Center-out & mid-trial target jumps	8 points on circle

Data	Target radius	Workspace diameter	Hold times
P1	0.75 cm	10 cm	500ms
J1	1 cm	15 cm	0.5-1.5s
R1	0.75 cm	12 cm	400ms
P2	0.75 cm	10 cm	600ms
J2	1.2 cm	13 cm	700ms

Three monkeys (J, P and R) performed point-to-point reach-to-hold movements under a variety of experimental conditions, including different target configurations, dynamical perturbations (curl force field), and target perturbations (mid-flight target jumps).

stochastic nature of the algorithms. Software was implemented in MATLAB and executed on a personal computer (Intel Core i7 processor, 16 GB RAM). The results of runtime and mean squared error (MSE) are shown in Figure 4.1.

In all datasets, the runtime of SP was significantly lower than that of scattershot and the runtime of SPEC was significantly lower than that of SP (Kruskal-Wallis test, $p < 0.01$). Absolute runtime varied across datasets, depending on task structure and any perturbations applied during the task. Notably, in all datasets SPEC was able to complete faster than real-time, though in general this will depend on the speed of the computing platform. Depending on the dataset, scattershot took 2-4 times longer than SPEC to decompose the same movements. Figure 4.1 shows that there was a statistically significant difference in MSE between the three algorithms, with SPEC outperforming SP, which in turn outperformed scattershot. Furthermore, SP and SPEC used only slightly more submovements than scattershot to fit the same data: averaged over all datasets and decompositions, 0.1% more for SP and 3.5% more for SPEC. Using slightly more submovements, SP and SPEC were able to decompose the same movements for less MSE and in significantly less time than scattershot.

We assessed the relationship between our extracted submovements and other feature extraction methods previously proposed which did not rely on explicit function fitting. Figure 4.2 shows the average acceleration of the hand at the start and end of submovements in dataset P1 extracted using the SPEC method. Local optima in the acceleration correspond to zero-crossings in the movement jerk. We see clearly that the average submovement onset time is followed by a zero-crossing in the jerk and the average end of a submovement is preceded by another jerk zero-crossing. Other datasets had the same property. This is consistent

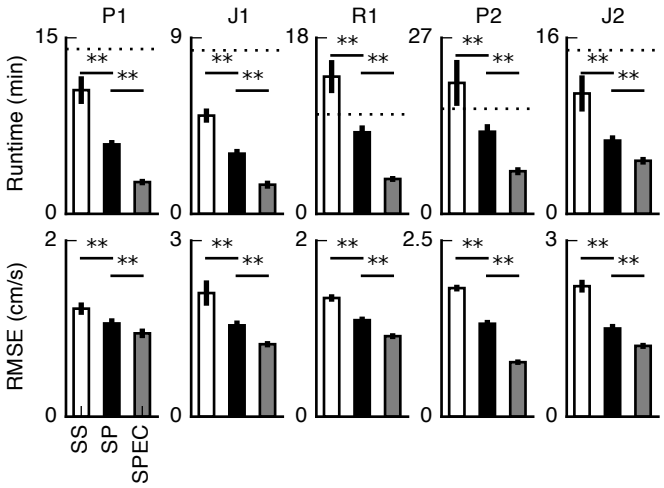

Figure 4.1: Runtime and quality of fit for decomposition algorithms. Submovement decomposition performance for all five datasets described in Section 4.2. The performance of the scattershot method [56] (SS) was compared to shape pursuit (SP), as well as shape pursuit combined with error concentration (SPEC). Performance was measured by mean and standard error of runtime and square-root MSE (RMSE) over 5 attempts. For runtime comparisons, the length of each dataset is marked with a horizontal dotted line. $^*p < 0.05$; $^{**}p < 0.01$; Kruskal-Wallis test.

with the submovement definition utilized by [52], which relies on finding inflection points based on local extrema of the acceleration or zero-crossings of the jerk. Similarly, we also observed that for the fitted submovements, the movement duration (D) scaled with tangential amplitude ($A_s = \sqrt{A_x^2 + A_y^2}$). Averaged over all datasets, the correlation between the fitted movement durations and the tangential amplitudes was significant ($r = 0.26$, $p < 10^{-8}$). This relationship was extracted without an explicit constraint, and similar properties have been observed in other studies which extracted submovements without fitting an explicit function prototype [62]. These two implicit properties of the submovement statistics extracted in our approach indicate that physiologically relevant parameters are being extracted.

We examined in detail one complex movement made by monkey J which was not included in the preceding performance analysis. Figure 4.3A-B shows a 7s circular movement during which the hand never comes to rest. A 7s movement is considerably longer than the typical movement duration of the five datasets in the preceding analysis (see Section 4.2). In various studies of reaching movements from patients with neurological disorders such as stroke, Parkinson's disease, multiple sclerosis and Alzheimer's disease, typical movements shown in those studies took 3-10s [44, 45, 50, 64]. Hence this 7s multi-peaked movement represents a clinically relevant duration. We fit 0-16 submovements to this movement, finding a relatively high number of submovements necessary due to the presence of many peaks in the speed profile, similar to what is observed in stroke patients. (In comparison, each movement segment in the five datasets previously analyzed required an average of only 2.3-3.4

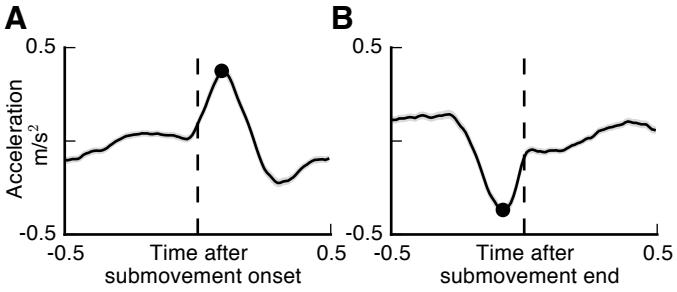

Figure 4.2: Average acceleration from dataset P1 at the *(A)* start and *(B)* end of submovements extracted using SPEC. The prominent peaks in the acceleration traces, marked by dots, correspond to zero-crossings in the movement jerk. These features resemble those extracted without optimization routines purely from higher derivatives of kinematic data (e.g., [52, 63]), indicating a relationship between our optimization-based method of submovement extraction and alternative methods that identify submovement beginning/end by time-series feature extraction.

submovements to obtain 2% MSE.) We fit this same movement 20 times to get dispersion estimates of percent MSE and runtime for scattershot, SP and SPEC. The MSE curve as a function of number of submovements was similar for all algorithms (Figure 4.3C) and all algorithms show exponential increases in runtime (Figure 4.3D) but the increase in runtime is visibly less steep for SP than for scattershot and less steep for SPEC than SP. Though the complexity of cost function evaluations is higher for SP and SPEC, and the complexity of evaluating the cost function grows with the number of submovements, Figure 4.3E shows that the number of cost function evaluations plateaus earlier for SP and SPEC than scattershot. SPEC remains subject to the same exponentially growing search space problems as scattershot but the point at which that begins to impact applications is "pushed back".

SP is an analytical method to exploit structure in the submovement decomposition optimization problem. We also explored the possibility that the same structure might be discoverable automatically through the use of optimization metaheuristics. For the same movement segment depicted in Figure 4.3, we again attempted to fit 16 submovements with either CMA-ES or SA (see Section 4.2). We limited CMA-ES to 500,000 cost function evaluations, which was roughly 100× the computation required for scattershot (2,500 cost function and gradient evaluations). On average, CMA-ES-SS achieved an average MSE of 14.6% versus CMA-ES-SP with an average MSE of 2.4%. This difference was statistically significant (Kruskal-Wallis test, $p < 10^{-10}$, 20 runs). In addition, CMA-ES-SS required an average of 168,000 function evaluations compared to 41,000 function evaluations for CMA-ES-SP. In the case of simulated annealing, SA-SS did not fare nearly as well, achieving 73% MSE compared to 9.2% MSE for SA-SP (Kruskal-Wallis test, $p < 10^{-10}$, 20 runs). Again, SA-SS required many more function evaluations than SA-SP: an average of 68,000 for SA-SS versus 34,000 for SA-SP. In both cases, the use of problem-specific structure aided the quality of solution found. That is, the advantages of SP over SS are not specific to gradient-based optimization solvers. Furthermore, given that the expense of calculating the gradient is the approximately

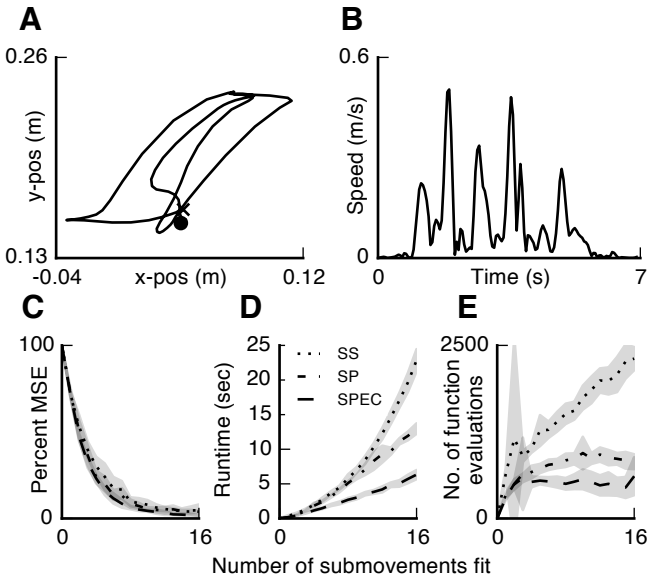

Figure 4.3: *(A)* Monkey J made a 7 sec. circular movement starting at the dot and ending at the "x". *(B)* The velocity had many peaks and thus required many submovements to capture. *(C)* The percent MSE of fitting an increasing number of submovements for scattershot (SS), SP and SPEC was similar for all algorithms. Error bars indicate standard deviation over 20 independent attempts. *(D)* For all algorithms, the mean runtime increases exponentially with the number of submovements being fit, but the steepness of the exponential increase was much steeper for scattershot than for SPEC. *(E)* Number of function evaluations for each of the three methods. The number of function evaluations plateaus earlier for SP and SPEC than for scattershot.

the same as evaluating the cost function (see Section 4.2), SQP is substantially more efficient than either metaheuristic.

4.4 Discussion

Optimization-based submovement decompositions allow for a parsimonious and quantitatively precise method of submovement decomposition, but can be computationally expensive. We have described methods for accelerating such decompositions. Further computational time savings might be acquired by performing computation on a more specialized hardware platform (e.g., graphical processing units). Though we have described this method in the context of the minimum-jerk submovement prototype function, the same fundamental partitioning of scale and shape parameters can be applied broadly to other typical submovement functions, e.g., Gaussian functions, log-normal functions, etc.

Iterative fitting algorithms, including the approach we use here, may suffer from overfitting if only goodness of fit criteria are applied, since increasing the number of parameters fit always provides the opportunity to decrease error. There are many standard penalty

methods that apply to these types of optimizations, in particular the Akaike information criterion and the Bayesian information criterion. This issue of methods to examine overfitting is deserving of further study. However, these standard methods describe termination conditions for when the number of submovements should stop being incremented, and do not affect the execution time differences at the core of this work.

The SP algorithm analytically finds submovement amplitudes after the shape parameters are fixed. This algorithm is very similar to some approaches in sparse coding (e.g., [65]). In the sparse coding problem, if the basis vectors are fixed, then the coefficients can also be found quickly (see [65] for a more detailed discussion). Though our approach exploits the same mathematical properties, there are a few differences. Here, we fit multi-dimensional data and enforce a consistency constraint between the different data dimensions (speed coefficients are required to be consistent with the Cartesian coordinate coefficients). More importantly, our dictionary is a continuous space of functions which we assume from the outset. An interesting area of future study would be to learn the feature vectors (i.e., the submovement shapes) using a sparse coding approach, which to our knowledge has not been formally performed. Unlike sparse coding approaches, the present approach strictly enforces sparsity constraints by limiting the number of submovements fit, effectively attempting to minimize a non-convex ℓ_0 norm. Interestingly, [66] reports that the procedure for optimizing the scale coefficients is slower than the procedure for optimizing the basis vectors. The reverse is true in the present study; further study of the differences between these two approaches may yield additional computational benefits.

Our methods enable submovement decompositions from optimization-based routines to be computed in real-time in many cases, which may possibly enable biofeedback applications. For instance, since stroke recovery is characterized by increased submovement blending [50], real-time biofeedback of submovement statistics during therapy sessions may have rehabilitation value. Though stroke rehabilitation in many ways resembles motor learning [67], the usefulness of this type of feedback during therapy remains unclear. For example, when visual feedback is used to augment natural feedback about postural stability and center of gravity, some studies indicate that the feedback has little therapeutic effect (e.g., [68]) while others report positive gain in the initial stages of therapy (e.g., [69]). The benefits of quantitative feedback of movement during rehabilitation require additional testing.

4.5 Conclusion

In this work, we developed two methodological innovations to accelerate the process of fitting natural movements to a sum of smooth submovements. Our innovations apply to a broad range of submovement functions explored in the literature and significantly reduce the time required to complete submovement decompositions. These innovations may accelerate analysis of submovements for basic neuroscience and enable real-time applications of submovement decomposition.

Chapter 5

Evidence of feedforward and model-based neuroprosthetic control

This chapter explores the hypothesis that BMI control involves the use of an internal model during motor adaptation and feedforward control. These results are preliminary in that the experiments have only been performed with one monkey.

5.1 Introduction

Model-based optimal control (e.g., [70]) is an appealing theory of motor control. A very general control policy $u_{t+1} = \pi(x_t, x_t^*)$ maps the current state x_t and the current target x_t^* to the next command u_{t+1}. However, in the highly complex sensorimotor system, it is difficult to assess whether skilled behavior is due to the use of an explicit "forward model" $x_{t+1} = f(x_t, u_t)$ and control policy calculations based on the goals at hand, or whether an "inverse model" lookup table stores all the next actions given the current state (i.e., $\pi(\cdot, \cdot)$ is computed directly). (See [71] for a review of relevant literature). One key line of evidence for model-based control comes from sensory feedback delay. Visual feedback delay may be more than 100ms and the long delay may lead to controller instabilities. An internal model used for forward prediction may be used to correct for such instabilities.

Another key line of evidence comes from experiments where a perturbation is applied to a reaching task, e.g., a curl force field [72]. When such perturbations are applied, subjects gradually improve their performance, suggesting that they use error measures to gradually adapt their internal models. Furthermore, when the perturbation is removed, "after-effects" are observed in which the subjects generate trajectories more consistent with the adapted internal model than the baseline model. A plethora of such experiments have demonstrated key findings about motor learning. Yet without finding the cortical substrate of internal models, the forward model and inverse model ambiguity remains.

BMI may be used as a tool for investigating such questions. The natural motor system is highly complex, with millions of motor cortical neurons projecting to the spinal cord; the

relationship between the firing of any single neuron and the muscular output cannot easily be found. The BMI circuit is considerably simpler than the natural motor circuit since the system is linear and the output neural layer is directly specified by the experimenter. In the BMI paradigm, it has been shown that subjects adapt adapt to a visuomotor rotation in a manner consistent with natural reaching studies under visuomotor rotation [19, 34, 20]. Methods have been developed to extract putative internal model parameters during BMI control [41].

There are practical implications for neuroprostheics in elucidating the presence (or absence) of internal models. If an internal model can be extracted, either by inference or by directed observations of neural activity in other brain regions, then the decoder structure may be changed to match the internal model to achieve more naturalistic control. This also may yield more efficient CLDA methods as well if the assumptions regarding intended kinematics are made using more accurate knowledge of the actual control policy.

In this chapter, we review data indicating that closed-loop BMI control is model-based. Some suggestion is made to this theory in Chapter 3, as model mismatch between the decoder and the subject may be the reason why unpredictable changes in the PVKF dynamical structure yield inferior performance. Data from variants and manipulations of center-out control are presented. In addition, we show evidence that as one subject improved task performance, control became more feedforward, indicating a better understanding of the decoder was internalized.

5.2 Methods

Behavioral tasks

Results in this chapter used variants of the center-out task described previously in Section 2.2. In certain experiments, the following variants of the center-out task were used:

1. The cursor was reset to the center at the end of each trial.

2. There was no explicit hold time requirement. Instead, reward at the end of each trial was proportional to the total time spent in the target. If a maximum hold time was reached, the trial ended in reward, capping the total possible reward per trial. This incentivized the subjects to hold in place for as long as possible.

3. On a randomly chosen subset of trials, the target jumped 8cm further away 0.5s after the go cue.

Each modification to the standard center-out task was used separately. Results presented in this chapter involve only Monkey G; surgical details were described previously in Chapter 2

KF motor commands

The "process noise" term in the KF model can be interpreted as a "motor command", by simple analogy.

$$x_{t+1} = Ax_t + w_t \qquad \text{[KF model]}$$
$$x_{t+1} = Ax_t + Bu_t \qquad \text{[Standard linear dynamical system]}$$

One interpretation of the KF then is that the process noise w_t plays the role of an unknown command input and the KF is forming an estimate of w_t. In this interpretation, A must be a perfect model. In the virtual BMI setting, this is a reasonable restriction as A may be entirely set by the experimenters to represent some system that the subject is tasked with controlling.

Notably, the KF's prior model of the motor command input is an i.i.d Gaussian model, a minimally informative model. Hence, the burden to generate appropriate w_t rests primarily on the subject. In reality of course it does not make sense for w_t to truly be IID Gaussian. For instance, when controlling a cursor and moving it along a smooth trajectory, velocities are likely to be correlated across time (e.g., if moving in a straight line, then the direction of movement commanded may be exactly the same for the entire trajectory, rather than drawn from a uniform distribution as assumed by the random walk model.

We therefore define the BMI command coherence as

$$c_t = \frac{\pi - \angle\left(w_{t-1}^{vel}, w_t^{vel}\right)}{\pi} \tag{5.2.1}$$

This measure results in a value between 0 and 1.

Error clamp trials

In studies of natural motor adaptation, a common technique is to introduce a perturbation in the reaching movement, allow multi-trial learning of the perturbation and then subsequently remove the perturbation and observe "after-effects" of the adaptation [72]. *Error clamp* trials can be a powerful tool for measuring motor action after a perturbation has been learned [73]. By creating a channel to constrain arm movements, any systematic error in motor action must be due to feedforward error. Concretely, suppose that we have the baseline KF cursor decoder structure

$$\begin{bmatrix} p_{t+1} \\ v_{t+1} \\ 1 \end{bmatrix} = \begin{bmatrix} I & 0.07I & \bar{p} \\ 0 & 0.5I & \bar{v} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_t \\ v_t \\ 1 \end{bmatrix} + \begin{bmatrix} \alpha K \\ K \\ 0 \end{bmatrix} y_t \tag{5.2.2}$$

where 0.07 and 0.5 are values selected arbitrarily based on the range of spontaneously generated values and which we confirmed experimentally the subject was able to control. As

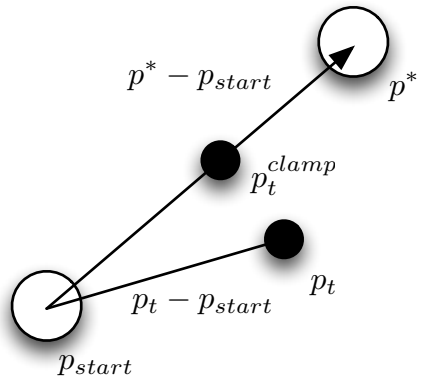

Figure 5.1: Error clamp projection for a kinematic cursor.

discussed at length in other chapters, Ky_t encapsulates the neural signal component and acts as a control signal to a linear system. Hence, to generate an equivalent to the visuomotor rotation commonly used in psychophysics. we can rotate the control signal Ky_t:

$$\begin{bmatrix} p_{t+1} \\ v_{t+1} \\ 1 \end{bmatrix} = \begin{bmatrix} I & 0.07I & \bar{p} \\ 0 & 0.5I & \bar{v} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_t \\ v_t \\ 1 \end{bmatrix} + \begin{bmatrix} R(\theta) & & \\ & R(\theta) & \\ & & 1 \end{bmatrix} \begin{bmatrix} \alpha K \\ K \\ 0 \end{bmatrix} y_t, \tag{5.2.3}$$

in which $R(\theta)$ is a 2-D counter clockwise rotation matrix:

$$R(\theta) = \begin{bmatrix} cos(\theta) & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}.$$

Alternatively, to manipulate the decoder to induce an equivalent of the curl field perturbation, we can make the angle of rotation applied to the control signal dependent on the cursor's speed rather than a fixed constant θ:

$$\begin{bmatrix} p_{t+1} \\ v_{t+1} \\ 1 \end{bmatrix} = \begin{bmatrix} I & 0.07I & \bar{p} \\ 0 & 0.5I & \bar{v} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_t \\ v_t \\ 1 \end{bmatrix} + \begin{bmatrix} R(b \cdot ||v_t||) & & \\ & R(b \cdot ||v_t||) & \\ & & 1 \end{bmatrix} \begin{bmatrix} \alpha K \\ K \\ 0 \end{bmatrix} y_t. \tag{5.2.4}$$

During error clamp trials, the position of the cursor is confined to the straight line between the origin and the target. To implemented this for natural arm movements, a "convergent" force field is applied to the endpoint to keep it from deviating from the specified path. For a cursor BMI, we project the position state onto the straight line between origin and target, as shown in Figure 5.1.

In systems with virtual or physical dynamics, methods similar to those used in natural motor studies can be used. These methods above are needed in the case where the system is massless and analogies must be used between forces and the actual parameters of the system.

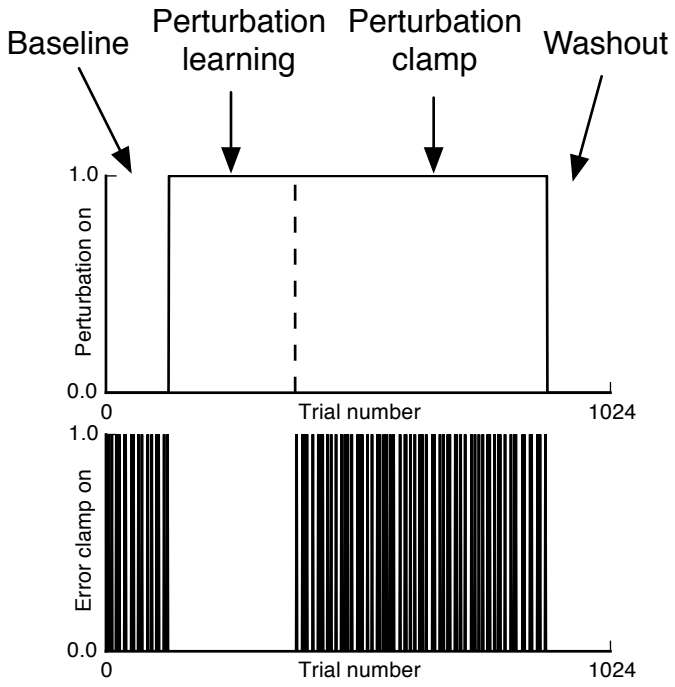

Figure 5.2: Error clamp session trial structure. An error clamp session had four epochs: baseline, perturbation learning, perturbation clamp and washout. During the baseline period, subjects operated the CLDA-trained cursor decoder as normal. Error clamp trials were interspersed to measure baseline feedforward behavior. Then the perturbation was turned on without cue to the subject. The subject was given time to learn and correct for the perturbation. After a set number of trials after which the subject had presumably learned to compensate for the perturbation, error clamps were interspersed to measure feedforward behavior after learning the perturbation. A washout period followed to examine after-effects.

For error clamp sessions, we used the center out task variant where the cursor was reset to the center at the end of each trial. This resetting helped to mitigate contamination from the end of one trial to the next trial by ensuring that the cursor always started in the same position. The trial structure is depicted in Figure. The center-out task had 8 targets and targets were block-randomized, i.e. after one block of 8 trials, one successful trial to each target was completed. During error clamp epochs, one out of 8 trials in the block was selected as the clamp trial randomly. The clamp and perturbation structure is depicted in Figure 5.2.

5.3 Results

Modulation of control signal coherence across time

Difference between hold versus target states

Holding a cursor in place during BMI control is known to be a challenging task [3, 24, 11]. We asked whether the control signals between the movement phase and the hold phase of the standard center-out tasks came from different distributions. Recall from Chapter 3 that the KF system equation can be described as

$$x_{t+1} = Fx_t + Ky_t,$$

in which the term Ky_t encapsulates the neural command velocity. In our 2-D position-velocity state space, the position control terms are linearly related to the velocity control terms; hence we only consider the velocity component, $K_{vel}y_t$, as made explicit by (5.2.2). Figure 5.1a shows histograms of the control signal magnitude $||K_{vel}y_t||$ during the movement epoch and the hold epoch. As expected, the typical control signal magnitude during the hold epoch is less than during the movement epoch. Figure 5.1b shows that the instantaneous displacement of the cursor, $||p_t - p_{t-1}||$ is also less during the hold epoch than in the movement epoch, as is implied by the ability to complete the task quickly. These differences in distributions imply a shift in the control policy during different phases of the task. We asked whether the shift was due to a gain reduction in the generated control signal, or whether the policy changed in some state-dependent manner. We measured the efficiency of each control signal by the ratio of displacement to control signal size: $\frac{||p_t - p_{t-1}||}{||Ky_t||}$. Figure 5.1c shows that the efficiency is slightly less during the hold epoch than in the movement epoch.

Similar results were found for the variant of the center-out task where Monkey G was incentivised to hold in place for as long as possible (see Section 5.2). Figure 5.2a shows a difference in distributions of control signal size during the hold and movement epochs and Figure 5.2b shows a difference in the size of the instantaneous cursor displacement during the two epochs. Similar to the short timed hold paradigm, Figure 5.2c shows that the displacement per unit control signal was less during the hold epoch. These results indicate that the subjects generated smaller signals as well as signals tailored to the current state to generate less endpoint displacement.

Response to target jump perturbation

Figure 5.3 shows the response to the target jump perturbation described in Section 5.2. Monkey G performed the target jump task using two different types of decoders, the Kalman filter (KF) and the point-process filter (PPF). The PPF enables high-performance BMI with decoding at much faster rate than the KF (180 Hz compared to 10 Hz, in this study), enabling study of the BMI control policy at a much finer time scale. The top row of Figure 5.3 shows the average cursor speed trajectories for jump trials and regular trials. Predictably, the

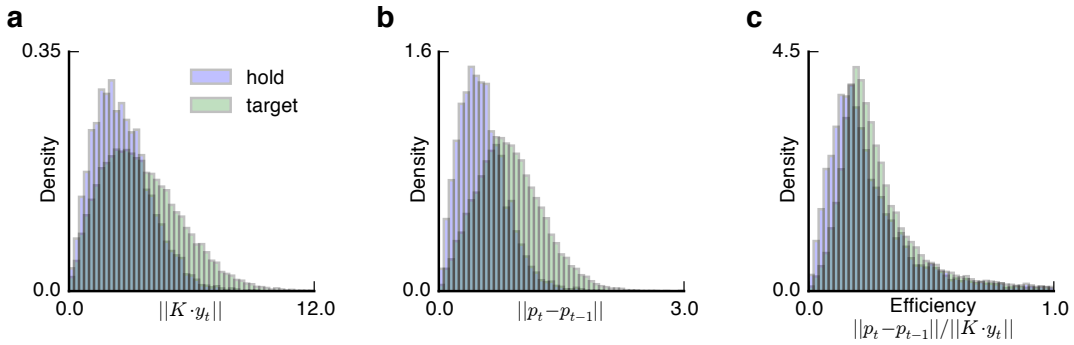

Figure 5.1: Kinematics of BMI cursor reaching versus holding. (a) Histograms of the instantaneous neural command magnitude during the hold phase (blue) and the reaching phase (green). The distributions are overlapping but visually it is clear that the distributions are different. (b) Histograms of the instantaneous displacement of the cursor during the hold phase and the movement phase. The movement phase typically has larger displacements. (c) Efficiency of the control signal in displacing the cursor, calculated as the ratio of the displacement to the size of the control signal. Efficiency was also reduced during the hold epoch.

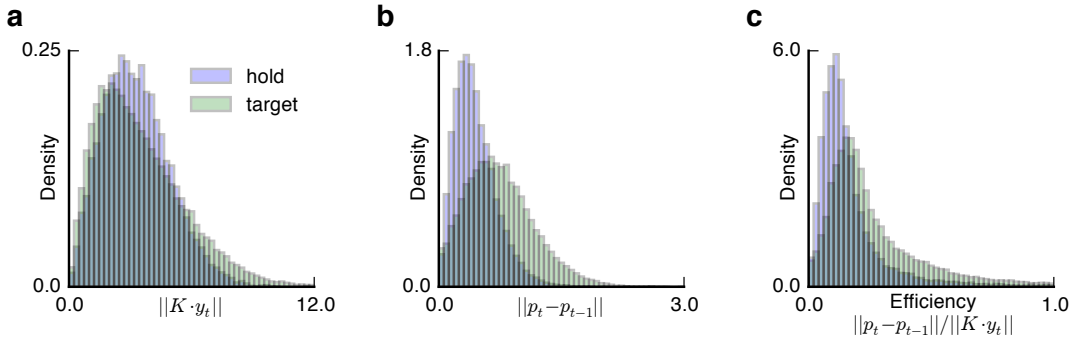

Figure 5.2: Similar to Figure 5.1 on a center-out task variant where the subject was incentivized to hold in place for as long as possible. The reward was set proportional to the time the cursor spent in the target. The command magnitudes between the reach and hold phases are more overlapping than in the timed case.

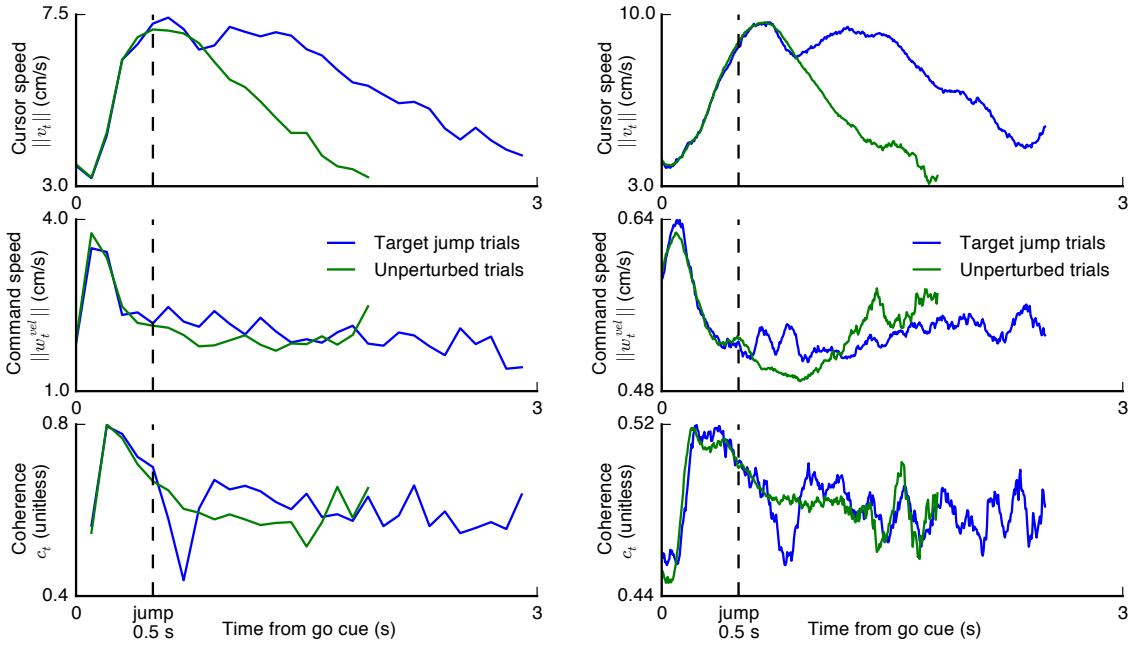

Figure 5.3: Cursor speed and control angle for monkey G during a target jump BMI task. Following the target perturbation, the control angle increases as the cursor slows down and decreases as it accelerates, demonstrating temporal modulation structure in the control policy under "surprise" conditions. The left shows the task performed with a Kalman filter with decoded state updated at 10 Hz. The right shows the same task performed with a point-process filter (PPF) with the decoded state updated at 180 Hz. For the PPF data, coherence is smoothed with a 10 Hz moving average.

cursor speed shows a "submovement" following the target jump during the jump trials but not during normal trials. The command speed (middle row) is similar for both types of trials, though slightly higher following the target jump. The coherence (bottom row) shows a drop and than an increase following the target jump, indicating a response in which the subject decelerates the cursor and then accelerates by altering how commands combine across time.

Internal model adaptation revealed using error clamps

The error clamp paradigm provides a method to measure feedforward control commands by eliminating task errors. Figure 5.4 shows the direction of the control signal as a function of the current speed of the cursor. In this analysis, trajectories of the cursor and the control signal have been rotated so that all trials appear as if they are to the center-out target at 0 degrees. Control signals are concentrated in the direction of the target, not surprisingly. As the cursor is moving more quickly, the control direction becomes more concentrated in the direction of the target.

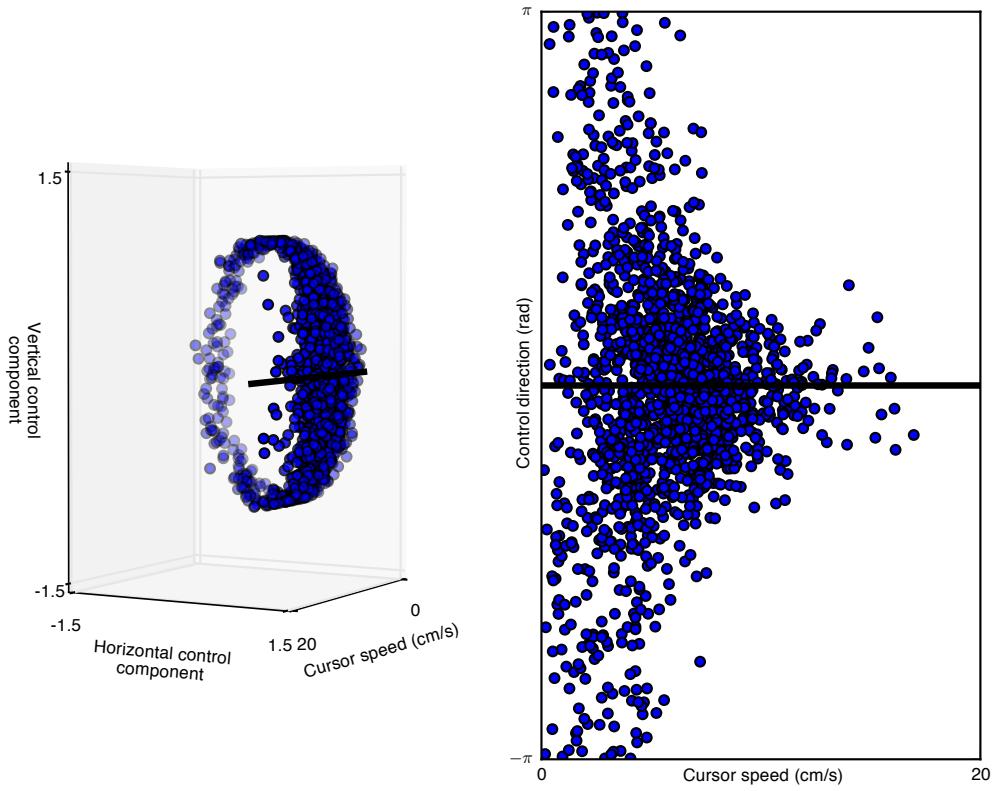

Figure 5.4: Direction of control signal during baseline clamp trials. Both plots show the same data in different perspectives. The left plot shows the horizontal/vertical control direction against the cursor speed state. When the cursor speed is fast, the control directions tend to cluster in the direction of the target. The right plot shows the same trend with the cursor speed on the x-axis and the control direction in radians (rather than a 2-D vector). Again, the control direction is centered around 0, and as the cursor speed is faster, the control direction is clustered closer to 0.

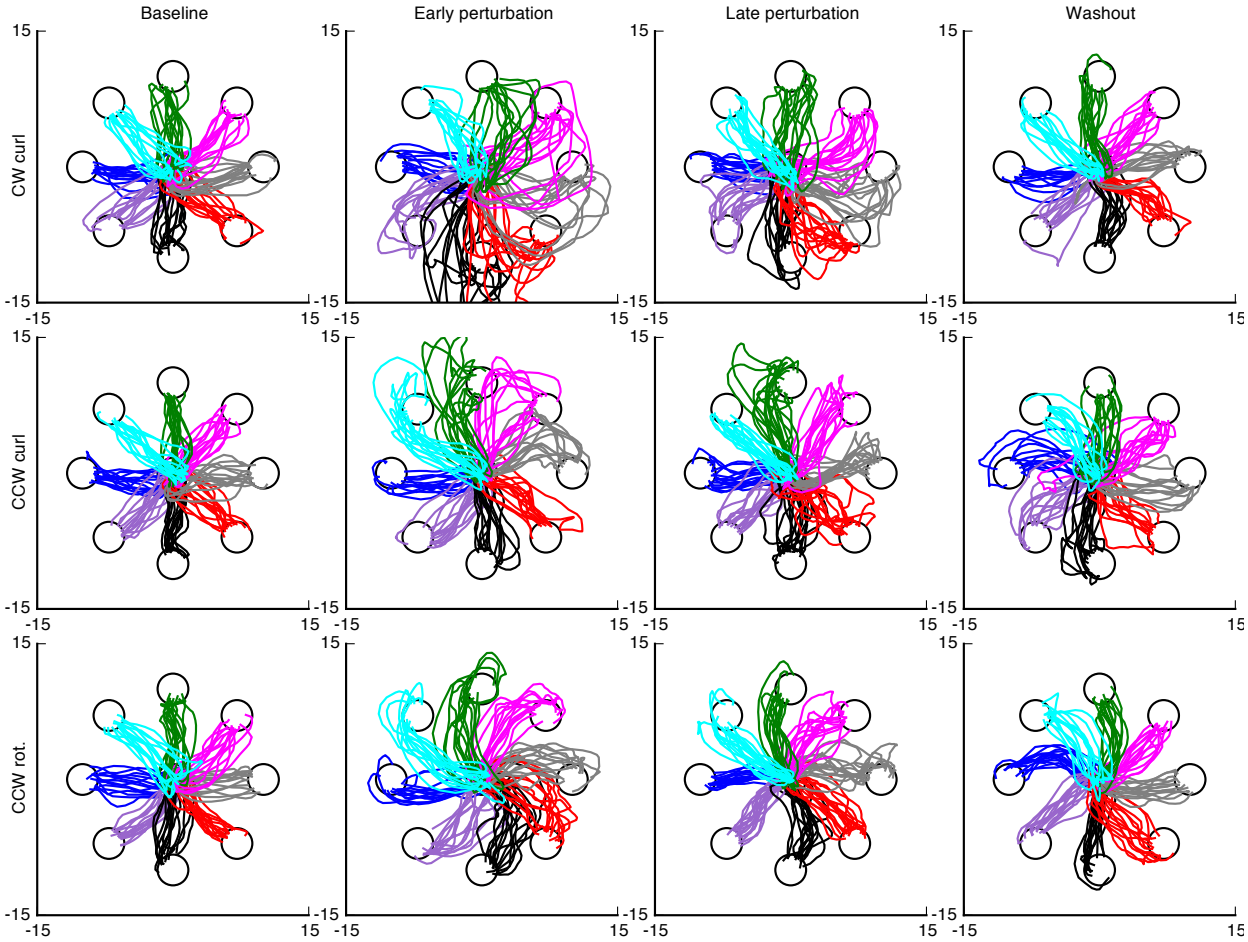

Figure 5.5: Trajectories during baseline, early perturbation, late perturbation and washout during various different perturbations (depicted in different rows).

The monkey improved the straightness of reach trajectories after being exposed to the various perturbations. Figure 5.5 shows the trajectories during various phases of the session: baseline, early perturbation, late perturbation and washout. Trajectories are relatively straight during the baseline period, are perturbed by the sudden onset of the perturbation, and gradually straighten out with practice as shown in the late perturbation trajectories (though they do not reach the straightness of the baseline period).

Analysis of the control signals during the error clamp trials after the perturbation has been learned is shown in Figure 5.6. Recall that the rotation applied to the control signal is either constant or proportional to the current speed of the cursor. Thick black lines indicate the optimal control direction in order to cancel out the rotation to be applied by our perturbation. Figure 5.6 shows that the control direction during the clamp trials follows the direction which would optimally cancel out the perturbation.

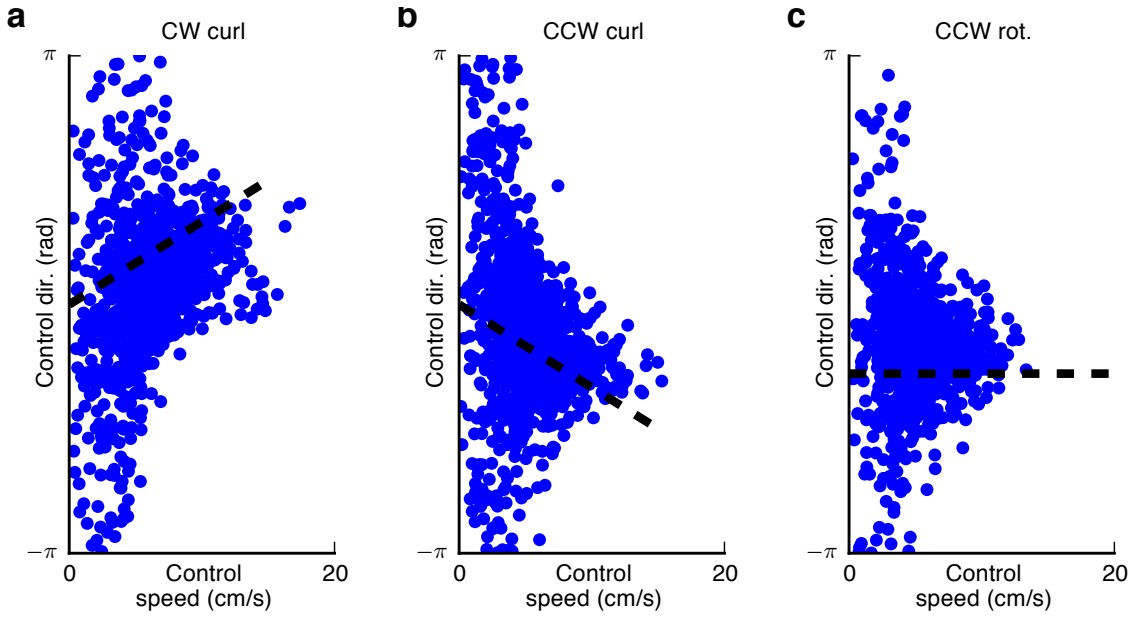

Figure 5.6: Control signal direction as a function of the cursor's velocity during error clamp trials for (a) CW curl perturbations, (b) CCW curl perturbations and (c) CCW rotation. Dashed black lines indicate the optimal command to move toward the target while canceling out the perturbation.

Movements became more "blended" with learning

In this experiment, the KF intended state space model was altered so that the position of the cursor evolved with simulated dynamics with the decoder outputting piecewise constant force in a 2-D plane:

$$\tilde{x}_t = \begin{bmatrix} F_x(t) \\ F_y(t) \\ 1 \end{bmatrix}.$$

Endpoint force acting on a 1 kg point mass was decoded at 10 Hz, a typical rate for the KF. Simulated physics were updated at 60 Hz to match the refresh rate of the screen. Piecewise constant acceleration was found using the relationship $F = ma$ and then position and velocity were found by integration of the piecewise constant acceleration. No explicit limitation was placed on how much the endpoint force was permitted to change from one time point to the next. To apply CLDA, intended kinematics were found using an experimenter-specified PD controller.

The decoder parameters were held nearly constant over the course of the experiment, with most days using the exact same parameters. Waveforms were similar though not exact across days, though the decoder parameters were not altered (e.g., using CLDA) to compensate for waveform instability. Figure 5.7a shows the movement times achieved by Monkey G over

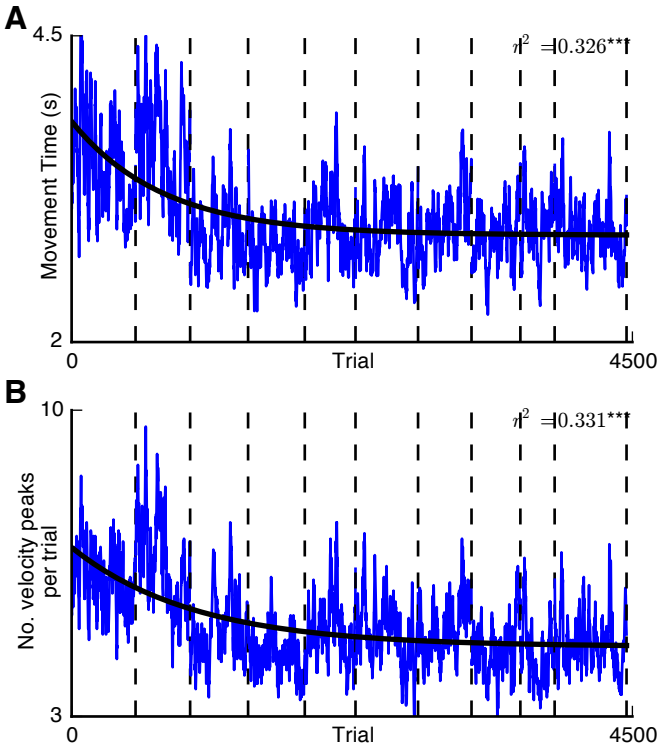

Figure 5.7: Monkey G learned to control a virtual point mass cursor with mass of 1 kg. Neural control was executed directly over endpoint forces acting on the mass. Over the course of several days (vertical dashed lines indicate day boundaries) and thousands of trials, the number of velocity peaks in the trajectory decreased, indicating more "blending" of submovement components and suggesting more accurate feedforward control of the prosthetic device.

the course of several thousand trials split over 10 days. The movement times improved over the first three days and then stabilized for the remaining days.

In natural movements from healthy individuals, movements are often comprised of smooth, bell-shaped velocity profiles (e.g., [53]). During recovery after stroke, movements initially have many submovements which appear to "blend" together as recovery progresses [50]. We asked whether similar trends might occur in BMI skill acquisition. 5.7b shows the number of speed peaks (local maxima in the speed profile) for each trial. As the movements became faster, the number of speed peaks per movement decreased, suggesting that movements became more feedforward as skill improved.

5.4 Discussion

The results here indicate that BMI control may be model based and when control is consolidated with learning may become more feedforward. These results of course remain preliminary as they involve only one monkey subject. Nevertheless, they are consistent with studies of natural motor learning.

During center-out control, we discovered that there was a difference in the subject's control policy between movement and hold task epochs. An optimal controller need not have a different policy between these epochs. For instance, an infinite horizon LQR controller could simply always move toward the center of the target and be able to use the same controller during both epochs to successfully complete the task. The shift in behavior we observed during BMI control may indicate a shift to enable the subject to more successfully deal with control noise (as BMI control is quite noisy) or may require less energetic cost. Further experiments are necessary to distinguish the various possible hypotheses.

Error clamp trials revealed that BMI control signals after learning a perturbation tended to optimally cancel the perturbation. Monkey G learned to compensate for a velocity-dependent perturbation. This suggests that Monkey G adapted to the perturbation and adjusted the BMI control policy to have a variable compensation for the perturbation depending on the speed state of the cursor. This required the subject to know the speed state of the cursor, possibly by learning a model of the state-dependent perturbation.

As Monkey G learned to control the point-mass system, the number of peaks in the speed profile decreased and the trajectory became less segmented. In natural motor studies, similar behavior after stroke recovery has been implicated as evidence of more feedforward control. Feedforward control may involve a forward model for prediction of where various sequences of control signals will drive the system.

In all these cases, degenerate instances of inverse models may also be able to play the same role as a forward model. The shift between movement and hold control signal generation can be accomplished by a switch between two different inverse models. The learning of sudden perturbations may be accomplished by rapidly reprogramming the entire inverse model to deal with the perturbation. And the appearance of more feedforward control may be accomplished by an internal model which expands to generate commands for multiple timesteps instead of a single timestamp. It may always be possible to construct an inverse model which emulates forward model behavior. These issues remain to be resolved by the field at large.

Chapter 6

Neural control of a redundant kinematic chain

This chapter describes an experimental demonstration of BMI control over a redundant virtual robot. The work was in collaboration with Helene Moorman (whose contribution was equal to mine) and Jose Carmena. At the time of this writing, a version of this work is in review for publication.

6.1 Overview

A Kalman filter was used to decode the joint velocities of a virtual four-link kinematic chain from single- and multi-unit neural activity in two non-human primate subjects. The subjects completed movements of the chain's endpoint to instructed target locations within a two-dimensional plane. This system was kinematically redundant for an endpoint movement task, as four DOFs were used to manipulate the 2-D endpoint position. This redundancy allows null movements in which the individual links of the chain move in a way that cancels out and does not result in endpoint movement. Both subjects successfully performed the task and improved with practice by producing faster endpoint velocity control signals. As the subjects became more proficient at controlling the chain, the amount of null movement also increased. Performance suffered when only the endpoint of the chain was visible but the links of the kinematic chain were hidden. Furthermore, all available degrees of freedom received task-relevant control inputs. All four DOFs of the joint-velocity control space exhibited significant task-relevant modulation. The relative fraction of variance explained by each DOF depended on the configuration of the chain, and trials in which the less-prominent DOFs were utilized also tended to have better task performance. Overall, these results indicate that the subjects incorporated the redundant components of the control space into their control strategy. Future BMI systems with kinematic redundancy, such as exoskeletal systems or anthropomorphic robotic arms, may benefit from allowing neural control over redundant configuration dimensions in addition to end-effector control.

6.2 Introduction

The natural motor system displays impressive dexterity in controlling the many degrees of freedom (DOFs) of arm movements. The arm contains kinematic redundancy in the transformation from shoulder, elbow, and wrist angles to hand position in 3-D space as there are many possible joint postures to achieve a single endpoint position. However, the neural mechanisms that enable dexterous control of redundant plants remain an area of debate. It remains unclear whether motor cortical neurons represent kinematic or kinetic parameters, or whether they represent intrinsic (e.g., joint) coordinates or extrinsic (e.g., hand) coordinates (see [74] for an overview). Furthermore, the point in the sensorimotor loop when kinematic redundancy is resolved remains unclear. Early studies indicated that the brain attempts to resolve redundancy in the planning stages by minimizing a cost function such as kinematic jerk [53], torque change[75], endpoint variance [76], etc. More recent studies have shown that variability in redundant dimensions may be greater than in "controlled" dimensions [77], possibly because movements in the redundant dimensions do not affect task performance and therefore controlling variability in those dimensions is unnecessary [70]. In addition, the ability of the natural motor system to control redundant systems transfers to novel mappings. Human subjects can learn a novel redundant mapping between naturally generated high-dimensional finger movements and a 2-D cursor [78, 79, 80]. Neural control of redundant manipulators is an important open question in motor control.

Brain-machine interface (BMI) systems hold tremendous potential to restore motor function lost to neurological injury or disease. Impressive proofs of concept have been shown for rodents, non-human primates, and humans controlling a variety of virtual and physical actuators [81]. The focus of this work is on kinematically redundant BMIs, where the combination of task and plant under neural control allow for multiple, equally optimal, task solutions. Typical BMI paradigms are not kinematically redundant. An example is the 2-D cursor movement task where the two degrees of freedom (DOFs) defining cursor position are controlled by neural activity [1, 2, 3, 4, 12, 5, 8, 9, 11]. In the cursor paradigm, there is only one way for the prosthesis to achieve any specific cursor trajectory. In contrast, the paradigm we use in this work has two added DOFs that allow many different manipulator trajectories to produce the same endpoint trajectory.

Even when greater numbers of controllable DOFs exist in a BMI system, the corresponding tasks are generally not kinematically redundant. For example, several studies have demonstrated control of arm-like robots that are able to grasp objects in 3-D space, but these paradigms lacked kinematic redundancy since neural control was only allowed for the end effector, e.g., endpoint velocity and 1-D grip [6, 13]. Any remaining mechanical DOFs remained under machine control and were not subject to neural input. Redundant kinematic solutions may have been possible in studies which supplemented 3-D velocity and 1-D grip with 3-D end effector orientation [**?**] and multiple grasp dimensions [82], and were possible when functional electrical stimulation was used to drive different muscles with similar effects on hand grasp [10]. The presence of redundant solutions in those scenarios would depend on the objects being grasped, and redundant solutions were not the focus of those studies.

To our knowledge, no studies have explicitly investigated control of a BMI system that exhibits kinematic redundancy. In order to examine redundant control in BMI, we designed a virtual four-link kinematic chain that moved in a two-dimensional plane. With this system, any endpoint position could be achieved by an entire 2-D space of kinematic chain configurations. Two non-human primate subjects were trained to control the four joint velocities of the chain with neural activity in a closed-loop BMI system. Single- and multi-unit activity was decoded using a Kalman filter and the parameters of the Kalman filter were calibrated daily using closed-loop decoder adaptation (CLDA). The subjects were tasked with moving the endpoint of the redundant kinematic chain to instructed target locations. Hence, the subjects controlled more DOFs than were necessary to perform the task.

6.3 Methods

Surgery and Electrophysiology

Two adult male rhesus macaques (Macaca mulatta), Monkeys C and G, were used in this study. Monkey C was implanted with bilateral chronic Teflon-coated 128-channel tungsten microwire electrode arrays (35 micron diameter, 500 micron wire spacing; Innovative Neurophysiology, Durham, NC) for neural recording. Monkey G was implanted with a similar array in the left hemisphere but was implanted with two 64-channel arrays (same electrode diameter and spacing) in the right hemisphere. The arrays targeted the hand and arm representation areas of primary motor cortex and dorsal premotor cortex based on stereotactic coordinates. Single and multi-unit activity was recorded using a 256-channel Omniplex system and sorted online using PlexControl (Plexon, Inc., Dallas, TX). All procedures were conducted in compliance with the National Institute of Health Guide for Care and Use of Laboratory Animals and were approved by the University of California, Berkeley Institutional Animal Care and Use Committee.

Behavioral tasks

The monkeys were trained on a virtual task in which they were required to move a cursor to an instructed target on a computer screen. After being trained to control the cursor with arm movements, the monkeys controlled the cursor with neural activity via a Kalman filter, which did not require overt arm movements. Once the monkeys had been familiarized with the brain-control cursor, they began controlling the joints of a four-link virtual kinematic chain with neural activity. Figure 6.1a illustrates the experimental setup. All tasks required moving the chain endpoint inside an instructed target (radius 1.3 cm for Monkey C, radius 1.5cm for Monkey G). They were given 7-15 seconds to complete each movement and needed to hold the endpoint inside the target for 200ms to receive a liquid reward. Failure to complete the movement in time or hold for sufficient time restarted the same trial without reward.

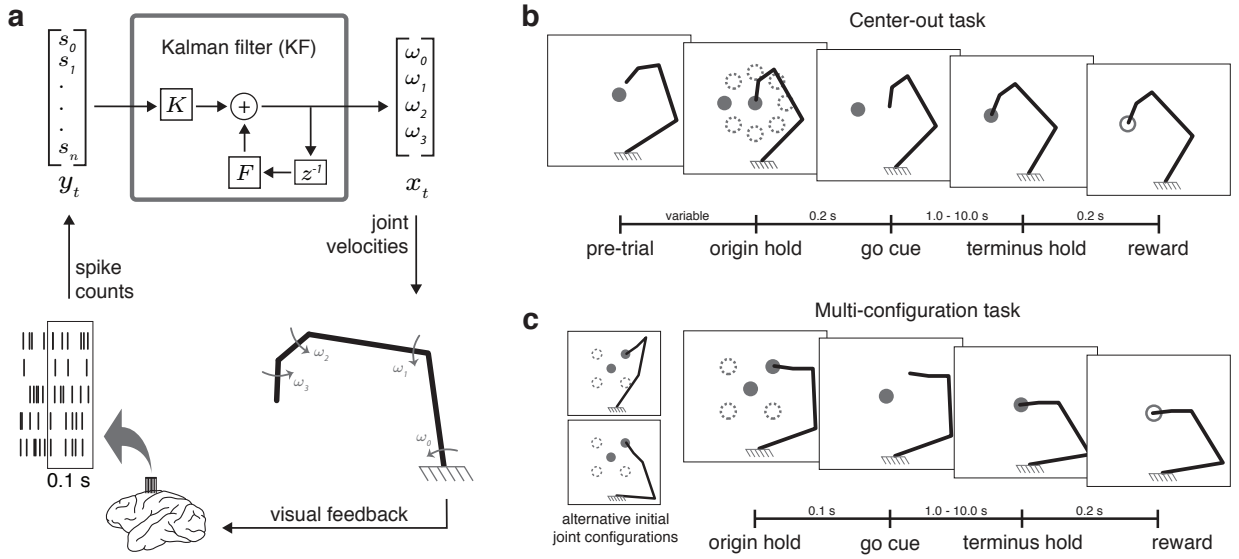

Figure 6.1: Experimental setup and task timelines. (a) Closed-loop BMI control of the kinematic chain. Observations of single and multi-unit spike counts were used to decode the joint velocities of a four-link kinematic chain moving in a 2-D plane. (b) Center-out task and timing. The subjects were required to move the chain's endpoint to the center to initiate trials and then move to a peripheral target. (c) Multi-configuration task and timing. Subjects were required to move the chain's endpoint to the specified target, starting from one of 3 possible joint configurations per starting position for Monkey G, or 5 possible joint configurations per starting position for Monkey C.

In the center-out task, illustrated in Figure 6.1b, the monkeys initiated trials at their own pace by holding the endpoint in the center for 200ms. Upon entering the center, the peripheral target appeared. After the center-hold period ended, the center disappeared, cuing the monkeys to initiate movement to the instructed target. Instructed targets (16 for Monkey C, 8 for Monkey G) were uniformly spaced about an 8 cm radius circle.

After the monkeys were familiar with the kinematic chain center-out task, they performed the multi-configuration task, illustrated in Figure 6.1c. At the start of each trial, the endpoint appeared at the center of one of four uniformly spaced peripheral targets. For each starting position, we set θ_4=-9 degrees and θ_3 to one of {135, 157.5, 180, 202.5, 225} degrees for Monkey C or {135, 180, 225} degrees for Monkey G, in absolute coordinates. The remaining joint angles θ_1 and θ_2 are fixed if the endpoint is known [83]. This yielded a total of 20 and 12 possible starting postures for Monkeys C and G, respectively. In all trials, the monkeys were instructed to move the endpoint to a target at the center of the workspace. Unlike the center-out task, this task was not continuous since there was a large, automatic movement of the chain back to one of the preset configurations before each new trial.

In some sessions, we tested the effect of visual feedback of the chain configuration on multi-configuration task performance. On 50% of trials randomly chosen, the entire chain

was hidden except for a cursor representing the endpoint. To determine whether the feedback reduction significantly impacted movement time, we used a randomization test. Since task performance varied by starting configuration, randomization was only performed within starting configuration.

For a small number of sessions that were analyzed separately, Monkey G performed a variant of the multi-configuration task with different starting configurations and instructed target locations. In each of these sessions, the starting configurations were selected such that the primary two principal components of the control signals (see Section 6.3 below) only allowed the endpoint to move in a single direction. That is, near these configurations, two DOFs in joint space only translated to one DOF in the endpoint space. In the robotics literature, such configurations are referred to as singularities. For each configuration, we placed the instructed target along the path of one of the remaining two principal components. This placed the target approximately perpendicular to the single path accessible to the primary principal components. An example is shown in Figure 6.3b. These combinations of starting configuration and instructed target were intended to elicit activity in the less prominent principal components.

Manipulator kinematics

Our subjects controlled the endpoint of a four-link virtual kinematic chain. The links of this chain, from proximal to distal, were 15cm, 15cm, 5cm and 5cm (selected arbitrarily). The four joint angles θ_1, θ_2, θ_3, θ_4 collectively specify the configuration of the chain. The endpoint position of the chain can be determined from the configuration by standard forward kinematics:

$$\begin{bmatrix} p_x \\ p_y \end{bmatrix} = f(\theta) = \begin{bmatrix} \sum_{l=1}^{N} l_i \cos\left(\sum_{k=1}^{i} \theta_k\right) \\ \sum_{l=1}^{N} l_i \sin\left(\sum_{k=1}^{i} \theta_k\right) \end{bmatrix},$$

in which $f(\cdot)$ is the forward kinematics function [84]. For sufficiently small velocities ω and small time-scales Δ, the forward kinematics can be accurately linearized:

$$f(\theta_t + \Delta\omega_t) \approx f(\theta_t) + \Delta \frac{\partial}{\partial\theta} f(\theta)|_{\theta=\theta_t} \cdot \omega_t = f(\theta_t) + \Delta J(\theta_t)\,\omega_t.$$

The manipulator Jacobian $J(\theta_t)$ is a 2×4 matrix mapping joint velocities to endpoint velocities. Thus it is possible to have joint velocities ω_{null} such that $J(\theta)\,\omega_{null} = 0$. These null velocities reconfigure the chain with little to no endpoint movement. In practice, such null velocities will induce a small endpoint movement due to linearization inaccuracies. For the maximum joint velocity we observed (0.6 rad/sec), this inaccuracy was less than 2 mm. In contrast, other velocities $\omega_{endpt} = J^{\dagger}(\theta)J(\theta)\omega_{endpt}$ in which $(\cdot)^{\dagger}$ is the matrix pseudoinverse. These represent the smallest joint velocities (in the ℓ_2-norm sense) that produce a particular endpoint velocity. All joint velocities can be decomposed into an endpoint component and a null component: $\omega = \omega_{null} + \omega_{endpt}$.

For tasks in which the endpoint must move toward a specified target p^*, endpoint velocity can be further divided into a goal component, which moves the endpoint along the axis $g = \frac{p^* - f(\vartheta)}{\|p^* - f(\vartheta)\|}$, and an error component, which moves the endpoint orthogonal to the goal axis. Thus we subdivide $\omega = \omega_{null} + \omega_{goal} + \omega_{error}$ in order to further analyze the endpoint component of joint velocities, using these equations:

$$\omega_{goal} = J^{\dagger}(\theta) g \cdot (J(\theta)\omega)^T g$$
$$\omega_{error} = \omega_{endpt} - \omega_{goal}.$$

BMI decoder architecture

In BMI systems, the decoder plays a central role in mapping the observed neural activity from the population of BMI neurons to kinematic control signals for the prosthesis. In this work, we utilize the Kalman filter (KF) as the decoder and calibrate it using closed-loop decoder adaptation (CLDA).

Kalman filter (KF)

Variants of the KF have been used in several online BMI experiments [25, 24, 27, 11, 26]. When used to control our four-link virtual kinematic chain, $\tilde{x}_t$ represents the subject's intended joint angles $\tilde{\theta}$ and joint velocities $\tilde{\omega}$ at time t:

$$\tilde{x}_t = \begin{bmatrix} \tilde{\theta}_t^{(1)} & \tilde{\theta}_t^{(2)} & \tilde{\theta}_t^{(3)} & \tilde{\theta}_t^{(4)} & \tilde{\omega}_t^{(1)} & \tilde{\omega}_t^{(2)} & \tilde{\omega}_t^{(3)} & \tilde{\omega}_t^{(4)} & 1 \end{bmatrix}^T.$$

The KF estimates the hidden state $\tilde{x}_t$ when it follows the Gaussian process

$$x_{t+1} = Ax_t + \alpha BL\left(x_t - x_{eq}\right) + w_t; \quad w_t \sim \mathcal{N}\left(0, W\right).$$

This model is a variant on the standard random walk model in which the walk is anchored around an equilibrium state x_{eq}. We hand-structured the state transition matrix A such that our model integrated joint velocities to update joint positions, structured the control input matrix B such that the inputs from the equilibrium state only affected the velocity states, and structured the state transition covariance W such that the process noise evolved independently for each joint:

$$A = \begin{bmatrix} I_4 & \Delta I_4 & 0 \\ 0 & 0.8 I_4 & 0 \\ 0 & 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 0_4 \\ \Delta I_4 \\ 0 \end{bmatrix}, W = \begin{bmatrix} 0_{4\times 4} & 0 & 0 \\ 0 & 0.01 I_4 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{6.3.1}$$

Though we selected values 0.8 and 0.01 arbitrarily, they were effectively overwritten by the CLDA procedure described below [32]. L is an LQR-optimal infinite horizon feedback gain matrix [85].

The observations y_t represent the spikes observed from a population of single and multi-units in the past 100 ms. The KF models $\tilde{x}_t$ and y_t as jointly Gaussian with the relationship

$$y_t = C\tilde{x}_t + q_t; \quad q_t \sim \mathcal{N}(0, Q).$$

This model of neural firing assumes that spikes are linearly related to joint kinematics. In cursor BMI studies, modeling neural activity as relating to position variables was found to detrimentally impact performance [24, 11, 32]. Therefore, we only modeled the neural dependence on the joint velocity and did not model a relationship between neural firing and joint position. The KF algorithm recursively generates a new state estimate x_t using the previous estimate x_{t-1} and the most recent observation y_t:

$$\begin{aligned} x_t &= (I - K_t C)[Ax_{t-1} + \alpha BL(x_{t-1} - x_{eq})] + K_t y_t \\ &= (I - K_t C)(A + \alpha BL)x_{t-1} - (I - K_t C)\alpha BL x_{eq} + K_t y_t \end{aligned} \tag{6.3.2}$$

More KF details are presented in [25]. For analysis purposes, it is convenient and sufficiently accurate to analyze the system as time-invariant, replacing K_t with the steady-state Kalman gain K as shown in Figure 6.1a [32].

The value α controls how strongly movement toward the equilibrium affects the state of the prosthesis. In our previous work, we found that a similar tendency toward an equilibrium state arises spontaneously in decoders with position-based models of neural activity [32]. Similar strategies have been used to prevent "drift" due to inattention or disengagement during control of a 4-DOF robot [6]. For Monkey C, the equilibrium was unused ($\alpha = 0$), and we sometimes observed the kinematic chain entering into configurations in which he would lose motivation to perform the task, possibly because the chain configuration became difficult to manipulate. To prevent this issue for Monkey G, we used $\alpha = 0.025$ during the center-out task to keep the chain configuration from drifting too far from the equilibrium. Since the equilibrium posture's endpoint position was at the center of the workspace, the subject remained fully responsible for movements out toward the peripheral targets. This method was not applied to the multi-configuration task ($\alpha = 0$) since in that task the configuration of the chain was reset at each trial onset and the purpose of the attractor was to reduce the control burden during continuous control conditions.

KF parameter calibration using closed-loop decoder adaptation (CLDA)

Parameters were calibrated using closed-loop decoder adaptation (CLDA), an emerging paradigm for rapidly improving closed-loop BMI performance by re-estimating decoder parameters while the subject continues to operate the system [2, 30, 11, 26, 86]. Parameter estimation procedures were adapted from [26] and [87].

Initial parameters for C and Q were set by regressing against neural activity evoked when the subjects passively observed computer-generated movements of the kinematic chain. They were then iteratively re-estimated as the subject performed the task in closed loop. If the intended kinematics are known, then maximum-likelihood estimates of C and Q can be

obtained by linear regression. While the intended kinematics cannot be obtained exactly, simple methods to infer intended kinematics based on the known goals of the task have been remarkably successful. For instance, the assumption that the subject always intends that the cursor velocity point toward the target has been used in several previous studies [11, 88].

The kinematic chain in this work has the added complication that the intended final joint angles are not fully specified by the intended endpoint location. To resolve this redundancy during CLDA, we constructed an arbitrary one-to-one mapping between target position p^* and intended configuration θ^* for each of the peripheral targets and the center target. CLDA was performed only on the center-out task. This mapping allows us to specify the goal state x^*, which we used to infer the intended kinematics by:

$$\tilde{x}_{t+1}^{est} = Ax_t + BL\left(x_t - x^*\right).$$

A, B and L are the same as in the KF state-space model. This intention estimation method is the multi-dimensional generalization of a feedback controller that slowly moves a cursor toward a target position [11].

With the estimates of intended kinematics, we use the recursive maximum likelihood method to update C and Q [87]. This calibration procedure was repeated daily for both monkeys. After 5-10 minutes of decoder adaptation, the decoder parameters were fixed for the remainder of the experimental session.

Control signal analysis and DOF manipulation

In equation (6.3.2), the term $K_t y_t$ can be interpreted as a control signal to a linear system. The instantaneous contribution of the neural control is encapsulated entirely inside this term. Our control signals Ky_t are elements of a 4-D space. The control signal Ky_t has 9 elements because x_t has 9 elements. However, because we do not model neural dependence on joint positions and because the last element of x_t is constant, only the 4 velocity inputs are unique and the other 5 control inputs are linearly related to the velocity inputs. Hence we limit our analyses to $u_t = K_{[5:8,:]}y_t$, the 4-D joint-velocity control. Our tasks only require movement of the 2-D endpoint, resulting in two redundant DOFs. For our analysis, we decomposed the control signals into goal, error, and null components (see Manipulator kinematics above), symbolized as g_t, e_t, and n_t, respectively.

Additionally, we analyzed principal components (PCs) of u_t, which represent the uncorrelated joint co-activation patterns of maximal variance [89]. If p_i is the i^{th} PC, then the component of a vector β along the ith PC is $\beta^{(i)} = p_i\left(p_i^T\beta\right)$. The scalar quantity $(p_i^T\beta)$ is the activation of dimension p_i. This gives us a method to split u_t into 12 components:

$$u_t = g_t^{(1)} + \cdots + g_t^{(4)} + e_t^{(1)} + \cdots + e_t^{(4)} + n_t^{(1)} + \cdots + n_t^{(4)}.$$

We tested several hypotheses regarding the fraction of variance (FV) accounted for by various dimensions or combinations of these components. For example, to determine the fraction of each PC contributing to either endpoint or null movement, we calculated the FV for $g_t^{(i)} + e_t^{(i)}$

versus $n_t^{(i)}$. Or to test the relative contribution of each PC to g_t, we calculated the FV due to $g_t^{(1)}$, $g_t^{(2)}$, $g_t^{(3)}$, and $g_t^{(4)}$. The variance of each vector was calculated by adding the variances of each of the individual components [89]. Variance estimates were made using 1000 samples (100 sec) of data between the go cue and the terminus hold, and only observations from rewarded trials were considered (see Figure 6.1).

We also examined the similarity of DOF use across tasks. PCA is a data-dependent determinant of directions of maximal variance, and in addition to fluctuating across days due to changes in the decoder or neural population, may change across tasks. Thus when comparing DOF usage across tasks, we determined which DOFs were used during center-out control (PCs averaged across sessions and then re-orthogonalized using Gram-Schmidt orthogonalization) and then used these center-out DOFs to make meaningful comparisons of control strategies across task manipulations.

In some closed-loop experiments, we restricted u_t to 2 PCs. If $\bar{u}$ is the mean of u_t, then

$$u_t - \bar{u} = \sum_{i=1}^{4} p_i p_i^T \left(u_t - \bar{u}\right).$$

To allow N PCs, we substituted $\bar{u} + \sum_{i=1}^{N} p_i p_i^T \left(u_t - \bar{u}\right)$ in place of Ky_t in equation (6.3.2).

6.4 Results

Subjects generated less stereotypic chain configurations as task performance improved

Both subjects successfully performed the center-out brain-control task with the 4-link kinematic chain. Figure 6.1a shows that movement time (the time elapsed between exiting the center and entering the target) decreased across days in both subjects (Monkey C: $R^2 = 0.61$, exponential fit; Monkey G: $R^2 = 0.355$, linear fit; $p < 0.02$) over the course of 17-24 sessions. Figure 6.1b shows that average movement time was 0.76 s faster for Monkey C and 0.39 s faster for Monkey G in late (last 5) sessions compared to early (first 5) sessions. The distributions of movement times between early and late sessions were significantly different as well (Kruskal-Wallis test, $p < 10^{-5}$).

The joint-space control signals, u_t, were partitioned into endpoint and null components, and endpoint components were further subdivided into goal and error components (see Methods). Figure 6.1c shows that the variance of the endpoint component increased from early to late sessions for both monkeys (Kruskal-Wallis test, $p < 0.001$). This increase suggests that in later sessions the monkeys utilized a broader range of control signals to accelerate or decelerate the endpoint. In parallel, we observed that in late sessions, a greater fraction of endpoint variability was related to control inputs along the goal axis than on the error axis (Figure 6.1d, Kruskal-Wallis test, $p < 0.001$), and was likely responsible for an observed decrease in the endpoint path length across sessions for Monkey G ($r = -0.61$, $p < 0.01$, linear

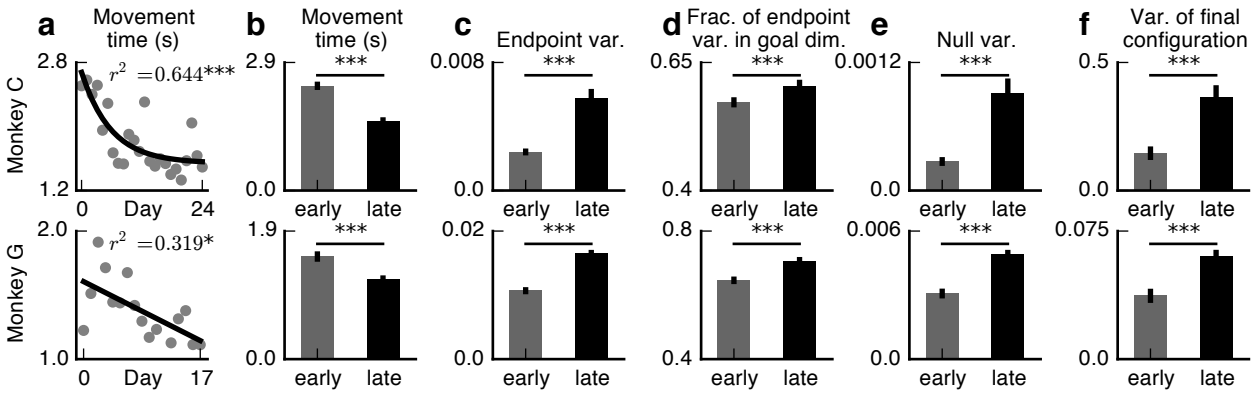

Figure 6.1: Performance improvements across sessions correspond to shifts in endpoint and null velocity control signals. (a) Both monkeys showed improved movement times across consecutive sessions on the center-out task. The performance improvements were significantly fit by an exponential curve for Monkey C ($R^2 = 0.644$, $p < 0.001$) and a line for Monkey G ($R^2 = 0.319$, $p < 0.05$). Early and late sessions were defined for further analysis as the first five and last five sessions. (b) Consistent with the curve fits from (a), movement times significantly improved in late sessions (Kruskal-Wallis (KW) test, $p < 0.001$). (c) Endpoint velocity variance, corresponding to the range of control inputs which accelerated or decelerated the endpoint, increased from early to late sessions (KW test, $p < 0.001$), consistent with faster movement times. (d) Endpoint velocity was split into goal and error dimensions, indicating endpoint velocity control inputs on the axis toward the target or the axis orthogonal to the target, respectively. The fraction of endpoint velocity on the goal axis increased in late sessions (KW test, $p < 0.001$), indirectly contributing to faster movement times by improving the instantaneous accuracy of endpoint control signals. (e) Null velocity variance, corresponding to control inputs that altered the chain configuration without moving the endpoint, increased in late sessions (KW test, $p < 0.001$). (f) The increase in null variance likely caused an increase in the total variance of joint configurations at the end of the trial (KW test, $p < 0.001$). Overall, in late sessions, the monkeys moved the endpoint of the chain to the target faster and in a larger range of final configurations. *:$p < 0.05$, **:$p < 0.01$, ***:$p < 0.001$.

fit). While Monkey C did not display a significant change in path length across sessions, he did demonstrate a decrease in the ratio of error variability to goal variability ($r = -0.81$, $p = 0.008$, linear fit) as well as path length ($r = -0.54$, $p < 0.001$) within-session. These improvements in performance are consistent with previous studies of co-adaptive BMI performance improvements [2, 88]. Both monkeys were able to generate faster, more accurate endpoint control signals in late sessions, resulting in shorter movement times.

A more unique component of our study is the redundancy in the kinematic chain, which allows null movements to reposition the chain without moving the endpoint. Figure 6.1e shows that variability in the null dimension increased from early sessions to late sessions (Kruskal-Wallis test, $p < 0.001$). On the behavioral level, we found that the configuration of the joints at the end of the trial became more variable across sessions. Figure 6.1f shows that the within-target, within-session configuration variance was significantly higher in late sessions than in early sessions (Kruskal-Wallis test, $p < 0.00002$ for both monkeys). These increases in null motion and configuration variance are somewhat counter to previously published observations of natural motor remapping studies [78], possibly because no physical movement is required for BMI control (see Discussion). The starting configuration of each trial, at the time of the go cue, was also more variable in later sessions than in early sessions (Kruskal-Wallis test, $p < 0.0005$), and the starting configuration in a trial was highly predictive of the ending configuration. Averaging across targets and sessions, the starting configuration explained 94% of the variance in the ending configuration variance for Monkey C and 69% for Monkey G. This may indicate that the subjects attempted to optimize their solutions on a trial-by-trial basis depending on the configuration at the start of each trial rather than minimizing variability across trials.

Altogether, the changes in null and endpoint control components allowed the monkeys to move to the targets faster and achieve a wider range of final configurations without sacrificing endpoint performance.

Movement times were slower without joint configuration feedback

We analyzed the importance of direct visual feedback of the chain configuration state on task performance. The subjects performed the multi-configuration task with all links of the chain hidden on 50% of trials (randomly chosen), leaving only the chain's endpoint visible. The multi-configuration task was used to begin each trial with the chain at carefully controlled starting configurations and prevent the chain configuration at the end of one trial from impacting the strategy for the next trial. Since there were multiple possible joint configurations for each starting endpoint position, subjects could not determine the exact chain configuration from the starting endpoint position alone. We found that movement times were longer on average for these reduced-feedback trials. Across starting configurations, movement times when the links were hidden were on average 5.5% longer for Monkey C and 13.3% longer for Monkey G ($p < 0.0001$, randomization test stratified by starting configuration; 3732 trials for Monkey C and 1763 trials for Monkey G). One configuration in particular required movement times that were 21.7% longer for Monkey C and 66.0% longer for Monkey G (Kruskal-Wallis

test, $p < 0.01$ corrected). This indicates that the control signal generated by the subjects was influenced by knowledge of the full configuration state of the redundant actuator and not exclusively the endpoint position.

To confirm that the observed feedback manipulation effect was specific to a plant in which the joint angles were not uniquely specified by the endpoint position, Monkey G performed a control experiment in which the four-link chain was replaced with a two-link chain. A two-link chain contains no kinematic redundancy, and the exact configuration of the joint angles can always be determined from the endpoint. Under these conditions, hiding the links had no effect on movement time ($p > 0.35$, randomization test stratified by starting configuration; 4632 trials total). The importance of seeing the kinematic chain links was limited to the four-link chain that lacked a one-to-one mapping between endpoint position and joint configuration.

Redundant principal components of the neural control signal contributed to task performance

To extract the DOFs controlled during the continuous center-out task, we performed principal component analysis on the joint velocity control signals u_t. The resulting PCs represent four orthogonal patterns of relative joint co-activation that span the joint velocity space. Since the four-link chain possesses two excess DOFs to complete the center-out task, we aimed to determine which control dimensions were modulated by task goals by calculating the average activation of each of the PCs during trials to different targets. We hypothesized that PC activity due to noise would occur at a consistent level regardless of the intended endpoint movement direction, while activity due to goal-directed control would appear tuned to target direction. Figure 6.2a shows that activity along all four PCs showed clear direction-dependent tuning (cosine curve fit, $p < 0.001$, averaged across sessions), with PCs 3 and 4 (the PCs with less FV) both most active during vertical endpoint movements. This suggests that despite the relatively small fraction of variance explained by these PCs, they were tuned to the target direction and thus modulated by the goals of the task.

The PC tuning in Figure 6.2a was significant when averaged across sessions, indicating that DOFs were used in a consistent manner across center-out sessions. Since PCs are defined by a particular set of joint movement observations and therefore may vary depending on the decoder, the neural ensemble, and the requirements of a given task, we also computed more general DOFs based on average PCs across multiple sessions of the center-out task (see Methods). Figure 6.2b shows that the first two DOFs had larger FV than the last two DOFs. Thus we refer to the first two DOFs as primary DOFs (pDOFs) and last two as secondary DOFs (sDOFs). To determine how each DOF contributes to movement of the endpoint, we partitioned each DOF into null and endpoint components. Figure 6.2c shows that the pDOFs were more than four times more active in the endpoint dimension than in the null dimension for both monkeys (Kruskal-Wallis test, $p < 10^{-7}$), while the reverse was true for the sDOFs (Kruskal-Wallis test, $p < 10^{-7}$). From this, we concluded that the pDOFs

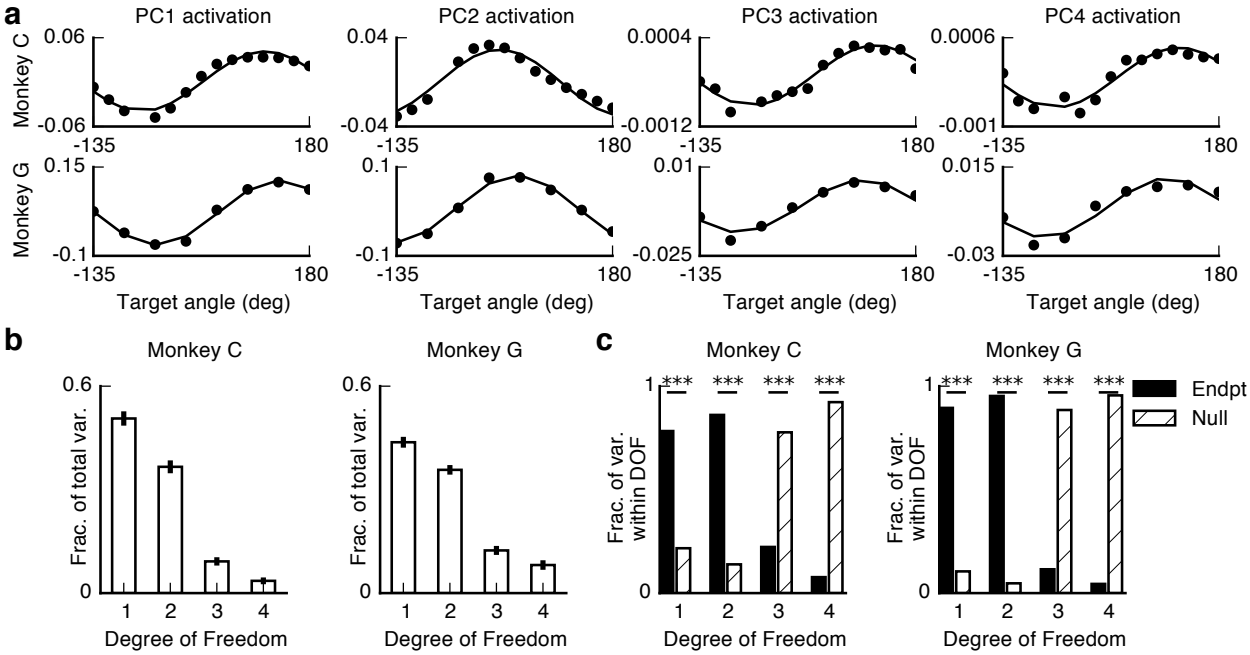

Figure 6.2: Principal components (PCs) and degrees of freedom (DOFs) during center-out control. (a) For the center-out task, we calculated PCs of the joint velocity control space (see Methods) and plotted the average tuning of these PCs to target direction across sessions. All PCs were significantly fit by a cosine tuning function ($p < 0.001$). (b) Fraction of variance (FV) along each DOF, averaged across sessions. Center-out DOFs were constructed by averaging the center-out PCs from individual sessions (see Methods). Center-out control was dominated by the use of the "primary" DOFs 1 and 2 (pDOFs), while the "secondary" DOFs 3 and 4 (sDOFs) accounted for a much smaller FV. (c) Within each DOF, we calculated the average FV by either endpoint or null movements. Activity in the pDOFs was primarily endpoint movement while activity in the sDOFs was primarily null movement.

were the main drivers of endpoint movement during center-out control.

We asked whether the seemingly minor contribution of the sDOFs to center-out control reflected a global, task-independent control strategy, or whether there were regions of configuration space not traversed during the center-out task where the sDOFs might play a more prominent role. We examined the usage of these same sDOFs during the multi-configuration task, in which a more varied range of joint configurations occurred than in the center-out task. Figure 6.3a illustrates sDOF use by target location (center-out task) or starting configuration (multi-configuration task). For many starting configurations of the multi-configuration task, the fraction of endpoint variance due to sDOFs was higher observed during the center-out task. To further explore this effect, we had Monkey G perform a modification of the multi-configuration task specifically designed to encourage movements using secondary PCs and to penalize activity in the primary PCs (see Methods; an example

is shown in Figure 6.3b). When controlling from these configurations, fraction of endpoint variance due to the sDOFs increased significantly (Figure 6.3a, "Singular Primary PCs") relative to both the standard multi-configuration and the center-out tasks. The sDOFs contributed 33% of endpoint movements from these configurations despite only contributing 13% during center-out control. Furthermore, the fraction of goal variance due to the sDOFs was on average 75% from these configurations. Not only were the sDOFs responsible for a large portion of endpoint movement from these particular configurations, they were primarily responsible for movements toward the goal.

We observed that faster trials in the multi-configuration task had a greater FV due to sDOFs. In all trials for this analysis, the entire chain was visible. For each starting configuration, we split trials into two groups based on whether they were faster or slower than the median movement time for that starting configuration. Figure 6.3c shows that the fast trials (aggregated over all starting configurations) tended to utilize more sDOF activity than the slow trials (Kruskal-Wallis test, $p < 0.001$). This same result held within-configuration for 10 of 20 configurations for Monkey C and all 12 configurations for Monkey G (Kruskal-Wallis test, $p < 0.01$ Holm-Bonferroni corrected). This relationship indicates that sDOFs enhanced task performance.

We performed a separate supplemental control experiment to confirm non-trivial contribution of the sDOFs during center-out control. Monkey G performed the center-out task with a decoder that was manipulated to only allow movement along the first two PCs, and the performance of this was compared to performance with the full four PCs (1189 trials in each condition, four PC data was collected on the same sessions as two PC data and did not overlap with any data in Figure 6.1). Average movement times were 11% worse in the 2-PC condition and increased significantly from 1.28 s to 1.42 s (Kruskal-Wallis test, $p = 0.002$). The performance gap was most significant when moving toward targets at -90 (0.83s (49.1%) difference, $p < 0.001$ corrected), -45 (0.32s (21.1%) difference, $p < 0.001$ corrected), and 180 degrees (0.15s (13.0%) difference, $p < 0.05$ corrected). These directions also exhibited corresponding large fractions of endpoint variance due to the sDOFs (Figure 6.2a).

Altogether, these results indicate that despite the 2-D nature of the task, the subjects' control strategies utilized the redundant sDOFs in addition to the pDOFs. The relative usage of sDOFs depended on the specific task requirements, and in fact, for some parts of configuration space, the sDOFs defined by the center-out task were more important than the pDOFs.

6.5 Discussion

The restoration of upper limb function is one of the most common goals of BMI research. To fully match the functionality of the natural arm, a BMI system must provide the neural circuitry a mechanism for generating specific joint postures in addition to hand movements, an ability that is essential for certain types of movements such as reaching around obstacles.

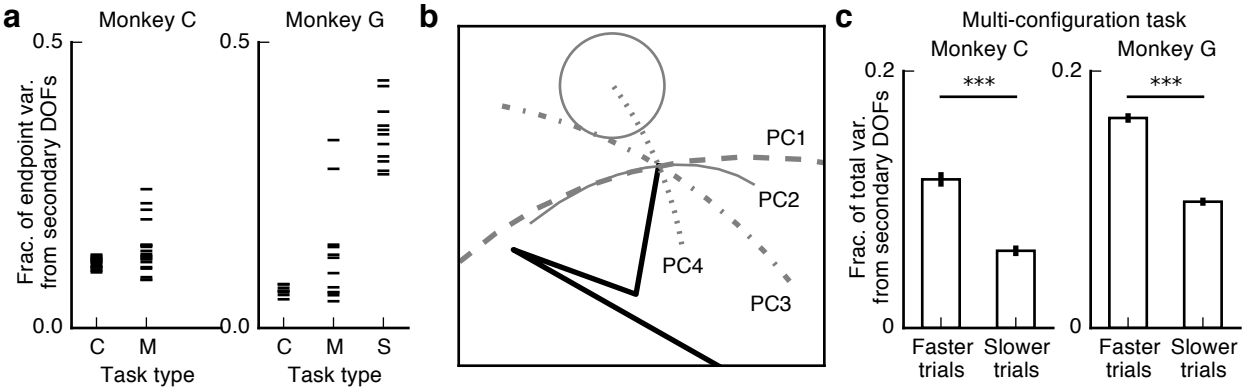

Figure 6.3: Contribution of secondary degrees of freedom (sDOFs) to BMI control and performance. (a) Comparison of fraction of variance due to sDOFs across different tasks (C: center-out, M: multi-configuration, S: Singular Primary PCs). Some starting configurations in the multi-configuration task showed a greater usage of sDOFs than in the center-out task, while even greater sDOF utilization was observed in a modified version of the multi-configuration task in which configurations and target locations were arranged to penalize primary DOF (pDOF) use and encourage sDOF use (Singular Primary PCs, Monkey G only). This demonstrated that the reliance on pDOFs observed during the center-out task was due the specific task requirements and not a global inability to modulate the sDOFs. (b) An example configuration for the modified multi-configuration task to encourage sDOF use. Velocity control signals from PCs 1 and 2 generate endpoint velocities that are roughly parallel to each other and orthogonal to the direction of the target, while control signals along PCs 3 and 4 generate endpoint velocities toward the target. (c) On the multi-configuration task, trials with faster than median movement times tended to have a larger fraction of total variance from sDOFs than trials with slower than median movement times.

Providing the user with direct control at the joint or even muscle level is one possible way to incorporate this functionality. Studies involving robotic BMI control have typically only allowed direct neural control of the end effector [3, 6, 13, **?**], with the generation of a corresponding configuration-space trajectory (e.g., joint positions) left to robotic path planning algorithms. Systems that instead seek to reanimate the natural limb through functional electrical stimulation of intact muscles have the potential to allow control of redundant aspects of the limb state, but relatively few examples of this type of control have been published. In a recent study in which brain activity directly controlled functional electrical stimulation (FES) of 3-5 hand and arm muscles to perform a grasping task [10], the stimulated muscles likely formed a redundant mapping to the subjects' grasp. The FES paradigm demonstrates the clinical value of understanding neural control of redundant actuators.

It is important to note that the neural redundancy present in many BMIs is distinct from kinematic redundancy. Most BMIs contain neural redundancy in that there are typically many more neurons directly involved in closed-loop control than there are controllable DOFs

in the plant. However, there is an important distinction between kinematic redundancy in the manipulable DOFs of the plant and redundancy in the neural control signal: the user receives direct sensory feedback about the redundant plant state, but feedback about the state of individual redundant neural inputs can only be gained indirectly through observation of their combined effect on the plant. Because BMIs are closed-loop control systems, this difference in feedback conditions may result in the two types of redundancy exerting different effects on subjects' control. The impact of neural redundancy on BMI performance remains an open question and deserves further study.

A series of experiments investigated abstract motor learning of kinematically redundant maps [78, 80, 90, 91, 92]. Subjects learned novel mappings from high-dimensional 19-DOF hand position to 2-D cursor position in order to perform cursor movement tasks. This paradigm differed from ours in that it remapped natural hand kinematics rather than utilizing direct cortical control of the redundant parameters. Subjects performing the hand-to-cursor control tasks reduced variability of the cursor position space by producing straighter trajectories, and reduced variability in the null position space by acquiring the target in more consistent configurations [78]. However, this null variability was not eliminated completely, and with continuous visual feedback subjects used remaining null movements to reduce energetic control costs [91]. In our data, we observed an increase in null variability in the velocity space as well as an increase in variability of the chain configuration at the end of the trial. This difference in strategy may be due to different cost functions between the two paradigms. The cost of null movement in our study is the metabolic cost of spiking activity to produce null movement, but it is unclear how to relate that cost to the cost of physical movement as in the hand remapping studies, or how to determine the cost of reshaping the network dynamics to suppress the null activity. It may have been difficult or impossible to generate neural activity to increase task-relevant speed without also generating larger null velocity components. In our paradigm, the more efficient strategy was apparently to increase null variability, consistent with a minimum intervention strategy in which the subjects learned to ignore or exploit motion in the null space rather than try to correct it [70].

Both subjects in our study engaged the redundant plant DOFs in small but non-trivial amounts. The coarse elements of control were dominated by the first two DOFs, but the third and fourth DOFs still influenced finer aspects of control. This control structure is similar to redundant DOFs that are engaged in natural motor control. For example, in hand grasping postures, though most of the postural variance can be explained by the first 2 PCs ($>$80%), the higher order PCs still contain information about the object being grasped [93].

While our subjects demonstrated that control of the redundant elements of the BMI plant can be achieved, it remains unclear whether permitting control of redundant DOFs would be advantageous in a clinical BMI system. Proprioception plays an important role in natural movement [94] and a few studies have shown improved BMI performance when some form of proprioceptive feedback was provided [9, 95]. Despite this, most BMI implementations are limited to visual feedback of a disembodied actuator. Under these impoverished sensory feedback conditions, it may be that increasing the number of controllable DOFs significantly increases the cortical control burden. Extensive practice at controlling such a BMI system

could potentially alleviate this issue to some degree, as many studies have demonstrated the brain's remarkable plasticity in learning both novel motor (e.g., [96, 97, 98, 99, 100]) and BMI skills (e.g., [19, 7, 8, 101, 17, 102]). However, clinical studies have shown that ease of use and training are important factors in patients' decisions to accept and use prosthetics [103, 104]. Thus in practice there will be a tradeoff between the complexity of learning to operate the system and the potential advantages of controlling more DOFs. Evaluating this tradeoff may be important to future clinical deployment of BMIs.

While our CLDA methodology was sufficient for subjects to achieve control of the plant in this study, it may be suboptimal for a redundant system since the presence of kinematic redundancy complicates the process of estimating intended movements for decoder calibration. Related calibration issues were raised by Collinger and colleagues [82]. Inferring the exact intended trajectory is never possible with any CLDA method, even when the optimal final state for a movement can be determined exactly. Our system poses the additional difficulty that for an endpoint movement task, the redundancy makes it such that any element of a 2-D state subspace is equally optimal. Our CLDA method makes strict assumptions about the goal pose of the kinematic chain. This method could be adapted to the plant of a clinical BMI system if a similar set of instructed configurations can be specified (e.g., useful limb postures for activities of daily living). Alternative CLDA methods based on reinforcement learning (e.g., [31]) or LMS-like methods to minimize endpoint error [105] may also achieve high performance without explicitly assuming the subject is trying to attain a particular chain configuration. Further experiments are necessary to test these and other methods for redundant systems. However, no matter how good the CLDA method may be, the lack of sensory feedback compared to natural motor control is likely to remain a problem.

The BMI paradigm introduced in this study provides a potential framework for investigating elements of motor learning. For example, the concept of muscle synergies involves a mapping from a redundant parameter space to one with fewer DOFs (see [106] for a review). A redundant BMI system in which muscle activations are decoded could provide a useful testbed for synergy theories by enabling researchers to easily specify or disable synergies at will and observe the resultant effects on control, learning, and adaptation. Such "virtual lesions" have been proposed to study myoelectric control (e.g., [107]). One strong confound to these studies in natural motor control may be force of habit in muscle coordination [108]. In an environment with novel geometry, such as a non-biomimetic plant, this problem may be mitigated. Though other novel environments can test control hypotheses [78], BMI systems may also provide a valuable insight into how cortical structures may be involved in learning novel redundant systems and compensating for loss in redundancy.

Chapter 7

Conclusion

BMIs provide the potential to restore the ability for millions to independently perform the activities of daily living in cases where motor function has been lost to neurological disease or injury. This dissertation aimed to understand the details of the strategy used during BMI operation and use this understanding to extend the capability of BMI systems to higher dimensional prostheses.

Chapter 3 demonstrated that day-to-day task performance is influenced by the specific decoder parameters selected by our calibration procedure. The presence of certain dynamical properties adversely impacted performance and a tradeoff between speed and accuracy was traced back to decoder parameters. We reasoned, based on simulations, that this performance impact was due to a model mismatch between the subject's model of the decoder and the actual decoder properties. Chapter 5 expanded on this line of inquiry, searching for evidence that such a model existed. We found that in the error clamp experimental paradigm, one monkey subject learned to compensate for a perturbation in a way consistent with constructing a model of the perturbation applied.

In Chapter 4, we developed a method for quickly extracting sparse "submovement" components from kinematic trajectories. Submovements are a well-known property of natural movement kinematics, but their origin is unknown. It is unclear whether they reflect the control policy itself or instead reflect interaction between the controller and the musculature. A submovement-based neural controller of movement suggests partial feedforward control. In Chapter 5, we briefly explored the possibility that submovements exist in BMI control, in the absence of the musculature. As one monkey subject learned to more quickly move a point-mass BMI cursor, the velocity profiles became smoother and had fewer submovement-like components. Similar results have been found in stroke recovery studies [50], suggesting that as the subjects learn the initial submovements become more accurate, fewer large corrections are necessary.

Our study of 2-D natural reaching movements and 2-D cursor BMI control enabled us to develop the architecture of the 4-DOF kinematic chain BMI described in Chapter 6. To our knowledge, this is the first investigation of how a BMI subject dealt with redundant system elements. Redundancy is a critical component of natural movements and how the natural

motor system resolves redundancy is still not well understood. In our work, we showed that with visual feedback alone, BMI subjects were able to manipulate redundant elements to achieve task goals. These results suggest that redundant DOFs should receive neural control input. Current studies typically leave redundant DOFs purely to machine control. By allowing a combination of neural control and robotic control, control over high-DOF BMIs may improve. Further experiments are needed to test this idea.

The motivation for much of this work came from the desire to build a BMI to control an upper arm exoskeleton. Though we have begun initial deployment of some of the concepts developed here into BMI control of an exoskeleton, much work remains to be done in the control of a real world dynamics. Thus far, the BMI field has yet to address the important aspect of object interaction. Force interactions with objects are difficult for robotic controllers in unknown environments as interactions with the environment may change the inertia of the robot in unknown ways. Animals are quite good at learning to manipulate impedance to deal with unknown environments (e.g., [109]), but BMIs have yet to enable control over limb impedance. This remains an important area for BMI algorithm development.

Bibliography

[1] M. D. Serruya, N. G. Hatsopoulos, L. Paninski, M. R. Fellows, and J. P. Donoghue, "Brain-machine interface: Instant neural control of a movement signal," *Nature*, vol. 416, no. 6877, pp. 141–142, 2002.

[2] D. M. Taylor, S. I. H. Tillery, and A. B. Schwartz, "Direct Cortical Control of 3d Neuroprosthetic Devices," *Science*, vol. 296, pp. 1829 –1832, June 2002.

[3] 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*, vol. 1, no. 2, p. e2, 2003.

[4] S. Musallam, "Cognitive Control Signals for Neural Prosthetics," *Science*, vol. 305, pp. 258–262, July 2004.

[5] G. Santhanam, S. I. Ryu, B. M. Yu, A. Afshar, and K. V. Shenoy, "A high-performance brain-computer interface," *Nature*, vol. 442, pp. 195–198, July 2006.

[6] M. Velliste, S. Perel, M. C. Spalding, A. S. Whitford, and A. B. Schwartz, "Cortical control of a prosthetic arm for self-feeding," *Nature*, vol. 453, pp. 1098–1101, May 2008.

[7] C. T. Moritz, S. I. Perlmutter, and E. E. Fetz, "Direct control of paralysed muscles by cortical neurons," *Nature*, vol. 456, pp. 639–642, Oct. 2008.

[8] K. Ganguly and J. M. Carmena, "Emergence of a stable cortical map for neuroprosthetic control," *PLoS biology*, vol. 7, no. 7, p. e1000153, 2009.

[9] 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*, vol. 30, pp. 16777–16787, Dec. 2010.

[10] C. Ethier, E. R. Oby, M. J. Bauman, and L. E. Miller, "Restoration of grasp following paralysis through brain-controlled stimulation of muscles," *Nature*, vol. 485, pp. 368–371, Apr. 2012.

[11] 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*, vol. 15, pp. 1752–1757, Nov. 2012.

[12] 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*, vol. 442, pp. 164–171, July 2006.

[13] L. R. Hochberg, D. Bacher, B. Jarosiewicz, N. Y. Masse, J. D. Simeral, J. Vogel, S. Haddadin, J. Liu, S. S. Cash, P. van der Smagt, and J. P. Donoghue, "Reach and grasp by people with tetraplegia using a neurally controlled robotic arm," *Nature*, vol. 485, pp. 372–375, May 2012.

[14] J. L. Collinger, B. Wodlinger, J. E. Downey, W. Wang, E. C. Tyler-Kabara, D. J. Weber, A. J. McMorland, M. Velliste, M. L. Boninger, and A. B. Schwartz, "High-performance neuroprosthetic control by an individual with tetraplegia," *The Lancet*, 2012.

[15] K. Ganguly, D. F. Dimitrov, J. D. Wallis, and J. M. Carmena, "Reversible large-scale modification of cortical networks during neuroprosthetic control," *Nature Neuroscience*, vol. 14, pp. 662–667, May 2011.

[16] K. B. Clancy, A. C. Koralek, R. M. Costa, D. E. Feldman, and J. M. Carmena, "Volitional modulation of optically recorded calcium signals during neuroprosthetic learning," *Nature Neuroscience*, vol. 17, pp. 807–809, Apr. 2014.

[17] A. C. Koralek, X. Jin, J. D. Long II, R. M. Costa, and J. M. Carmena, "Corticostriatal plasticity is necessary for learning intentional neuroprosthetic skills," *Nature*, vol. 483, pp. 331–335, Mar. 2012.

[18] T. Gulati, D. S. Ramanathan, C. C. Wong, and K. Ganguly, "Reactivation of emergent task-related ensembles during slow-wave sleep after neuroprosthetic learning," *Nature Neuroscience*, vol. 17, pp. 1107–1113, July 2014.

[19] 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*, vol. 105, no. 49, pp. 19486–19491, 2008.

[20] 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*, vol. 108, pp. 624–644, Apr. 2012.

[21] 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*, vol. 512, pp. 423–426, Aug. 2014.

[22] E. V. Evarts, "Relation of pyramidal tract activity to force exerted during voluntary movement.," *Journal of neurophysiology*, vol. 31, no. 1, pp. 14–27, 1968.

[23] 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," *The Journal of Neuroscience*, vol. 2, no. 11, pp. 1527–1537, 1982.

[24] 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*, vol. 5, pp. 455–476, Dec. 2008.

[25] 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*, vol. 18, no. 1, pp. 80–118, 2006.

[26] A. L. Orsborn, S. Dangi, H. G. Moorman, and J. M. Carmena, "Closed-Loop Decoder Adaptation on Intermediate Time-Scales Facilitates Rapid BMI Performance Improvements Independent of Decoder Initialization Conditions," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 20, pp. 468–477, July 2012.

[27] Z. Li, J. E. O'Doherty, M. A. Lebedev, and M. A. Nicolelis, "Adaptive decoding for brain-machine interfaces through Bayesian parameter updates," *Neural computation*, vol. 23, no. 12, pp. 3162–3204, 2011.

[28] W. Truccolo, "A Point Process Framework for Relating Neural Spiking Activity to Spiking History, Neural Ensemble, and Extrinsic Covariate Effects," *Journal of Neurophysiology*, vol. 93, pp. 1074–1089, Sept. 2004.

[29] G. J. Gage, K. A. Ludwig, K. J. Otto, E. L. Ionides, and D. R. Kipke, "NaÃ¯ve coadaptive cortical control," *Journal of Neural Engineering*, vol. 2, pp. 52–63, June 2005.

[30] L. Shpigelman, H. Lalazar, and E. Vaadia, "Kernel-ARMA for Hand Tracking and Brain-Machine interfacing During 3d Motor Control.," in *NIPS*, pp. 1489–1496, 2008.

[31] B. Mahmoudi and J. C. Sanchez, "A symbiotic brain-machine interface through value-based decision making," *PloS one*, vol. 6, no. 3, p. e14760, 2011.

[32] 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*, pp. 1–1, 2014.

[33] 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*, vol. 10, p. 026011, Apr. 2013.

[34] S. M. Chase, A. B. Schwartz, and R. E. Kass, "Latent Inputs Improve Estimates of Neural Encoding in Motor Cortex," *Journal of Neuroscience*, vol. 30, pp. 13873–13882, Oct. 2010.

[35] E. Hwang, P. Bailey, and R. Andersen, "Volitional Control of Neural Activity Relies on the Natural Motor Repertoire," *Current Biology*, vol. 23, pp. 353–361, Mar. 2013.

[36] B. D. O. Anderson and J. B. Moore, *Optimal control: linear quadratic methods*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1990.

[37] 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*, vol. 19, pp. 25–34, Feb. 2011.

[38] P. Berens, "CircStat: a MATLAB toolbox for circular statistics," *Journal of Statistical Software*, vol. 31, no. 10, pp. 1–21, 2009.

[39] L. Srinivasan, U. T. Eden, A. S. Willsky, and E. N. Brown, "A state-space analysis for reconstruction of goal-directed movements using neural signals," *Neural computation*, vol. 18, no. 10, pp. 2465–2494, 2006.

[40] A. Vato, M. Semprini, E. Maggiolini, F. D. Szymanski, L. Fadiga, S. Panzeri, and F. A. Mussa-Ivaldi, "Shaping the Dynamics of a Bidirectional Neural Interface," *PLoS Computational Biology*, vol. 8, p. e1002578, July 2012.

[41] M. Golub, S. Chase, and B. M. Yu, "Learning an Internal Dynamics Model from Control Demonstration," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)* (S. Dasgupta and D. Mcallester, eds.), vol. 28, pp. 606–614, JMLR Workshop and Conference Proceedings, 2013.

[42] S. Gowda, S. Overduin, M. Chen, Y.-H. Chang, C. Tomlin, and J. Carmena, "Accelerating submovement decomposition with search space reduction heuristics," *IEEE Transactions on Biomedical Engineering*, pp. 1–1, 2015.

[43] M. A. Smith, J. Brandt, and R. Shadmehr, "Motor disorder in Huntington's disease begins as a dysfunction in error feedback control," *Nature*, vol. 403, pp. 544–549, Feb. 2000.

[44] T. Schenk, E. U. Walther, and N. Mai, "Closed-and open-loop handwriting performance in patients with multiple sclerosis," *Eur. J. Neurol.*, vol. 7, no. 3, pp. 269–279, 2000.

[45] J. H. Yan, S. Rountree, P. Massman, R. S. Doody, and H. Li, "Alzheimer's disease and mild cognitive impairment deteriorate fine movement control," *J Psychiat Res*, vol. 42, pp. 1203–1212, Oct. 2008.

[46] A. Fishbach, S. A. Roy, C. Bastianen, L. E. Miller, and J. C. Houk, "Deciding when and how to correct a movement: discrete submovements as a decision making process," *Exp Brain Res*, vol. 177, pp. 45–63, Jan. 2007.

[47] L. Dipietro, H. Poizner, and H. I. Krebs, "Spatiotemporal Dynamics of Online Motor Correction Processing Revealed by High-density Electroencephalography," *J Cognitive Neurosci*, vol. 26, pp. 1966–1980, Sept. 2014.

[48] T. Hall, F. de Carvalho, and A. Jackson, "A Common Structure Underlies Low-Frequency Cortical Dynamics in Movement, Sleep, and Sedation," *Neuron*, Aug. 2014.

[49] A. M. Krylow and W. Z. Rymer, "Role of intrinsic muscle properties in producing smooth movements," *IEEE Trans. Biomed. Eng.*, vol. 44, no. 2, pp. 165–176, 1997.

[50] B. Rohrer, S. Fasoli, H. I. Krebs, B. Volpe, W. R. Frontera, J. Stein, and N. Hogan, "Submovements grow larger, fewer, and more blended during stroke recovery.," *Motor control*, vol. 8, no. 4, 2004.

[51] D. E. Meyer, R. A. Abrams, S. Kornblum, C. E. Wright, and J. E. K. Smith, "Optimality in human motor performance: ideal control of rapid aimed movements," *Psychol Rev*, vol. 95, pp. 340–370, 1988.

[52] A. Fishbach, S. A. Roy, C. Bastianen, L. E. Miller, and J. C. Houk, "Kinematic properties of on-line error corrections in the monkey," *Exp Brain Res*, vol. 164, pp. 442–457, Aug. 2005.

[53] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *J Neurosci*, vol. 5, no. 7, pp. 1688–1703, 1985.

[54] R. Plamondon, A. M. Alimi, P. Yergeau, and F. Leclerc, "Modelling velocity profiles of rapid movements: a comparative study," *Biol Cybern*, vol. 69, no. 2, pp. 119–128, 1993.

[55] B. Rohrer and N. Hogan, "Avoiding spurious submovement decompositions: a globally optimal algorithm," *Biol Cybern*, vol. 89, pp. 190–199, Sept. 2003.

[56] B. Rohrer and N. Hogan, "Avoiding Spurious Submovement Decompositions II: A Scattershot Algorithm," *Biol Cybern*, vol. 94, pp. 409–414, May 2006.

[57] T. Flash and E. Henis, "Arm trajectory modifications during reaching towards visual targets," *J Cognitive Neurosci*, vol. 3, no. 3, pp. 220–230, 1991.

[58] J. Friedman, T. Hastie, H. HÃ¶fling, and R. Tibshirani, "Pathwise coordinate optimization," *Ann Appl Stat*, vol. 1, pp. 302–332, Dec. 2007.

[59] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer Series in Statistics, New York, NY, USA: Springer New York Inc., 2001.

[60] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT press, 2005.

[61] N. Hansen, A. Niederberger, L. Guzzella, and P. Koumoutsakos, "A Method for Handling Uncertainty in Evolutionary Optimization With an Application to Feedback Control of Combustion," *IEEE Trans. Evolut. Comput.*, vol. 13, pp. 180–197, Feb. 2009.

[62] A. V. Roitman, S. G. Massaquoi, K. Takahashi, and T. J. Ebner, "Kinematic Analysis of Manual Tracking in Monkeys: Characterization of Movement Intermittencies During a Circular Tracking Task," *J Neurophysiol*, vol. 91, no. 2, pp. 901–911, 2004.

[63] K. Novak, L. Miller, and J. Houk, "Kinematic properties of rapid hand movements in a knob turning task," *Exp Brain Res*, vol. 132, pp. 419–433, June 2000.

[64] T. Flash, R. Inzelberg, E. Schechtman, and A. D. Korczyn, "Kinematic analysis of upper limb trajectories in Parkinson's disease," *Exp Neurol*, vol. 118, no. 2, pp. 215 – 226, 1992.

[65] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," in *Adv. neural info. proc. sys. (NIPS)*, pp. 801–808, 2006.

[66] R. Raina, A. Madhavan, and A. Y. Ng, "Large-scale deep unsupervised learning using graphics processors," in *Proc. ICML 26*, pp. 873–880, ACM, 2009.

[67] J. W. Krakauer, "Motor learning: its relevance to stroke recovery and neurorehabilitation," *Curr Opin Neurol*, vol. 19, no. 1, pp. 84–90, 2006.

[68] C. Walker, B. J. Brouwer, and E. G. Culham, "Use of visual feedback in retraining balance following acute stroke," *Phys Ther*, vol. 80, no. 9, pp. 886–895, 2000.

[69] C. M. Sackley and N. B. Lincoln, "Single blind randomized controlled trial of visual feedback after stroke: effects on stance symmetry and function," *Disabil Rehabil*, vol. 19, no. 12, pp. 536–546, 1997.

[70] E. Todorov and M. I. Jordan, "Optimal feedback control as a theory of motor coordination," *Nat Neurosci*, vol. 5, pp. 1226–1235, Nov. 2002.

[71] A. M. Haith and J. W. Krakauer, "Model-Based and Model-Free Mechanisms of Human Motor Learning," in *Progress in Motor Control* (M. J. Richardson, M. A. Riley, and K. Shockley, eds.), vol. 782, pp. 1–21, New York, NY: Springer New York, 2013.

[72] R. Shadmehr and F. A. Mussa-Ivaldi, "Adaptive representation of dynamics during learning of a motor task," *The Journal of Neuroscience*, vol. 14, no. 5, pp. 3208–3224, 1994.

[73] R. A. Scheidt, D. J. Reinkensmeyer, M. A. Conditt, W. Z. Rymer, and F. A. Mussa-Ivaldi, "Persistence of Motor Adaptation During Constrained, Multi-Joint, Arm Movements," *Journal of Neurophysiology*, vol. 84, no. 2, pp. 853–862, 2000.

[74] S. H. Scott, "The role of primary motor cortex in goal-directed movements: insights from neurophysiological studies on non-human primates," *Current Opinion in Neurobiology*, vol. 13, pp. 671–677, Dec. 2003.

[75] Y. Uno, M. Kawato, and R. Suzuki, "Formation and control of optimal trajectory in human multijoint arm movement," *Biological cybernetics*, vol. 61, no. 2, pp. 89–101, 1989.

[76] C. M. Harris and D. M. Wolpert, "Signal-dependent noise determines motor planning," *Nature*, vol. 394, no. 6695, pp. 776–780, 1998.

[77] J. P. Scholz and G. SchÃ¶ner, "The uncontrolled manifold concept: identifying control variables for a functional task," *Experimental brain research*, vol. 126, no. 3, pp. 289–306, 1999.

[78] K. M. Mosier, "Remapping Hand Movements in a Novel Geometrical Environment," *Journal of Neurophysiology*, vol. 94, pp. 4362–4372, Dec. 2005.

[79] X. Liu and R. A. Scheidt, "Contributions of Online Visual Feedback to the Learning and Generalization of Novel Finger Coordination Patterns," *Journal of Neurophysiology*, vol. 99, pp. 2546–2557, Feb. 2008.

[80] 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*, vol. 105, pp. 454–473, Jan. 2011.

[81] J. M. Carmena, "Advances in Neuroprosthetic Learning and Control," *PLoS Biology*, vol. 11, p. e1001561, May 2013.

[82] B. Wodlinger, J. E. Downey, E. C. Tyler-Kabara, A. B. Schwartz, M. L. Boninger, and J. L. Collinger, "Ten-dimensional anthropomorphic arm control in a human brain-machine interface: difficulties, solutions, and limitations," *Journal of Neural Engineering*, vol. 12, p. 016011, Feb. 2015.

[83] J. W. Burdick, "On the inverse kinematics of redundant manipulators: Characterization of the self-motion manifolds," in *Advanced Robotics: 1989*, pp. 25–34, Springer, 1989.

[84] P. Corke, *Robotics, vision and control: fundamental algorithms in MATLAB*, vol. 73. Springer Science & Business Media, 2011.

[85] M. M. Shanechi and J. M. Carmena, "Optimal feedback-controlled point process decoder for adaptation and assisted training in brain-machine interfaces," in *Neural Engineering (NER), 2013 6th International IEEE/EMBS Conference on*, pp. 653–656, IEEE, 2013.

[86] S. Dangi, A. L. Orsborn, H. G. Moorman, and J. M. Carmena, "Design and analysis of closed-loop decoder adaptation algorithms for brain-machine interfaces," *Neural computation*, vol. 25, no. 7, pp. 1693–1731, 2013.

[87] S. Dangi, S. Gowda, H. G. Moorman, A. L. Orsborn, K. So, M. 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*, vol. 26, pp. 1811–1839, Sept. 2014.

[88] A. Orsborn, H. Moorman, S. Overduin, M. Shanechi, D. Dimitrov, and J. Carmena, "Closed-Loop Decoder Adaptation Shapes Neural Plasticity for Skillful Neuroprosthetic Control," *Neuron*, vol. 82, pp. 1380–1393, June 2014.

[89] I. Jolliffe, "Principal Component Analysis," in *Wiley StatsRef: Statistics Reference Online*, John Wiley & Sons, Ltd, 2014.

[90] F. A. Mussa-Ivaldi, M. Casadio, Z. C. Danziger, K. M. Mosier, and R. A. Scheidt, "Sensory motor remapping of space in human-machine interfaces," in *Progress in Brain Research*, vol. 191, pp. 45–64, Elsevier, 2011.

[91] R. Ranganathan, A. Adewuyi, and F. A. Mussa-Ivaldi, "Learning to be Lazy: Exploiting Redundancy in a Novel Task to Minimize Movement-Related Effort," *Journal of Neuroscience*, vol. 33, pp. 2754–2760, Feb. 2013.

[92] R. Ranganathan, J. Wieser, K. M. Mosier, F. A. Mussa-Ivaldi, and R. A. Scheidt, "Learning Redundant Motor Tasks with and without Overlapping Dimensions: Facilitation and Interference Effects," *Journal of Neuroscience*, vol. 34, pp. 8289–8299, June 2014.

[93] M. Santello, M. Flanders, and J. F. Soechting, "Postural hand synergies for tool use," *The Journal of Neuroscience*, vol. 18, no. 23, pp. 10105–10115, 1998.

[94] K. G. Pearson, "Common principles of motor control in vertebrates and invertebrates," *Annual review of neuroscience*, vol. 16, no. 1, pp. 265–297, 1993.

[95] M. Gomez-Rodriguez, J. Peters, J. Hill, B. SchÃ¶lkopf, A. Gharabaghi, and M. Grosse-Wentrup, "Closing the sensorimotor loop: haptic feedback facilitates decoding of motor imagery," *Journal of Neural Engineering*, vol. 8, p. 036005, June 2011.

[96] A. Karni, G. Meyer, P. Jezzard, M. M. Adams, R. Turner, and L. G. Ungerleider, "Functional MRI evidence for adult motor cortex plasticity during motor skill learning," *Nature*, vol. 377, pp. 155–158, Sept. 1995.

[97] R. J. Nudo, G. W. Milliken, W. M. Jenkins, and M. e. M. Merzenich, "Use-dependent alterations of movement representations in primary motor cortex of adult squirrel monkeys," *Journal of Neuroscience*, vol. 16, no. 2, pp. 785–807, 1996.

[98] J. A. Kleim, S. Barbay, and R. J. Nudo, "Functional reorganization of the rat motor cortex following motor skill learning," *Journal of neurophysiology*, vol. 80, no. 6, pp. 3321–3325, 1998.

[99] 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*, vol. 30, no. 2, pp. 593–607, 2001.

[100] J. A. Kleim, "Cortical Synaptogenesis and Motor Map Reorganization Occur during Late, But Not Early, Phase of Motor Skill Learning," *Journal of Neuroscience*, vol. 24, pp. 628–633, Jan. 2004.

[101] K. Ganguly and J. M. Carmena, "Neural Correlates of Skill Acquisition with a Cortical Brain-Machine Interface," *Journal of Motor Behavior*, vol. 42, pp. 355–360, Oct. 2010.

[102] E. E. Fetz, "Operant conditioning of cortical unit activity," *Science*, vol. 163, no. 3870, pp. 955–958, 1969.

[103] D. J. Atkins, D. C. Heard, and W. H. Donovan, "Epidemiologic Overview of Individuals with Upper-Limb Loss and Their Reported Research Priorities.," *JPO: Journal of Prosthetics and Orthotics*, vol. 8, no. 1, pp. 2–11, 1996.

[104] E. Biddiss and T. Chau, "Upper-Limb Prosthetics: Critical Factors in Device Abandonment," *American Journal of Physical Medicine & Rehabilitation*, vol. 86, pp. 977–987, Dec. 2007.

[105] Z. Danziger, A. Fishbach, and F. Mussa-Ivaldi, "Learning Algorithms for Human-Machine Interfaces," *IEEE Transactions on Biomedical Engineering*, vol. 56, pp. 1502–1511, May 2009.

[106] M. C. Tresch and A. Jarc, "The case for and against muscle synergies," *Current Opinion in Neurobiology*, vol. 19, pp. 601–607, Dec. 2009.

[107] D. J. Berger, R. Gentner, T. Edmunds, D. K. Pai, and A. d'Avella, "Differences in Adaptation Rates after Virtual Surgeries Provide Direct Evidence for Modularity," *Journal of Neuroscience*, vol. 33, pp. 12384–12394, July 2013.

[108] A. de Rugy, G. E. Loeb, and T. J. Carroll, "Muscle Coordination Is Habitual Rather than Optimal," *Journal of Neuroscience*, vol. 32, pp. 7384–7391, May 2012.

[109] E. Burdet, R. Osu, D. W. Franklin, T. E. Milner, and M. Kawato, "The central nervous system stabilizes unstable dynamics by learning optimal impedance," *Nature*, vol. 414, pp. 446–449, Nov. 2001.

Appendix A

Appendix: Decoding algorithm derivations

A.1 Derivation of the point-process observation model

Point-process conditional intensity function

$$\lambda\left(t|H_t\right) = \frac{p(t|H_t)}{1 - \int_0^t p(u|H_u)du}$$

Let $p\left(ISI_{(0,t]}|H_t\right) = p(t|H_t)$ represent the probability of observing an inter-spike interval $(0, t]$. It may be useful to define the ISI distribution only using the conditional intensity function. This requires some algebra to evaluate $\left[1 - \int_0^t p(u|H_u)du\right]$:

$$
\begin{aligned}
\frac{d}{dt}\log\left[1-\int_0^t p(u|H_u)du\right] &= \frac{1}{1-\int_0^t p(u|H_u)du} \\
&\quad \times \frac{d}{dt}\left[1-\int_0^t p(u|H_u)du\right] \\
&= -\frac{1}{1-\int_0^t p(u|H_u)du} \\
&\quad \times \frac{d}{dt}\int_0^t p(u|H_u)du \\
&= -\frac{p(t|H_t)}{1-\int_0^t p(u|H_u)du} && \text{[Fund. thm. of calculus]} \\
&= -\lambda(t|H_t) && \text{[Using defn above]} \\
&= \frac{d}{dt}\int_0^t -\lambda(u|H_u)du && \text{[Fund. thm of calculus]} \\
\log\left[1-\int_0^t p(u|H_u)du\right] &= \int_0^t -\lambda(u|H_u)du && \text{[drop derivative]} \\
\left[1-\int_0^t p(u|H_u)du\right] &= \exp\left\{\int_0^t -\lambda(u|H_u)du\right\} && \text{[exponentiate both sides]}
\end{aligned}
$$

Thus,

$$
p(ISI_{(0,t]}|H_t) = p(t|H_t) = \lambda\left(t|H_t\right)\left[1-\int_0^t p(u|H_u)du\right] = \lambda\left(t|H_t\right)\exp\left\{\int_0^t -\lambda(u|H_u)du\right\}.
$$

Point-process joint distribution

To evaluate model likelihood it is necessary to define a joint distribution. Suppose we have spike timestamps $\{u_1, \cdots, u_n\}$. With no constraints on when timestamp u_{n+1} is observed, the probability of observing $\{u_1, \cdots, u_n\}$ is defined by the ISI distribution:

$$
\begin{aligned}
Pr\left(\{u_1, \cdots, u_n\}\right) &= \prod_{k=1}^{n} p\left(ISI_{(u_{k-1},u_k]}\middle|\, H_{u_k}\right) \\
&= \prod_{k=1}^{n} \lambda\left(u_k\middle|\, H_{u_k}\right)\exp\left\{-\int_{u_{k-1}}^{u_k} \lambda\left(v\middle|\, H_v\right)dv\right\}.
\end{aligned}
$$

It is also useful to define the probability of observing $\{u_1, \cdots, u_n\}$ in a fixed time interval $(0, T]$. To do so, a key component is to define the probability that timestamp u_{n+1} occurs

after time T:

$$\begin{aligned} Pr\left(u_{n+1} > T|\, H_T\right) &= 1 - Pr\left(u_n < u_{n+1} < T|H_T\right) \\ &= 1 - \int_{u_n}^{T} p\left(ISI_{(u_n,u]}|H_T\right) du \\ &= \exp\left\{-\int_{u_n}^{T} \lambda\left(u|\, H_T\right) du\right\}, \end{aligned}$$

where we have again made use of the algebraic relationship above defining the ISI distribution as a function of the conditional intensity function.

A.2 Generic inference on hidden Markov models using Laplace's method

Chapman-Kolmogorov equation

The generic idea for Bayesian decoders is

$$\text{posterior} \propto \text{data likelihood} \times \text{model prediction}$$

The goal is to use a model of the data likelihood $p(y_t|x_t)$ to infer the posterior $p(x_t|y^t)$ using Bayes' rule:

$$\begin{aligned} p\left(x_t|y^t\right) &= p\left(x_t|y_t, y^{t-1}\right) && \text{[expand observations]} \\ &= \frac{p\left(x_t, y_t, y^{t-1}\right)}{p\left(y_t, y^{t-1}\right)} && \text{[Bayes' rule]} \\ &= \frac{p\left(y_t|x_t, y^{t-1}\right) p\left(x_t|y^{t-1}\right) p\left(y^{t-1}\right)}{p\left(y_t|y^{t-1}\right) p\left(y^{t-1}\right)} && \text{[Bayes' rule]} \\ &= \frac{p\left(y_t|x_t, y^{t-1}\right) p\left(x_t|y^{t-1}\right)}{p\left(y_t|y^{t-1}\right)}. && \text{Cancel out } p\left(y^{t-1}\right) \end{aligned} \tag{A.2.1}$$

The distribution $p\left(x_t|y^{t-1}\right)$ represents the "model prediction" distribution, which is computed using the previous posterior and the Chapman-Kolmogorov equation:

$$\begin{aligned} p\left(x_t|y^{t-1}\right) &= \int p\left(x_t, x_{t-1}|y^{t-1}\right) dx_{t-1} && \text{[Reintroduce } x_{t-1}\text{]} \\ &= \int p\left(x_t|x_{t-1}, y^{t-1}\right) p\left(x_{t-1}|y^{t-1}\right) dx_{t-1}. && \text{[chain rule]} \end{aligned} \tag{A.2.2}$$

If a model of x_t given x_{t-1} is known (as is often the case), then the Chapman-Kolmogorov recursion calculates $p\left(x_t|y^{t-1}\right)$ for each possible x_{t-1} weighted by the posterior probability $p\left(x_{t-1}|y^{t-1}\right)$. Note that the Chapman-Kolmogorov equation is only analytically computable with simple models, e.g. Kalman filters with linear state evolution and observation models.

Inference using the Chapman-Kolmogorov equation

We denote estimates of x_t by the mean and variance of the estimator:

$$x_t|\, y^s \sim \mathcal{N}\left(\hat{x}_{t|s}, P_{t|s}\right).$$

Suppose that we have a prior estimate of the state in the form

$$p\left(x_{t-1}|y^{t-1}\right) = \mathcal{N}\left(\hat{x}_{t-1|t-1}, P_{t-1|t-1}\right).$$

With the random-walk SSM (or more generally, with any linear SSM), properties of the multivariate Gaussian make it simple to define the prediciton distribution $p\left(x_t|y^{t-1}\right)$, i.e the integral in Eq. A.2.2 above can be computed analytically. Specifically, linear transformations of Gaussian RVs remain Gaussian RVs, and it sufficies to find the mean and covariance of the distribution after transformation. In the case of the random walk model,

$$\begin{aligned} p\left(x_t|y^{t-1}\right) &= \mathcal{N}\left(\hat{x}_{t|t-1}, P_{t|t-1}\right) \\ \hat{x}_{t|t-1} &\triangleq E\left[x_t|y^{t-1}\right] = E\left[Ax_{t-1} + w_t|y^{t-1}\right] = A\hat{x}_{t-1|t-1} \\ P_{t|t-1} &\triangleq AP_{t-1|t-1}A^T + W. \end{aligned}$$

KF posterior distribution calculated using Laplace's method

To compute the posterior distribution parameters $\hat{x}_{t|t}$ and $P_{t|t}$, we employ Laplace's approximation method based on a Taylor series expansion of the posterior. In the case of the KF model, this derivation is analogous to using a sledgehammer to carve a diamond; all you're going to end up with is a million little pieces. In this case, however, that's exactly what we want. We begin by restating Eq. A.2.1 for the KF model:

$$\begin{aligned} \mathcal{N}\left(x_t; \hat{x}_{t|t}, P_{t|t}\right) &\propto \mathcal{N}\left(y_t; Cx_t, Q\right)\mathcal{N}\left(x_t; \hat{x}_{t|t-1}, P_{t|t-1}\right) \\ \log\mathcal{N}\left(x_t; \hat{x}_{t|t}, P_{t|t}\right) &= \text{const.} + \log\mathcal{N}\left(y_t; Cx_t, Q\right) + \log\mathcal{N}\left(x_t; \hat{x}_{t|t-1}, P_{t|t-1}\right) \\ -\frac{1}{2}\left[\left(x_t - \hat{x}_{t|t}\right)^T P_{t|t}^{-1}\left(x_t - \hat{x}_{t|t}\right)\right] &= \text{const.} - \frac{1}{2}\left[\left(y_t - Cx_t\right)^T Q^{-1}\left(y_t - Cx_t\right)\right] \\ &\quad - \frac{1}{2}\left[\left(x_t - \hat{x}_{t|t-1}\right)^T P_{t|t-1}^{-1}\left(x_t - \hat{x}_{t|t-1}\right)\right] \end{aligned}$$

To find $\hat{x}_{t|t}$, we begin by evaluating the derivative of both sides w.r.t. x_t:

$$\frac{\partial}{\partial x_t}\left\{-\frac{1}{2}\left[\left(x_t-\hat{x}_{t|t}\right)^T P_{t|t}^{-1}\left(x_t-\hat{x}_{t|t}\right)\right]\right\} =$$

$$= \frac{\partial}{\partial x_t}\left\{\text{const.}-\frac{1}{2}\left[\left(y_t-Cx_t\right)^T Q^{-1}\left(y_t-Cx_t\right)\right]\right\} \tag{A.2.3}$$

$$- \frac{\partial}{\partial x_t}\left\{\frac{1}{2}\left[\left(x_t-\hat{x}_{t|t-1}\right)^T P_{t|t-1}^{-1}\left(x_t-\hat{x}_{t|t-1}\right)\right]\right\} \tag{A.2.4}$$

$$-\frac{1}{2}\cdot 2P_{t|t}^{-1}\left(x_t-\hat{x}_{t|t}\right) = -\frac{1}{2}\left[-2C^TQ^{-1}\left(y_t-Cx_t\right)\right] - \frac{1}{2}\cdot 2P_{t|t-1}^{-1}\left(x_t-\hat{x}_{t|t-1}\right)$$

$$P_{t|t}^{-1}\left(x_t-\hat{x}_{t|t}\right) = P_{t|t-1}^{-1}\left(x_t-\hat{x}_{t|t-1}\right) - C^TQ^{-1}\left(y_t-Cx_t\right). \tag{A.2.5}$$

Since the derivative is true for any value of x_t, we evaluate both sides at $x_t=\hat{x}_{t|t}$ (the posterior mean) and then solve for $\hat{x}_{t|t}$. We choose this point for evaluating (A.2.5) because the left hand side is zero when evaluated at the posterior mean, i.e. we choose the posterior mean as the expansion point for simpler algebra.

$$\begin{aligned}
P_{t|t}^{-1}\left(x_t-\hat{x}_{t|t}\right)\big|_{x_t=\hat{x}_{t|t}} &= P_{t|t-1}^{-1}\left(x_t-\hat{x}_{t|t-1}\right) - C^TQ^{-1}\left(y_t-Cx_t\right)|_{x_t=\hat{x}_{t|t}} \\
0 &= P_{t|t-1}^{-1}\left(\hat{x}_{t|t}-\hat{x}_{t|t-1}\right) - C^TQ^{-1}\left(y_t-C\hat{x}_{t|t}\right) \\
&= P_{t|t-1}^{-1}\hat{x}_{t|t} - P_{t|t-1}^{-1}\hat{x}_{t|t-1} - C^TQ^{-1}y_t - C^TQ^{-1}C\hat{x}_{t|t} \\
&= \left(P_{t|t-1}^{-1}-C^TQ^{-1}C\right)\hat{x}_{t|t} - P_{t|t-1}^{-1}\hat{x}_{t|t-1} - C^TQ^{-1}y_t \\
\left(P_{t|t-1}^{-1}-C^TQ^{-1}C\right)\hat{x}_{t|t} &= \left(P_{t|t-1}^{-1}\hat{x}_{t|t-1} + C^TQ^{-1}y_t\right) \\
\hat{x}_{t|t} &= \left(P_{t|t-1}^{-1}-C^TQ^{-1}C\right)^{-1}\left(P_{t|t-1}^{-1}\hat{x}_{t|t-1} + C^TQ^{-1}y_t\right) \\
&= \left(P_{t|t-1}^{-1}-C^TQ^{-1}C\right)^{-1}P_{t|t-1}^{-1}\hat{x}_{t|t-1} \\
&\quad + \underbrace{\left(P_{t|t-1}^{-1}-C^TQ^{-1}C\right)^{-1}C^TQ^{-1}}_{K_t}y_t \\
&= \left(P_{t|t-1}^{-1}-C^TQ^{-1}C\right)^{-1}P_{t|t-1}^{-1}\hat{x}_{t|t-1} + K_ty_t
\end{aligned}$$

Making use of the matrix inversion lemma,

$$\begin{aligned}
&= \left[P_{t|t-1} - P_{t|t-1}C^T\left(Q+CP_{t|t-1}C^T\right)^{-1}CP_{t|t-1}\right]P_{t|t-1}^{-1}\hat{x}_{t|t-1} + K_ty_t \\
&= \left[I - \underbrace{P_{t|t-1}C^T\left(Q+CP_{t|t-1}C^T\right)^{-1}C}_{K_t}\right]\hat{x}_{t|t-1} + K_ty_t \\
&= \left(I-K_tC\right)\hat{x}_{t|t-1} + K_ty_t \\
&= \hat{x}_{t|t-1} + K_t\left(y_t - C\hat{x}_{t|t-1}\right).
\end{aligned}$$

The last equation is the familiar "update" equation of the KF. In the preceeding derivation, we made use of the equivalence of two different formulations of the Kalman gain:

$$\begin{aligned}
K_t &= P_{t|t-1}C^T\left(Q + CP_{t|t-1}C^T\right)^{-1} \\
&= P_{t|t-1}C^T\left[Q^{-1} - Q^{-1}C\left(P_{t|t-1}^{-1} + C^TQ^{-1}C\right)^{-1}C^TQ^{-1}\right] \\
&= \left[P_{t|t-1}C^TQ^{-1} - P_{t|t-1}C^TQ^{-1}C\left(P_{t|t-1}^{-1} + C^TQ^{-1}C\right)^{-1}C^TQ^{-1}\right] \\
&= \left[P_{t|t-1}\left(P_{t|t-1}^{-1} + C^TQ^{-1}C\right) - P_{t|t-1}C^TQ^{-1}C\right]\left(P_{t|t-1} + C^TQ^{-1}C\right)^{-1}C^TQ^{-1} \\
&= \left(P_{t|t-1} + C^TQ^{-1}C\right)^{-1}C^TQ^{-1}.
\end{aligned}$$

The corresponding calculation for the covariance matrix is much shorter. We differentiate the posterior equation again (i.e. differentiate Eq. A.2.5 with respect to x_t):

$$\begin{aligned}
\frac{\partial}{\partial x_t}\left\{P_{t|t}^{-1}\left(x_t - \hat{x}_{t|t}\right)\right\} &= \frac{\partial}{\partial x_t}\left\{P_{t|t-1}^{-1}\left(x_t - \hat{x}_{t|t-1}\right) - C^TQ^{-1}\left(y_t - Cx_t\right)\right\} \\
P_{t|t}^{-1} &= P_{t|t-1}^{-1} + C^TQ^{-1}C \\
P_{t|t} &= \left(P_{t|t-1}^{-1} + C^TQ^{-1}C\right)^{-1} \\
&= P_{t|t-1} - P_{t|t-1}C^T\left(Q + CP_{t|t-1}C^T\right)^{-1}CP_{t|t-1},
\end{aligned} \tag{A.2.6}$$

where we make the same use of the matrix inversion lemma used above to derive the standard update to the prediction covariance.

PPF posterior distribution approximated using Laplace's method

Our goal is (again) to obtain a Gaussian approximation to the posterior density $p\left(x_t|y^t\right)$, i.e. $p\left(x_t|y^t\right) = \mathcal{N}\left(x_{t|t}, P_{t|t}\right)$ to complete the recursive algorithm. Concretely, we begin with Eq.A.2.1:

$$\begin{aligned}
p\left(x_t|y^t\right) &\propto p\left(y_t|x_t, y^{t-1}\right)\cdot p\left(x_t|y^{t-1}\right) \\
\mathcal{N}\left(\hat{x}_{t|t}, P_{t|t}\right) &\propto p\left(y_t|x_t, y^{t-1}\right)\cdot\mathcal{N}\left(\hat{x}_{t|t-1}, P_{t|t-1}\right) \\
\exp\left\{-\frac{1}{2}\left(x_t - \hat{x}_{t|t}\right)^T P_{t|t}^{-1}\left(x_t - \hat{x}_{t|t}\right)\right\} &\propto p\left(y_t|x_t, y^{t-1}\right) \\
&\quad\times\exp\left\{-\frac{1}{2}\left(x_t - x_{t|t-1}\right)^T P_{t|t-1}^{-1}\left(x_t - x_{t|t-1}\right)\right\} \\
&= \prod_{k=1}^{N}\left(\left[\lambda_k\left(t|x_t, y^t\right)\Delta\right]^{y_{[k,t]}}\exp\left\{-\lambda_k\left(t|x_t, y^t\right)\Delta\right\}\right) \\
&\quad\times\exp\left\{-\frac{1}{2}\left(x_t - \hat{x}_{t|t-1}\right)^T P_{t|t-1}^{-1}\left(x_t - \hat{x}_{t|t-1}\right)\right\}
\end{aligned}$$

Applying a logarithm to both sides of the last equation above yields

$$-\frac{1}{2}\left(x_t-\hat{x}_{t|t}\right)^T P_{t|t}^{-1}\left(x_t-\hat{x}_{t|t}\right)=\text{const.}+\sum_{k=1}^{N}\left[y_{[k,t]}\log\left[\lambda_k\left(t|x_t,y^t\right)\Delta\right]+\lambda_k\left(t|x_t,y^t\right)\Delta\right] - \frac{1}{2}\left(x_t-\hat{x}_{t|t-1}\right)^T P_{t|t-1}^{-1}\left(x_t-\hat{x}_{t|t-1}\right).$$

This equation can be solved for $\hat{x}_{t|t}$ and $P_{t|t}$ by equating derivatives of both sides. Unlike the Laplace approximation used for the KF above, the key difference here is the computation advantages gained by approximating the posterior distribution around the *prior* mean, not the posterior mean. We will make use of the fact that

$$\frac{\partial}{\partial x_t}\lambda_k\left(t|x_t,y^t\right)=\frac{\lambda_k\left(t|x_t,y^t\right)}{\lambda_k\left(t|x_t,y^t\right)}\frac{\partial}{\partial x_t}\lambda_k\left(t|x_t,y^t\right)=\lambda_k\left(t|x_t,y^t\right)\frac{\partial}{\partial x_t}\left[\log\lambda_k\left(t|x_t,y^t\right)\right],$$

which is useful if the conditional intensity functions λ_k are log-linear. For conciseness, we write $\lambda_k(t)$ in place of $\lambda_k\left(t|x_t,y^{t-1}\right)$.

$$\begin{aligned}
&\frac{\partial}{\partial x_t}\left[-\frac{1}{2}\left(x_t-\hat{x}_{t|t}\right)^T P_{t|t}^{-1}\left(x_t-\hat{x}_{t|t}\right)\right]\\
&\quad=\frac{\partial}{\partial x_t}\left\{\sum_{k=1}^{N}\left(y_{[k,t]}\log\lambda_k(t)+y_{[k,t]}\log\Delta-\lambda_k(t)\Delta\right)\right\}\\
&\qquad+\frac{\partial}{\partial x_t}\left[-\frac{1}{2}\left(x_t-\hat{x}_{t|t-1}\right)^T P_{t|t-1}^{-1}\left(x_t-\hat{x}_{t|t-1}\right)\right]\\
-P_{t|t}^{-1}\left(x_t-\hat{x}_{t|t}\right)&=\frac{\partial}{\partial x_t}\left\{\sum_{k=1}^{N}\left(y_{[k,t]}\log\lambda_k(t)+y_{[k,t]}\log\Delta-\lambda_k(t)\Delta\right)\right\}\\
&\qquad+\frac{\partial}{\partial x_t}\left[-\frac{1}{2}\left(x_t-\hat{x}_{t|t-1}\right)^T P_{t|t-1}^{-1}\left(x_t-\hat{x}_{t|t-1}\right)\right]\\
&=\sum_{k=1}^{N}\frac{\partial}{\partial x_t}\left(y_{[k,t]}\log\lambda_k(t)+y_{[k,t]}\log\Delta-\lambda_k(t)\Delta\right)-P_{t|t-1}^{-1}\left(x_t-\hat{x}_{t|t-1}\right)\\
&=\sum_{k=1}^{N}\left(y_{[k,t]}\frac{\partial}{\partial x_t}\log\lambda_k(t)-\Delta\lambda_k(t)\frac{\partial}{\partial x_t}\log\lambda_k(t)\right)-P_{t|t-1}^{-1}\left(x_t-\hat{x}_{t|t-1}\right)\\
&=\sum_{k=1}^{N}\left(\frac{\partial}{\partial x_t}\log\lambda_k(t)\right)\left[y_{[k,t]}-\Delta\lambda_k(t)\right]-P_{t|t-1}^{-1}\left(x_t-\hat{x}_{t|t-1}\right).
\end{aligned}$$

We take a second derivative with respect to x_t for the second order expansion:

$$\frac{\partial}{\partial x_t}\left[-P_{t|t}^{-1}\left(x_t-\hat{x}_{t|t}\right)\right]=\frac{\partial}{\partial x_t}\left[\sum_{k=1}^{N}\left\{\left[y_{[k,t]}-\Delta\lambda_k(t)\right]\frac{\partial}{\partial x_t}\log\lambda_k(t)\right\}-P_{t|t-1}^{-1}\left(x_t-\hat{x}_{t|t-1}\right)\right]$$

$$\begin{aligned}
-P_{t|t}^{-1} &= \sum_{k=1}^{N}\frac{\partial}{\partial x_t}\left(\frac{\partial}{\partial x_t}\log\lambda_k(t)\right)\left[y_{[k,t]}-\Delta\lambda_k(t)\right]-P_{t|t-1}^{-1}\\
&= \sum_{k=1}^{N}\left(\frac{\partial^2}{\partial x_t\partial x_t^T}\log\lambda_k(t)\right)\left[y_{[k,t]}-\Delta\lambda_k(t)\right]\\
&\quad -\sum_{k=1}^{N}\left(\frac{\partial}{\partial x_t}\log\lambda_k(t)\right)\Delta\frac{\partial}{\partial x_t}\lambda_k(t)-P_{t|t-1}^{-1} \qquad \text{(A.2.7)}\\
&= \sum_{k=1}^{N}\left(\frac{\partial^2}{\partial x_t\partial x_t^T}\log\lambda_k(t)\right)\left[y_{[k,t]}-\Delta\lambda_k(t)\right] \qquad \text{(A.2.8)}\\
&\quad -\sum_{k=1}^{N}\left(\frac{\partial}{\partial x_t}\log\lambda_k(t)\right)\Delta\lambda_k(t)\left(\frac{\partial}{\partial x_t}\log\lambda_k(t)\right)-P_{t|t-1}^{-1} \qquad \text{(A.2.9)}
\end{aligned}$$

The first equation can be evaluated at the prior mean $x_t=\hat{x}_{t|t-1}$ to get

$$\begin{aligned}
-P_{t|t}^{-1}\left(x_t-\hat{x}_{t|t}\right)\Big|_{x_t=\hat{x}_{t|t-1}} &= \left(\frac{\partial}{\partial x_t}\log\lambda_t\right)\left[y_t-\Delta\lambda_t\right]-P_{t|t-1}^{-1}\left(x_t-\hat{x}_{t|t-1}\right)\Big|_{x_t=\hat{x}_{t|t-1}}\\
-P_{t|t}^{-1}\left(\hat{x}_{t|t-1}-\hat{x}_{t|t}\right) &= \left(\frac{\partial}{\partial x_t}\log\lambda_t\right)\left[y_t-\Delta\lambda_t\right]\\
\hat{x}_{t|t} &= \hat{x}_{t|t-1}+P_{t|t}^{-1}\left(\frac{\partial}{\partial x_t}\log\lambda_t\right)\left[y_t-\Delta\lambda_t\right] \qquad \text{(A.2.10)}
\end{aligned}$$

PPF for independent units with no history dependence

Furthermore, the second derivatives of the log conditional intensity function are zero, so (A.2.8) simplifies to

$$-P_{t|t}^{-1} = -\sum_{k=1}^{N}\left(\frac{\partial}{\partial x_t}\log\lambda_k(t)\right)\Delta\lambda_k(t)\left(\frac{\partial}{\partial x_t}\log\lambda_k(t)\right) - P_{t|t-1}^{-1}$$

$$P_{t|t}^{-1} = \sum_{k=1}^{N} C_{[k,:]}\Delta\lambda_k(t)C_{[k,:]}^T + P_{t|t-1}^{-1}$$

$$= \begin{bmatrix} C_{[0,:]}^T & \cdots & C_{[N,:]}^T \end{bmatrix}\begin{bmatrix} \Delta\lambda_0(t) & & \\ & \ddots & \\ & & \Delta\lambda_N(t) \end{bmatrix}\begin{bmatrix} C_{[0,:]} \\ \vdots \\ C_{[N,:]} \end{bmatrix} + P_{t|t-1}^{-1} \tag{A.2.11}$$

$$= \begin{bmatrix} C_{[0,:]}^T & \cdots & C_{[N,:]}^T \end{bmatrix}\underbrace{\begin{bmatrix} \Delta\exp\left(C_{[0,:]}x_t\right) & & \\ & \ddots & \\ & & \Delta\exp\left(C_{[N,:]}x_t\right) \end{bmatrix}}_{Q^{-1}}\begin{bmatrix} C_{[0,:]} \\ \vdots \\ C_{[N,:]} \end{bmatrix} + P_{t|t-1}^{-1}$$

$$= \begin{bmatrix} C_{[0,:]}^T & \cdots & C_{[N,:]}^T \end{bmatrix}\underbrace{diag\left(\exp\left(Cx_t\right)\right)}_{Q^{-1}}\begin{bmatrix} C_{[0,:]} \\ \vdots \\ C_{[N,:]} \end{bmatrix} + P_{t|t-1}^{-1} \tag{A.2.12}$$

$$P_{t|t}^{-1} = C^TQ^{-1}C + P_{t|t-1}^{-1}$$

$$P_{t|t} = \left(C^TQ^{-1}C + P_{t|t-1}^{-1}\right)^{-1} \tag{A.2.13}$$

$$= P_{t|t-1} - P_{t|t-1}C^T\left(Q + CP_{t|t-1}C^T\right)^{-1}CP_{t|t-1} \tag{A.2.14}$$

A.3 Rate-matching between the KF and PPF state-space models

Suppose that our KF SSM matrices are

$$A = \begin{bmatrix} 1 & 0 & \Delta_{KF} & 0 & 0 \\ 0 & 1 & 0 & \Delta_{KF} & 0 \\ 0 & 0 & a_{KF} & 0 & 0 \\ 0 & 0 & 0 & a_{KF} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad W = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & w_{KF} & 0 & 0 \\ 0 & 0 & 0 & w_{KF} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

To construct a PPF equivalent state space model, obviously the same structure needs to be followed but $a_{PPF} \neq a_{KF}$ because $\Delta_{PPF} \ll \Delta_{KF}$. To find the corresponding parameters

a_{PPF} and w_{PPF} we note that the state-space models are equivalent if

$$\begin{aligned}
w_{KF}\Delta_{PPF} &= \Delta_{KF} w_{PPF} \\
a_{PPF}^{n\cdot\frac{\Delta_{KF}}{\Delta_{PPF}}} &= a_{KF}^{n} \\
n \cdot \frac{\Delta_{KF}}{\Delta_{PPF}} \log a_{PPF} &= n \cdot \log a_{KF} \\
\frac{\Delta_{KF}}{\Delta_{PPF}} \log a_{PPF} &= \log a_{KF} \\
\log a_{PPF} &= \frac{\Delta_{PPF}}{\Delta_{KF}} \log a_{KF} \\
a_{PPF} &= \exp\left(\frac{\Delta_{PPF}}{\Delta_{KF}} \log a_{KF}\right) \\
&= \exp\left(\log a_{KF}^{\frac{\Delta_{PPF}}{\Delta_{KF}}}\right) \\
&= a_{KF}^{\frac{\Delta_{PPF}}{\Delta_{KF}}}
\end{aligned}$$

The formulas for a and w are different because a acts multiplicatively while w acts additively.