Designing plant dynamics to optimize task-specific closed-loop performance of brain-machine interfaces

Suraj Gowda

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2013-41
http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-41.html

May 1, 2013

Acknowledgement

This work would not have been possible without other members of the Camena lab. I performed all computational analyses of data from subject S and subject J, with great feedback from Amy Orsborn and Jose Carmena. Both subjects were trained by Amy Orsborn, with assistance from Helene Moorman and Simon Overduin, and Dragan Dimitrov provided surgical assistance for both surgeries. The conception of novel mathematical methods presented here are largely my own contribution, with excellent feedback from Siddharth Dangi, Amy Orsborn and Jose Carmena. This manuscript was edited by myself, Amy Orsborn, Simon Overduin, and Jose Carmena, who I thank for assisting me in making this report coherent.

Designing plant dynamics to optimize task-specific closed-loop performance of brain-machine interfaces

December 14, 2012

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Jose Carmena, Research Advisor

Date

Professor Michel Maharbiz, Second Reader

Date

Author Contribution and Acknowledgements

This work would not have been possible without other members of the Camena lab. I performed all computational analyses of data from subject S and subject J, with great feedback from Amy Orsborn and Jose Carmena. Both subjects were trained by Amy Orsborn, with assistance from Helene Moorman and Simon Overduin, and Dragan Dimitrov provided surgical assistance for both surgeries. The conception of novel mathematical methods presented here are largely my own contribution, with excellent feedback from Siddharth Dangi, Amy Orsborn and Jose Carmena. This manuscript was edited by myself, Amy Orsborn, Simon Overduin, and Jose Carmena, who I thank for assisting me in making this report coherent.

Abstract

Closed-loop brain-machine interface (BMI) systems are dynamical systems whose plant properties ultimately influence controllability. For instance, a 2D cursor in which velocity is controlled using a Kalman filter (KF) will, by default, model a correlation between horizontal and vertical velocity. In closed-loop, this translates to a "curling" dynamical effect, and such an effect is unlikely to be beneficial for BMI performance. However, there have been few empirical studies exploring how various dynamical effects of closed-loop BMIs ultimately influence performance. In this work, we utilize experimental data from a monkey subject performing closed-loop control of a 2D cursor BMI and show that the presence of certain dynamical properties correlate with performance loss. We also show that other dynamical properties represent tradeoffs between naturally competing objectives, such as speed versus accuracy. These empirical findings demonstrate the need to eliminate detrimental dynamics by accounting for the feedback control strategy employed by the user, as well as the need to fine-tune plant dynamics to optimize task-specific performance tradeoffs. To this end, we develop general-purpose tools for designing BMI plant dynamics. Using these tools, we show two different ways to improve accuracy and hold performance at the expense of speed. In a closed-loop BMI simulator, the two accuracy-optimized decoders, the symmetrically dampened (SD) velocity KF (SDVKF) and velocity linear system (SDVLS), had significantly better reaching accuracy than standard KF variants. In a closed-loop experiment, a monkey subject demonstrated significantly better holding performance and more accurate reaches at the cost of slightly slower reaches when operating the SDVKF than when operating the standard position/velocity KF. Thus, BMI design can be improved by using parameter estimation techniques that craft the BMI plant to both user and task.

1 Introduction

Brain-machine interfaces (BMIs) have tremendous potential to restore motor function impaired by spinal cord injuries and other neurological disorders. BMIs drive artificial actuators using volitional neural activity to bypass damaged neural circuitry. Numerous experimental demonstrations in rodents [1, 2, 3], monkeys [4, 5, 6, 7, 8, 9, 10, 11, 12, 13] and humans [14, 15, 16] have together provided a compelling proof of concept. However, before BMIs are viable as a clinical treatment option for humans, significant performance improvements are required.

Recent work suggests that BMI performance improvements may require a deeper understanding of BMIs as closed-loop systems, in which brain and machine interact with one another [17, 18]. In closed-loop control, subjects attempt to control the state of the BMI prosthesis using real-time feedback. Both the user's feedback control strategy and the BMI decoder contribute to performance because the user can compensate for decoder errors by volitionally altering neural activity. Abstractly, the most controllable BMI system will be one in which feedback corrections are easiest.

Previous BMI studies have made significant progress in understanding and designing closed-loop BMI systems. Closed-loop experiments have shown that BMI controllability depends in part on which kinematic variables are controlled by the user and decoded by the BMI [6, 15, 17]. Furthermore, BMI subjects can adapt to rotation effects intentionally introduced into the decoder by the experimenter [19, 20, 21]. These experiments highlight the importance of the user's feedback control strategy as these decoders are controllable with closed-loop feedback but have poor offline reconstruction accuracy. Similarly, BMI subjects can control arbitrary linear decoders through learning because the tuning patterns of neurons can change over several days and adapt to a fixed mapping [11]. Additionally, many BMIs have utilized closed-loop decoder adaptation (CLDA), in which decoder parameters are altered while the subject continues to operate the BMI [2, 5, 17, 18, 22, 23, 24], as a practical method to maximize closed-loop performance. CLDA can rapidly create high-performance decoders even when little is known *a priori* about the tuning properties of the BMI neural ensemble [18].

This work centers on the relationship between BMI plant dynamics, which are the properties of closed-loop BMI systems, and task performance. BMI plant dynamics define the transformation of neural activity into movement and thus have an important impact on performance. Just as changes in arm biomechanics (changes in external load, dynamic perturbations, etc.) affect arm movements, changes in BMI plant dynamics affect BMI movements. Thus, it is worthwhile to understand precisely how changes in plant dynamics affect closed-loop performance. To this end, we apply control theory to determine which plant properties are detrimental to controllability, use experimental data to verify an empirical correlation between plant dynamics and performance, and utilize closed-loop simulations and experiments to demonstrate that alterations to the BMI plant cause predictable changes to performance.

We explored the impact of plant dynamics in the context of recursive linear BMI decoding algorithms, specifically the Kalman filter (KF) algorithm. A BMI decoding algorithm is recursive if it decodes the BMI state using previously decoded outputs and the most recent neural input. The KF is a linear prediction algorithm commonly used in BMIs, and previous demonstrations of closed-loop BMIs have used the KF to control a cursor's velocity (VKF)

[15, 17], position/velocity (PVKF) [25, 18], and position/velocity/acceleration [22]. Non-recursive decoding algorithms are a subclass of recursive algorithms that do not use past decoder outputs. Perhaps for this reason, the recursive KF was shown to be more controllable than the non-recursive linear Wiener filter [15]. Here, we focus our analysis entirely on the plant dynamics of recursive linear BMI decoders.

We apply optimal control theory to explain why closed-loop plant dynamics unexpected by the BMI user are detrimental to reaching accuracy and target hold performance. It is worth clarifying how "unexpected" plant dynamics might arise. For instance, the default modeling assumption of the KF is that all kinematic variables are correlated. A specific example is provided by the PVKF and the VKF, which both model correlations between horizontal and vertical velocity. This modeling assumption generates the property that a non-zero vertical speed will increase horizontal speed and vice versa. Interactions between horizontal and vertical velocity translate to closed-loop curling dynamics, which are similar to curl force fields applied as external perturbations to arm movements. In the PVKF, similar interactions between other kinematic variables can cause the cursor's position to drift without changes to the cursor's velocity. These unpredictable dynamical effects are intuitively detrimental to BMI controllability because they are unlikely to conform to the BMI user's control strategy. To warrant this intuitive argument, we use optimal control theory to demonstrate why unexpected dynamics are problematic even for closed-loop control of a noiseless system.

To understand the empirical relationship between plant properties and performance, we analyzed a closed-loop BMI experiment in which one monkey subject operated a PVKF. The PVKF contains a rich set of dynamical properties which changed across sessions and created diverse kinematic variable interactions. The presence of several plant properties, including the curling dynamics and position drift we described above, corresponded with a decline in reach accuracy and hold performance. However, some plant properties improved reaching speed at the expense of accuracy. These findings suggest that some plant dynamics should be entirely eliminated whereas others should be fine-tuned by task and user, to balance competing objectives like speed and accuracy. Consequently, we devise two novel methods for precisely specifying BMI plant properties during the parameter estimation process. These methods were used in simulations and closed-loop experiments to demonstrate that we can affect performance in predictable ways by manipulating plant dynamics.

2 Methods

2.1 Electrophysiology

Two adult male rhesus macaques (*Macaca mulatta*), subject S and subject J, were used in this study. The subjects were chronically implanted with microwire electrode arrays for neural recording. One array of 128 teflon-coated tungsten electrodes ($35\mu m$ diameter, $500\mu m$ wire spacing, 8×16 array configuration; Innovative Neurophysiology, Durham, NC) was implanted in each brain hemisphere. Arrays were implanted targeting the arm areas of primary motor cortex (M1) and dorsal premotor cortex (PMd). Single and multi-unit activity was recorded using a 128-channel MAP system and sorted online using Sort Client (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.

2.2 Task

Subject 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 subjects performed closed-loop control of the cursor using a KF BMI without overt arm movements. Figure 1 shows an illustration of the task setup and trial timeline. The subjects initiated trials by moving the cursor to the center and holding for 400 (250) ms. The subjects 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 subjects were cued (the center turned off and the peripheral target turned on) 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 they effectively resulted in no penalty to the subject. 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. We analyzed experimental data from subject S to determine how BMI plant properties affected performance and validated those conclusions using experimental data from subject J.

2.3 BMI using the Kalman filter

Variants of the KF have been used in several online BMI experiments [15, 17, 18, 22, 26]. In the PVKF, the x_t represents the cursor position and velocity:

$$x_t = \begin{bmatrix} \text{horiz. pos.} \\ \text{vert. pos.} \\ \text{horiz. vel.} \\ \text{vert. vel.} \\ \text{offset} \end{bmatrix} = \begin{bmatrix} p_x(t) \\ p_y(t) \\ v_x(t) \\ v_y(t) \\ 1 \end{bmatrix} = \begin{bmatrix} p(t) \\ v(t) \\ 1 \end{bmatrix}.$$

When it helps to clarify mathematical expressions, we will switch between the notation x_t and $x(t)$. The KF provides a way to estimate the hidden state x_t when the state follows the Gaussian process

$$x(t+1) = Ax(t) + w(t);\ w(t) \sim \mathcal{N}(0, W),$$

and when observations y_t related to the state are available. The observations y_t represent the observed spikes from a neural ensemble in the past 100 ms. The KF models x_t and y_t as jointly Gaussian with the relationship

$$y(t) = Cx(t) + q(t);\ q(t) \sim \mathcal{N}(0, Q).$$

A and W define the "state space" model while C and Q constitute the neural firing model. This model of neural firing assumes that spikes are linearly related to cursor kinematics. q_t represents "neural noise", i.e. neural firing covariance not captured by the linear model. $\hat{x}(t)$ is the minimum mean squared error estimate of $x(t)$, which is both linear and recursive under the KF model:

$$\hat{x}(t) = (I - K_t C)A\hat{x}(t-1) + K_t y(t). \tag{1}$$

The KF algorithm provides a recursive way to estimate $x(t)$, using the previous estimate $\hat{x}(t-1)$ and the most recent observation y_t. The Kalman gain K_t is an optimal linear mapping which combines the observation vector with the previous state estimate. The decoder is updated as frequently as new observations of neural activity can be made (every 100 ms in our case). We present the Kalman gain derivation in A.1 and the maximum-likelihood estimators (MLEs) for the parameters A, W, C and Q in A.2.

2.4 SmoothBatch Closed-loop Decoder Adaptation (SB-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. The goal of CLDA is to alter decoder parameters so that the decoder's output accurately reflect the subject's intended movements. CLDA methods utilize an error signal to determine how decoder parameters should be updated. This error estimate can formed using binary evaluations of trial success [24], non-causal Bayesian inference [22], or as we use here, knowledge of task goals [5, 17, 18]. CLDA methods have been utilized in population vector decoders [5], auto-regressive moving average decoders [23], neural network decoders [24], and KF decoders [17, 18, 22, 2].

To estimate the error signal using knowledge of the task, we assume that (1) the task always specifies a target to acquire and (2) the subject is always trying to acquire that target. These assumptions can be used to estimate the subject's *intended kinematics* each time the decoder makes a prediction. With this knowledge, we can estimate decoder error and apply methods from adaptive filter theory to update the decoder parameters while the subject continues to operate the BMI [27].

Previous work showed that CLDA methods are particularly useful when prior knowledge about the relationship between neural firing and BMI kinematics is limited [18]. This might make CLDA useful in a clinical setting in which overt contralateral arm movements are impossible. When initial decoder performance was poor, the best KF parameter update strategy

was to recalculate a new parameter setting for C and Q every 1-2 minutes and smoothly average with previous estimates [18]. This specific CLDA method, named SmoothBatch, was the training method used for all the decoders analyzed in this work. Without contralateral arm movements, C and Q were initially estimated using neural activity while the subject (1) passively watched a cursor making artificially generated center-out movements (visual feedback), (2) performed the task with ipsilateral arm movements, or (3) sat quietly in his chair with no task. In addition, we (4) generated arbitrary initial decoders by shuffling C and Q matrices estimated during previous sessions. The SmoothBatch CLDA (SB-CLDA) algorithm was a robust method for quickly improving the performance of decoders seeded from these adverse conditions.

Parameter settings were updated by the equations

$$\begin{aligned} C^{(i+1)} &= \alpha C^{(i)} + (1-\alpha)\,\hat{C} \\ Q^{(i+1)} &= \alpha Q^{(i)} + (1-\alpha)\,\hat{Q}, \end{aligned}$$

where $C^{(i+1)}$ is the updated parameter setting, $\hat{C}$ is an estimate of C from the newest "batch" of data, and $C^{(i)}$ is the previous parameter setting; Q quantities are defined similarly [18]. The parameter $\alpha \in [0, 1]$ determines the rate at which parameters change. MLEs are used to calculate $\hat{C}$ and $\hat{Q}$ by modeling a linear relationship between our estimate of the subject's intended kinematics and neural spiking (A.2). A and W represent a linear model of hand endpoint kinematics over time, which can be estimated independently of C and Q. Thus, they are independent of the specific neural ensemble available to the decoder and we did not retrain them using SB-CLDA. SB-CLDA was usually stopped after the rate of successful trials exceeded 10 trials per minute. The number of updates performed is defined by the "training time" (typically 20-30 minutes) and the frequency of updates (every 1-2 minutes).

Figure 2 illustrates how SB-CLDA was applied in experimental sessions for both subjects. Decoders were initialized using one of the four conditions discussed above. We adapted decoder parameters with SB-CLDA until performance was adequate, after which the subjects performed the BMI task with a fixed-parameter decoder. Usually the decoder parameters were fixed to the C and Q last estimated by SB-CLDA (60 sessions). In 11 sessions, the fixed-parameter decoders first modified A or the last parameter setting for C by (1) altering the velocity-related terms in A, (2) rescaling velocity-relevant terms in C, or (3) removing units from the decoder. Finally, in 11 sessions, SB-CLDA was resumed to further improve closed-loop performance, but only in cases in which we did modify A or C. For subject J, no perturbations to A or C were applied to the final SB-CLDA parameter settings.

2.5 Performance measures

We quantified BMI task performance as well as the accuracy with which the subjects performed the task. Upon successfully initiating a trial (via center-hold), 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 subjects were able to reach the peripheral target, including both successful trials and trials which ended in peripheral hold errors. BMI task performance was assessed using 2 metrics:

- Hold error rate: The frequency of peripheral-target hold errors per successful trial. This quantity can exceed 1, e.g. 2 hold errors per successful trial corresponds to a hold error rate of 2. Hold errors had a relatively strict penalty in that the subject was required to return to the center before re-attempting the target hold.

- Reach time: The time elapsed between leaving the center and entering the peripheral target.

In addition, we utilized 4 trial-by-trial measures of reach accuracy along the task axis, the straight line between the center and peripheral target p_{target}:

- Movement error (ME): The average of the perpendicular distance between the cursor trajectory and the task axis at each time-point.

- Movement variability (MV): The standard deviation of the perpendicular distance between cursor trajectory and the task axis.

- Effective control deviation (ECD): The angle between $p_{t+1} - p_t$ and $p_{target} - p_t$.

- Velocity control deviation (VCD): The angle between the velocity control input (defined more rigorously below) and $p_{target} - p_t$ (see Figure 3).

There were no explicit accuracy requirements to acquire the reward. For calculations of accuracy performance, we discarded trials in which the cursor touched one of the other targets. These trials corresponded to a small percentage of highly inaccurate trials. Lower values correspond to better performance for all of these performance measures. ME and MV have been used previously to characterize the accuracy of 2D BMIs [15, 18] as well as generic pointing devices (e.g. computer mice) [28].

It is worth noting that low VCD is not strictly required to generate accurate movements. Consider a decoder in which all the decoded movements are rotated 90 degrees. In this scenario, the subject may use control inputs rotated 90 degrees away from the intended movement direction (high VCD), but these inputs may generate accurate end effector movements (low ECD).

We analyzed how the mean and standard deviation of these performance measures varied with changes in decoder parameters across sessions. The number of initiated trials also varied across sessions because the task was self-paced and the duration of experimental blocks varied. For statistical comparisons of performance across sessions, we weighted each target-in trial equally.

2.6 Generalizing from the KF to other linear BMI algorithms

Linear BMIs, including the KF, produce an estimate of the kinematic state $\hat{x}(t)$ using the previous estimate $\hat{x}(t-1)$ and the current spike observations y_t:

$$\hat{x}(t) = \bar{A}_t \hat{x}(t-1) + \bar{B}_t y(t). \tag{2}$$

$\bar{A}_t$ is the state transition matrix of the system, and $\bar{B}_t$ is the control input matrix. In the case of the KF, $\bar{A}_t$ and $\bar{B}_t$ are based only on the model parameters, the time index t, and the

uncertainty (covariance) of the initial state x_0; they are independent of any observations. In the case of the KF, $\bar{A}_t = (I - K_t C)A$ and $\bar{B}_t = K_t$.

It is important to note that $\bar{A}_t$ is not the same as A. By default, the KF assumes that all variables are jointly Gaussian and thus correlated. Due to the Kalman gain present in the equation for $\bar{A}_t$ above, all of the parameters $\{A, W, C, Q\}$ contribute to $\bar{A}_t$. Thus, $\bar{A}_t$ depends on A but changes to the neural firing model (C and Q) will cause it to change.

In stable KFs with a time-invariant model, $\bar{A}_t$ and $\bar{B}_t$ converge to $\bar{A}$ and $\bar{B}$, respectively. This "steady-state" Kalman filter has been used in offline predictions in place of the standard KF to reduce computational overhead [29]. Experimental decoders converged to within machine precision in seconds. All the BMI experimental blocks we analyzed in this study were substantially longer than the steady-state convergence time, so we analyzed KF decoders as time-invariant systems.

Casting the PVKF decoder as the linear dynamical system (LDS) in Eq. 2 allows us to analyze how plant dynamics may impact control. In the BMI decoding context, *dynamics* are not forces or motion. Rather, dynamics can be any property of a dynamical system such as Eq. 2. Formulating the KF as an LDS is slightly different from the traditional presentation of the KF [26], and it enables us to analyze the decoder dynamics directly relevant to closed-loop control.

For 2D linear position/velocity BMIs, we can further sub-divide $\bar{A}$ and $\bar{B}$ into components related to position and velocity:

$$\begin{bmatrix} p_x(t) \\ p_y(t) \\ v_x(t) \\ v_y(t) \\ 1 \end{bmatrix} = \begin{bmatrix} t_{xx} & t_{xy} & s_{xx} & s_{xy} & \bar{p}_x \\ t_{yx} & t_{yy} & s_{yx} & s_{yy} & \bar{p}_y \\ m_{xx} & m_{xy} & n_{xx} & n_{xy} & \bar{v}_x \\ m_{yx} & m_{yy} & n_{yx} & n_{yy} & \bar{v}_y \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x(t-1) \\ p_y(t-1) \\ v_x(t-1) \\ v_y(t-1) \\ 1 \end{bmatrix} + \bar{B}y(t) \tag{3}$$

$$\begin{bmatrix} p(t) \\ v(t) \\ 1 \end{bmatrix} = \begin{bmatrix} T & S & \bar{p} \\ M & N & \bar{v} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p(t-1) \\ v(t-1) \\ 1 \end{bmatrix} + \begin{bmatrix} \bar{B}_{[0:1,:]} \\ \bar{B}_{[2:3,:]} \\ 1 \end{bmatrix} y(t). \tag{4}$$

When indexing matrices, we use the convention that $z[0,0]$ refers to the top-left element of matrix z. We also will utilize the MATLAB sub-matrix indexing notation, in which $0{:}n = [0, 1, 2, \cdots, n]$, so that $z_{[0:1,:]}$ refers to rows 0 and 1 of matrix z. Figure 3 shows a system diagram of the PVKF. In Eq. 4, T and M determine position-dependent dynamics while S and N determine velocity-dependent dynamics. We refer to $\bar{B}_{[0:1,:]}$ as $\bar{B}_{pos}$ because it maps spike observations into a position control input; similarly $\bar{B}_{vel} = \bar{B}_{[2:3,:]}$ maps spike observations into a velocity control input.

The LDS representation of the BMI decoder shows that the plant has properties which represent "sensitivity". It will be useful in our analyses to roughly capture the "size" of certain plant features using matrix norms, e.g. $\|\cdot\|_2$ is the matrix ℓ_2 norm and $\|\cdot\|_F$ is the Frobenius norm. The magnitude of $\bar{B}_{vel}$ and $\bar{B}_{pos}$, i.e. $\left\|\bar{B}_{pos}\right\|_2$ and $\left\|\bar{B}_{vel}\right\|_2$, define the size of the kinematic control input that can be generated by a single set of spike observations y_t. Larger values for $\left\|\bar{B}_{pos}\right\|_2$ and $\left\|\bar{B}_{vel}\right\|_2$ make the velocity control and position control more sensitive to each individual spike.

The duration of time that a velocity control input $\bar{B}_{vel}y_t$ influences the system state x_t depends on N. For example, $N = 0$ allows a control input to be active for exactly 1 system iteration, $\|N\|_2 \geq 1$ creates an unstable system in which control inputs affect the system state forever, and $\|N\|_2 < 1$ produces a smoothing dynamic which causes the influence of a control input to slowly decay over multiple time steps. In this last case, which represents all SB-CLDA trained decoders, how long a control input remains relevant to the system state depends on the magnitude of N. Consequently, in analyses of plant dynamics we will say that the plant has a "control memory", represented by N and quantified by $\|N\|_2$.

The relationship between $p(t)$ and $v(t-1)$ is defined by S, which is a scaling factor applied to the cursor's velocity before it is integrated to update the position. We demonstrate in the Results that for the PVKF, applying S to the cursor's velocity is not the same as multiplying by dt. Larger values for S, i.e. larger $\|S\|_2$, mean that the same velocity can cause larger changes in the cursor's position. S therefore represents a third mechanism in which sensitivity to velocity control can be specified.

Every linear BMI decoder can be written in the form of Eq. 2. If we let y_t represent spikes from the current observation as well as a fixed history length, then the Wiener filter can be written simply as $x(t) = \bar{B}_{WF}y(t)$ in which $\bar{B}_{WF}$ is the Wiener filter matrix [6, 12, 30].

The relationship between the VKF and the PVKF is easily observed from the dynamical systems perspective. In the VKF, the state $x_t = [v_x(t), v_y(t), 1]^T$ contains only the velocity components. Position estimates are generated by "integrating" the decoded velocity: $\hat{p}_t = dt \cdot \hat{v}_t + \hat{p}_{t-1}$, where dt is the rate at which new decoder outputs are generated. For the VKF, we can write

$$\begin{bmatrix} p(t) \\ v(t) \\ 1 \end{bmatrix} = \begin{bmatrix} I_2 & dt \cdot I_2 & 0 \\ 0 & N & \bar{v} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p(t-1) \\ v(t-1) \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ \bar{B}_{vel} \\ 1 \end{bmatrix} y(t). \tag{5}$$

The steady-state VKF is simply a special case of the steady-state PVKF. Position dynamics and position control are *degenerate* in the case of the VKF because the control and dynamics of the system are purely controlled by the velocity terms. To quantify the extent to which position dynamics are *not* degenerate, we will measure the quantity $\|T - I_2\|_2$.

M defines position-dependent velocity offsets, which are potentially not uniform over the workspace. Noting that $v_{t+1} = Nv_t + Mp_t + \bar{v}$, the velocity offset at the position p_t is given by $Mp_t + \bar{v}$. Along each of the 8 task axes, this offset can be calculated as $Mp_t + \bar{v} = (1 - \lambda)Mp_{center} + \lambda Mp_{target} + \bar{v}$, where λ is the fraction of the distance between the center and the target already covered by the cursor ($\lambda \in [0, 1]$ can represent any point on the task axis). As the cursor moves from the center to the target, the velocity offset changes linearly between the velocity offset due to p_{center} and that due to p_{target}. Mp_{target} may be different for each target, which creates target-specific velocity offsets. Every task axis has the same origin, so for each target we measure $v_{\text{offset}}(target)$, the vector projection of $Mp_{target} + \bar{v}$ onto the task axis $p_{target} - p_{center}$. The scalar quantity $v_{\text{offset}}(target)$ represents the magnitude of the velocity offset at to each target, a quantity that was useful assessing the impact of M on task performance.

In general, the properties of S and N need not be symmetric with respect to horizontal and vertical velocity terms. We quantified both the asymmetry and cross-coupling

of velocity dynamics by calculating the distance between N and the set of scalar matrices. Scalar matrices are matrices of the form λI, where $\lambda \in \mathbb{R}$. As a matrix projection, $min_n \|N - nI_2\|_F$ is the size of the smallest change to N necessary to make it a scalar matrix. We note that $n^* = \arg\min_n \|N - nI_2\|_F = \frac{1}{2}(n_{xx} + n_{yy})$, which we use to define $dist(N, nI_2) = \left\|N - \frac{1}{2}(n_{xx} + n_{yy})\, I_2\right\|_F$ [31]. Alternatively, we can measure $\Delta n = |n_{xx} - n_{yy}| = 2\,|n_{xx} - n^*|$, which measures the difference between the diagonal elements of N. Both Δn and $dist(N, nI_2)$ quantify the distance between N and the scalar matrix set, but $dist(N, nI_2)$ depends on n_{xy} and n_{yx} whereas Δn does not. Both distance measures were useful in assessing the impact of asymmetric plant properties on task performance.

2.7 Simulating BMI optimal control

Simulations of optimal control provide useful intuition about BMI plant dynamics. We simulated center-out reaches by generating control inputs to the BMI plant using an optimal linear feedback controller. Suppose that our BMI linear dynamical system is given by

$$x_t = \underbrace{\begin{bmatrix} I_2 & dt \cdot I_2 & 0 \\ 0 & nI_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\bar{A}} x_{t-1} + \underbrace{\begin{bmatrix} 0 \\ I_2 \\ 0 \end{bmatrix}}_{\bar{B}} y_t,$$

where y_t in this context represents a two-dimensional velocity control signal. The control signal y_t was designed to drive the cursor from the initial state $x_0 = [0, 0, 0, 0, 1]^T$ (zero velocity at the center of the workspace) to the target state $x_{targ} = \left[p_x^*, p_y^*, 0, 0, 1\right]^T$ (zero velocity at the peripheral target). The control inputs were designed to minimize the cost function:

$$cost = \sum_{t=1}^{T} (x_t - x_{targ})^T Z (x_t - x_{targ}) + y_t^T R y_t. \tag{6}$$

The theory of linear-quadratic-Gaussian (LQG) control allows us to find the optimal feedback controller L if know $\bar{A}$ and $\bar{B}$ [32]. Thus, $\bar{A}$ is analogous to the BMI subject's "internal model" of cursor dynamics.

After L was found, control sequences were executed in both open-loop and closed-loop. The feedback controller generates the control input sequence $y_t = Lx_t$. The source of x_t used to generate the input y_t is what differentiates open- versus closed-loop execution. Open-loop execution uses x_t^{OL} pre-computed using the model parameters:

$$x_t^{OL} = \bar{A}^t x_0 + \sum_{k=1}^{t} \bar{A}_{\text{internal model}}^{t-k} \cdot \bar{B} y_k,$$

where $\bar{A}_{\text{internal model}}$ is the approximation of $\bar{A}$ used to compute L. Thus, $y_t^{OL} = Lx_t^{OL}$. In general, $x_t \neq x_t^{OL}$ due to control noise and/or mismatch between $\bar{A}$ and $\bar{A}_{\text{internal model}}$. In contrast, closed-loop operation uses the true x_t generated by the system, i.e. $y_t^{CL} = Lx_t$. Thus, closed-loop control is generally more robust to noise and modeling errors.

2.8 Simulating SB-CLDA

To evaluate new decoding algorithms, we simulated closed-loop BMI and compared simulation performance. Comparisons of simulated performance are preferable to comparisons of offline reconstruction accuracy because experimental evidence suggests that the offline accuracy of linear decoders often does not translate to good closed-loop control [20, 33]. This is perhaps due to the inherent feedback differences between BMI control during which the subject only has visual feedback, unlike arm control during which proprioceptive feedback is also available. Furthermore, BMIs require the brain to solve a control problem that is different from the problem of controlling the natural arm because (1) the dynamics of the BMI plant are different from arm dynamics and (2) the BMI is controlled using a different neural pathway, with fewer neurons directly connected to the plant, than the natural arm control mechanism. Thus, we use simulations to compare the performance of different decoding algorithms.

We simulated a BMI feedback control policy (Figure 3A) using simulated neurons tuned to the "intended" direction of movement, which was always toward the target. The simulated subject observed the current position of the cursor and calculated a velocity that would direct it toward the target. The intended velocity direction was corrupted by zero-mean additive white Gaussian noise bounded in the range $(-\pi, \pi]$. The noise variance was set to 0.13 rad^2 based on the angular error variance of subject S's natural arm movements while performing the center-out task. The intended velocity v_t^{int} was used to generate spike counts from simulated neurons, in which the spike rate $\lambda_i(t)$ of neuron i was given by

$$\lambda_i(t) = max\left(0, \langle PD_i, v_t^{int}\rangle + b_i\right)$$

$$y_t[i] \sim Poisson\left(\lambda_i(t) \cdot dt\right).$$

Neuron i had a baseline rate of b_i and preferred direction PD_i [34]. The baseline firing rate was fixed at 10 Hz for all neurons. $\langle PD_i, v_t^{int}\rangle$ is the dot product between the neuron's preferred direction PD_i and the intended velocity v_t^{int}. Spike counts were sampled from a Poisson distribution with minimum spike rate of 0. PD_i was randomly sampled from the distribution

$$\angle PD_i \sim uniform[0, 2\pi), \quad \|PD_i\| = \frac{14\,\text{spikes/sec}}{20\,\text{cm/sec}},$$

independently for each neuron. This method of simulating neural spiking activity was used previously in a BMI simulation in which human subjects generated the intended velocity with natural arm movements [35]. The simulation parameters for modulation depth ($\|PD_i\|$) and baseline firing rate have also been used previously [36]. Using this procedure, the firing properties of 15 neurons were sampled once and kept fixed for all simulations. Simulations used $dt = 100$ ms to match experimental conditions.

We simulated the same center-out task performed by the experimental subjects. The exact same pseudorandom target sequence was used in every simulation. Decoder parameters were initialized with no knowledge of neural firing properties. For simulations of KF BMI decoders, elements of C were initialized by sampling from a standard normal distribution and Q was initialized as $Q = 0.001 \cdot I_{15}$, in which I_J is the $J \times J$ identity matrix. SB-CLDA

was used to retrain KF decoder parameters while the simulated subject operated the BMI in closed-loop. Each decoding algorithm was simulated 1000 times. For each simulation, the first 8 targets in the task target sequence were all different (spanning all the targets) and were the only trials used for SB-CLDA training. SB-CLDA was terminated when 1 successful trial to each of the 8 targets was completed.

3 Results

3.1 Optimal control requires that the BMI plant contain no unexpected dynamical effects

The BMI plant (Eq. 4) contains 4 different dynamical subsystems: T defines the dynamics of $p(t)$ and $p(t+1)$, S defines the dynamics of $v(t)$ and $p(t+1)$, M defines the dynamics of $p(t)$ and $v(t+1)$, and N defines the dynamics of $v(t)$ and $v(t+1)$. We might expect that a physically realistic plant would (1) integrate cursor velocity to set the cursor position and (2) allow independent control of horizontal and vertical velocity. In equation form, if s and n are scalars in the range $(0, 1)$, then the first condition corresponds to $T = I_2$, $M = 0$, and $S = sI_2$ and the second condition corresponds to $N = nI_2$.

However, this plant structure is not generated when PVKF decoder parameters are estimated using standard MLEs, even when the training data come from natural arm movements. In this study, we initialized the KF parameters A and W from hand movements which were not perfectly straight. The MLE for A and W models path curvature as a correlation between horizontal and vertical velocity. Thus, $\bar{A}$ may represent physically inaccurate dynamics even when KF parameters are trained using data from natural arm movements. Our hypothesis is that in closed-loop control, these modeling idiosyncrasies translate to plant dynamics which are more difficult to control.

To understand how unexpected dynamics impact optimal feedback control, we simulated optimal control of a cursor plant in which one dynamical property of the plant was unknown to the controller. In other words, we simulated a situation in which the subject incorrectly assumed that plant dynamics were physically realistic. The LQG solution determines the optimal feedback controller using an "internal model" of plant dynamics. For our simulations, the internal model was

$$\bar{A}_{\text{internal model}} = \begin{bmatrix} I_2 & sI_2 & 0 \\ 0 & nI_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{7}$$

In the internal model, $T = I_2$, $M = 0$, $S = sI_2$, and $N = nI_2$. We selected the values $s = 0.055$ and $n = 0.6$ from typical values of experimental decoders. Based on this internal model, optimal feedback control was used to execute a center-out reach. Figure 4A shows the planned reach, which was identical for open-loop and closed-loop control simulations. In the simulations below, the true $\bar{A}$ differed from $\bar{A}_{\text{internal model}}$ by exactly one element. We simulated open- and closed-loop control in this manner four times, where each simulation perturbed a different parameter. Specifically, $T_{[0,0]}$ was replaced with its typical experimental value of 0.98 to form

$$\bar{A}_{\Delta T_{[0,0]}} = \begin{bmatrix} 0.98 & 0 & 0.055 & 0 & 0 \\ 0 & 1 & 0 & 0.055 & 0 \\ 0 & 0 & 0.6 & 0 & 0 \\ 0 & 0 & 0 & 0.6 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$M_{[0,0]}$ was replaced with its typical experimental value of 0.03 to form

$$\bar{A}_{\Delta M_{[0,0]}} = \begin{bmatrix} 1 & 0 & 0.055 & 0 & 0 \\ 0 & 1 & 0 & 0.055 & 0 \\ 0.03 & 0 & 0.6 & 0 & 0 \\ 0 & 0 & 0 & 0.6 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$N_{[0,1]}$ was replaced with its typical experimental value of -0.1 to form

$$\bar{A}_{\Delta N_{[0,1]}} = \begin{bmatrix} 1 & 0 & 0.055 & 0 & 0 \\ 0 & 1 & 0 & 0.055 & 0 \\ 0 & 0 & 0.6 & -0.1 & 0 \\ 0 & 0 & 0 & 0.6 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

and $\Delta n = |n_{xx} - n_{yy}|$ was replaced with its typical experimental value of 0.1 to form

$$\bar{A}_{\Delta(\Delta n)} = \begin{bmatrix} 1 & 0 & 0.055 & 0 & 0 \\ 0 & 1 & 0 & 0.055 & 0 \\ 0 & 0 & 0.6 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

For clarity, we will refer to $\bar{A}$ as the plant for these simulations only, as $\bar{B}$ is the same in each simulation. Dynamical disturbances were created because the LQG feedback controller was not updated after the plant was altered; the internal model used for planning and control did not match the true plant dynamics.

Mathematical analysis shows that each perturbation has a predictable effect. $\bar{A}_{\Delta T_{[0,0]}}$ has a dynamic which drives $|p_x|$, the absolute value of the cursor's horizontal position, to 0; the dynamical effect grows stronger as the cursor moves farther from the origin. $\bar{A}_{\Delta M_{[0,0]}}$ has a dynamic 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. $\bar{A}_{\Delta N_{[0,1]}}$ has a dynamic which causes horizontal velocity to decrease as vertical velocity increases, resulting in counterclockwise curl dynamics. Finally, $\bar{A}_{\Delta(\Delta n)}$ has a dynamic which makes velocity "control memory" asymmetric, causing the cursor to respond differently to velocity control inputs in the horizontal and vertical dimensions.

The simulated subject planned an open-loop control sequence using its internal model of plant dynamics, $\bar{A}_{\text{internal model}}$. This control sequence was executed with each of the four perturbed $\bar{A}$ matrices in turn. Figures 4B-E show this open-loop control executed after modifying $T_{[0,0]}$, $M_{[0,0]}$, $N_{[0,1]}$ and Δn, respectively. Figure 4B shows open-loop control of $\bar{A}_{\Delta T_{[0,0]}}$, where at the end of the reach, the cursor drifts left toward the vertical axis where $|p_x| = 0$. Figure 4C illustrates open-loop control of $\bar{A}_{\Delta M_{[0,0]}}$, where the cursor overshoots the planned vertical endpoint due to the non-zero vertical velocity at the endpoint. Figure 4D depicts open-loop control of $\bar{A}_{\Delta N_{[0,1]}}$, where the cursor trajectory curls off to the left. Intuitively, $\bar{A}_{\Delta N_{[0,1]}}$does not allow the cursor to move along the horizontal axis by controlling

only the horizontal velocity. Lastly, Figure 4E demonstrates open-loop control of $\bar{A}_{\Delta(\Delta n)}$, where the trajectory was still a straight line reach but short of the target in the vertical dimension. Simply put, open-loop control was unable to compensate for any of these effects because they were unknown at the time of trajectory planning.

Open-loop control problems cannot be fixed simply by switching to closed-loop control, as closed-loop control generated better but still suboptimal trajectories. Figures 4F-J show closed-loop control executed after modifying $T_{[0,0]}$, $M_{[0,0]}$, $N_{[0,1]}$ and Δn, respectively. For modifications to $T_{[0,0]}$, $M_{[0,0]}$ and $N_{[0,1]}$, closed-loop control reduced the impact of dynamical effects which were problematic for open-loop control, but did not eliminate them entirely. Interestingly, Figure 4J shows that when controlling $\bar{A}_{\Delta(\Delta n)}$, a curl effect was present in closed- 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 plant with symmetric velocity dynamics, it made the horizontal velocity larger even though it was originally correct. Therefore, unexpected decoder dynamics can have a deleterious effect on cursor controllability, even when control inputs are generated in closed-loop without control noise. These theoretical results imply that if subjects have a consistent control strategy, then BMI performance will be optimized by a particular structure of $\bar{A}$.

3.2 Closed-loop BMI performance correlates significantly with plant properties

Plant properties were correlated with closed-loop BMI performance. We measured the closed-loop BMI performance of subject S controlling the PVKF. These experimental blocks are represented by the dashed box in Figure 2, in which the parameters of the PVKF were kept fixed after SB-CLDA retraining. We analyzed 68 different PVKF parameter sets, which had a diverse range of dynamical effects. The plant properties of each PVKF parameter set, which were different in every session, were quantified using the measures $\|T - I_2\|_2$, $\left\|\bar{B}_{pos}\right\|_2$, $dist(N, nI_2)$, Δn, $\left\|\bar{B}_{vel}\right\|_2$, $||N||_2$, $||S||_2$ and $v_{\text{offset}}(target)$, as defined in Section 2.6. When correlating performance against plant properties across sessions, we weighted each session by the number of target-in trials, which varied greatly (min: 42, max: 438, mean: 231). This weighting ensured that each trial, rather than each session, had equal statistical weight. Table 1 summarizes the numerous significant correlations between decoder feature metrics and performance.

Linear position dynamics (quantified by $||T - I_2||_2$) and position control (quantified by $||\bar{B}_{pos}||_2$) were detrimental to target hold performance and reach accuracy. Table 1 shows that correlations between numerous performance measures and $||T - I_2||_2$ or $||\bar{B}_{pos}||_2$ suggest that non-degenerate position dynamics and position control were detrimental to cursor controllability. Interestingly, $||T - I_2||_2$ had a small range of experimental values, as T closely approximated I_2: $t_{xx} = 0.97 \pm 0.0144$ (mean $\pm$ std), $t_{xy} = 0.002 \pm 0.005$, $t_{yx} = 0.002 \pm 0.005$, and $t_{yy} = 0.972 \pm 0.009$. This seemingly small difference between between T and I_2 still correlated significantly with performance variability.

Our results regarding the utility of linear position dynamics and position control are consistent with two previous closed-loop cursor BMI experiments. These experiments showed that linear decoders which only estimate velocity and integrate it to obtain position outper-

form those which estimate position directly [15, 17]. Extending previous results, our data show that PVKF performance improved continuously, without sharp changes, as position-dependent dynamics became more degenerate. Velocity decoders, such as the VKF, may be more controllable than position/velocity decoders, such as the PVKF, because velocity decoders guarantee that both position dynamics and position control are degenerate. Consistent with previous closed-loop BMI studies, PVKF performance improved when position dynamics/control became more degenerate.

Asymmetries and cross-coupling in velocity dynamics were also correlated with performance loss. The hold error rate increased as $dist(N, nI_2)$ increased while MV and ECD increased as Δn increased. Unsurprisingly, the two different measures of velocity dynamical asymmetry correlated with different performance measures. Recall that $dist(N, nI_2)$ depends on n_{xy} and n_{yx} whereas Δn does not. These differences suggest that nonzero n_{xy} and n_{yx} were more problematic for holding than they were for reaching, in which their impact was mitigated by the much larger values of n_{xx} and n_{yy}. Performance improved when velocity dynamics were symmetric and orthogonal.

Cursor sensitivity and control memory had perhaps the most predictable relationship with performance. Higher sensitivity to neural control inputs, measured by $||\bar{B}_{vel}||_2$, corresponded to higher variability in reaching accuracy and reaching speed. Longer control memory, measured by $||N||_2$, corresponded to more frequent hold errors and less accurate reaches, but also faster reach times. As $||N||_2$ increased, past control inputs were "remembered" longer, resulting in a faster cursor. Figure 5 shows s_{xx}, s_{yy}, n_{xx}, and n_{yy} versus both the hold error rate and the mean reach time. The hold error rate and mean reach time had inverse relationships with n_{xx} and inverse relationships with n_{yy}. In other words, there were no control memory settings that simultaneously optimized both hold performance and reach speed. Performance improved when velocity sensitivity was not too high, whereas the control memory represented a continuous tradeoff between hold performance and reaching speed.

Non-uniform velocity offsets had target-specific correlations with hold performance, reach times, and the accuracy of velocity control. Target-specific significance was expected because M was not systematically varied during SB-CLDA. This resulted in different ranges of velocity offsets for each target. The average M for this set of 68 PVKF parameter settings and for subject S's workspace is shown in Figure 6B. For targets 3, 4, 6 and 7, as numbered in Figure 1, v_{offset}(target) was correlated with increases in the mean reach time ($r = 0.26$, 0.3, 0.28, 0.31 respectively, with $p < 0.05$). In words, reach times increased when the velocity offset vector pointed away from the center. For targets 2, 3, 4, 5 and 6, $|v_{\text{offset}}(\text{target})|$ was correlated with increases in hold error rate ($r = 0.3$, 0.26, 0.53, 0.45, 0.27 respectively, with $p < 0.05$), regardless of whether the offset pointed toward or away from the center. Thus, non-uniform velocity offsets represent a counter-intuitive tradeoff between hold performance and speed. For all targets, VCD mean and standard deviation increased as v_{offset}(target) increased ($r > 0.65$ and $r > 0.41$ respectively, $p < 0.001$). Despite inaccuracies in the direction of velocity control from $\bar{B}_{vel} y_t$ indicated by increases in VCD, the overall angular accuracy of the cursor measured by ECD did not vary significantly with v_{offset}(target). This in turn implies that either (1) subject S adapted to compensate for M or (2) dynamical effects due to M were partly canceled by interactions with any one of the other dynamical effects.

It is important to understand why non-uniform velocity offsets might be beneficial to

center-out task performance. The presence of non-zero M is best explained by examining its MLE from manual control center-out data in which (1) the center of the workspace is the origin, and (2) A is unconstrained. The center-out task inherently correlates hand position and hand velocity. When the hand is at the periphery of the workspace, the task always directs the hand back to the center. In other words, when the position of the hand is near the periphery, the velocity is most likely to point to the center. Under these conditions, the MLE for the KF learns the inherent correlation between v_t and p_t, where $M \approx -0.5I_2$ provides a simple linear method to capture the structure of the center-out task. Figure 6A illustrates $M = -0.5I_2$ when the center of the workspace is at coordinates (0,0). As the cursor gets farther from the center, the offset driving the cursor back to the center becomes proportionally larger. In this scenario, M creates a velocity offset pointing to the center of the workspace along every task axis. However, in our experimental setup, the origin was not at the center target, which made it impossible for non-uniform velocity offsets to be properly aligned to every task axis. Figure 6B depicts velocity offsets in subject S's BMI workspace in which the M generating the non-uniform velocity offsets was obtained by averaging all 68 sessions in the preceding correlation analysis. For the average experimental M for subject S, the velocity offsets are clearly not uniform over the workspace or symmetric over all targets. Thus, we only observed the effects of M when the non-uniform velocity offsets were properly aligned to the task axes.

Performance variability cannot be entirely attributed to cross-day learning or decoder neural ensemble size. Hold error rate and mean reach time did not improve significantly over the course of the experiment ($p > 0.89$ and $p > 0.23$, respectively), indicating that cross-session learning did not play a significant role in day-to-day performance variability. The number of units used in each decoder was uncorrelated with the hold error rate ($p > 0.65$) but it was significantly correlated with improvements in mean reach time and reaching accuracy (Table 1). However, the decoder features we analyzed were independent of neural ensemble size, and for several performance measures these decoder properties were more significantly correlated with performance than the number of units. Therefore, our results cannot be explained as the inherent variability of BMI performance across sessions.

3.3 Optimizing linear BMI plant dynamics for reach accuracy and hold performance

The preceding experimental results suggest two simple design rules for optimizing closed-loop accuracy and hold performance.

First, eliminate position-dependent dynamics and position control. We found that the hold error rate and several measures of reach accuracy improved when $T \to I_2$ and $\bar{B}_{pos} \to 0$. M represented a target-specific tradeoff between hold performance and reach speed. When $||M||$ decreased, we observed target-specific relationships in which the hold error rate would decrease but mean reach time would increase. By proposing that $M = 0$, we explicitly prioritize hold performance over speed.

Second, allow independent control of horizontal and vertical velocity. As we noted in Section 3.1, horizontal and vertical velocity cannot be controlled independently if $N \neq nI_2$ or $S \neq sI_2$. For the KF, the requirement of independent horizontal and vertical velocity

is satisfied by imposing condition that $N = nI_2$, because properties of the Kalman gain dictate that if $N = nI_2$, then $S = sI_2$ (A.3). We found that the hold error rate, movement variability, and movement angular accuracy all improved as $N \rightarrow nI_2$. Furthermore, for both n_{xx} and n_{yy}, values around 0.3 correspond to peak hold performance and values around 0.6 correspond to peak reach speed. These data suggest that $n_{xx} = n_{yy}$ and $n_{xy} = n_{yx} = 0$ are the optimal structure for velocity dynamics.

Together, these two conditions restrict the decoder to the form

$$\begin{bmatrix} p(t) \\ v(t) \\ 1 \end{bmatrix} = \begin{bmatrix} I_2 & sI_2 & \bar{p} \\ 0 & nI_2 & \bar{v} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p(t-1) \\ v(t-1) \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ \bar{B}_{vel} \\ 1 \end{bmatrix} y(t). \tag{8}$$

Alternative justifications for the first restriction on on position-dependent plant dynamics were given by Shenoy and colleagues, who pointed out that spike observations are actions taken by the subject based on visual feedback of the decoded cursor position [17]. Since the cursor position displayed to the subject causes changes to the observed neural activity, they argue based on causal calculus that visual feedback should eliminate uncertainty about the cursor's position. Thus, the ReFIT-KF decoding algorithm described in [17] intervenes in the Kalman gain computation in order to ensure that the estimator covariance of position terms ($P_{t|t-1}$ as defined in A.1) should be 0. However, these causal interventions assume that the subject always reacts immediately to visual feedback. Our simulations and experimental data motivate similar modifications without explicit assumptions about reaction time. Thus, the closed-loop perspective provides multiple ways to arrive at similar ways to improve BMI performance.

3.4 Cross-session consistency of BMI plant dynamics

PVKF plant properties defined by $\bar{A}$ and $\bar{B}$ changed during the course of SB-CLDA training as parameter settings for C and Q changed. We are interested in how $\bar{A}$ and $\bar{B}$ evolve during SB-CLDA to determine if plant properties become more (or less) like the structure described in Section 3.3 as the length of training time increases. As shown in our previous work, C and Q converged in norm during the SB-CLDA parameter re-estimation process [18]. Here, we are interested in whether plant properties defined by $\bar{A}$ and $\bar{B}$ (1) were consistent across sessions and (2) converged to values that were favorable to accuracy and/or hold performance.

Recall that SB-CLDA generates a sequence of parameter settings for C and Q, which in turn define a sequence of settings for $\bar{A}$ and $\bar{B}$. We compared each intermediate $\bar{A}$ and $\bar{B}$ generated during the SB-CLDA process. This is effectively a cross-session comparison of properties due to C and Q because A and W were identical for the sessions included in this analysis. Figure 7 shows median trajectories for $\bar{A}$ features during SB-CLDA training for 58 sessions with subject S, which corresponded to the dotted boxes in Figure 2. Decoder properties were linearly interpolated between the time points when parameters were updated (every 1-2 minutes) before the median trajectory was calculated. We utilized Kruskal-Wallis analysis of variance (KW-ANOVA) with a significance level of $p = 0.05$ to determine if the median final parameters varied significantly with the SB-CLDA training time (as defined in Section 2.4). .

Parameter trajectories for s_{xx}, s_{yy}, n_{xx}, and n_{yy} converged to values corresponding to peak hold performance. Figure 7 shows median trajectories for these parameters during SB-CLDA retraining. Experimental values of s_{xx} and s_{yy} started near 0.1 before SB-CLDA. After SB-CLDA training, their final values were in the much lower range $s_{xx} = 0.067 \pm 0.004$ (mean $\pm$ std) and $s_{yy} = 0.066 \pm 0.005$. This created a "dampening" effect on the cursor velocity, scaling down the velocity before it was integrated to update the position. Similarly, n_{xx} and n_{yy} were approximately 0.8 before SB-CLDA, and decreased to $n_{xx} = 0.396 \pm 0.06$ and $n_{yy} = 0.413 \pm 0.05$. In other words, SB-CLDA consistently determined that a smaller control memory, much smaller than the control memory prescribed by A and W trained from manual control data, was beneficial. Across training times, final values for s_{yy} and n_{yy} did not change significantly ($p > 0.05$) whereas s_{xx} and n_{xx} had statistically significant but small differences. Thus, SB-CLDA consistently drove the median values of s_{xx}, s_{yy}, n_{xx}, and n_{yy} toward values that corresponded to peak hold performance.

Conversely, SB-CLDA did not consistently eliminate dynamics that were correlated with performance loss, even when training time was increased. Figure 7 also shows median trajectories for Δn, $||M||_2$, $||\bar{B}_{pos}||_2$, and $||T - I_2||_2$. Experimental data and simulations suggested that these quantities should all be 0 for peak accuracy and hold performance, but these values were not achieved during experimental SB-CLDA sessions. Trajectories for Δn appear more like random walks than convergent trajectories, and these trajectories had higher final values than initial values. Similarly, $||M||_2$ typically had higher final values than initial values. $||T - I_2||_2$ decreased initially, but additional training time did not help to drive $||T - I_2||_2$ to 0. Likewise, median trajectories for $\left\|\bar{B}_{pos}\right\|_2$ decreased initially but plateaued at a non-zero value. KW-ANOVA indicated that for Δn, $||M||_2$, and $||T - I_2||_2$, median final values were not significantly different as training time increased; final values of $\left\|\bar{B}_{pos}\right\|_2$ had statistically significant but extremely small differences as training time increased.

This analysis reveals the inherent limitations of using an unconstrained MLE to calculate parameter settings for C and Q. Increasing training time insufficient to eliminate unwanted dynamical properties. Specifically, Δn does not display convergent behavior whereas $\left\|\bar{B}_{pos}\right\|_2$, $||M||_2$, and $||T - I_2||_2$ consistently converge to sub-optimal values. Thus, modifications to the MLE used by SB-CLDA are required to generate consistent plant dynamics.

3.5 Simulation results: Modifications to plant dynamics improved reach accuracy

Conceptually, the constraints we propose on $\bar{A}$ constitute a prior distribution on the feedback control strategies we believe the subject will employ. The $\bar{A}$ we proposed to optimized reach accuracy and hold performance is similar to the plant structure of the VKF, with two important differences: N must be diagonal to ensure symmetric velocity dynamics and $s < dt$ to create a dampening effect on the velocity dynamics. Two options are available to estimate dynamical systems parameters that are compatible with our prior beliefs about the subject's feedback control strategy.

As a first option, we can use constrained MLEs to estimate parameters A, W, C, and Q that conform to Eq. 8. In this case, Eq. 8 is a symmetrically dampened VKF (SDVKF). The decoder state still includes position terms, but position dynamics and position control

are degenerate. The parameter estimation procedure constrains A, W, and $C^T Q^{-1} C$ to have diagonal velocity terms and degenerate position terms. It was insufficient to only constrain the KF state-space model (A and W) because the neural firing model (C and Q) also affects plant dynamics through the Kalman gain (Eq. 13). For example, in our experiments with subject S, $A_{[0:1,0:1]} = I_2$ and $W_{[0:1,0:1]} = 0$ by design, but still we found that $T \neq I_2$ due to interactions with the neural firing model parameters. In addition to constraining the structure of $\bar{A}$, we can also pre-specify exact values for n and s, which allows us to keep $\bar{A}$ fixed across sessions even when C and Q must change due to changes in the neural decoding population. Pre-specifying n and s may reduces the parameter space that SB-CLDA must search over, which may be useful because SB-CLDA across many sessions for subject S produced extremely similar final values for s_{xx}, s_{yy}, n_{xx} and n_{yy}. Details of the parameter estimation procedure are presented in A.3.

As a second option, $\bar{A}$ and $\bar{B}$ can be estimated directly using stochastic gradient descent, without explicitly estimating a set of KF model parameters:

$$\begin{aligned}
\bar{B}^{(t+1)} &= \bar{B}^{(t)} - \mu \nabla_{\bar{B}^{(t)}} \mathbb{E}\left[||e(t)||^2\right] \\
&= \bar{B}^{(t)} - \mu \mathbb{E}\left[e(t) \nabla_{\bar{B}^{(t)}} \left(x_t^* - \bar{A} x_{t-1} - \bar{B}^{(t)} y_{t-1}\right)\right] \\
&= \bar{B}^{(t)} + \mu \mathbb{E}\left[e(t) y_{t-1}\right] \\
&= \bar{B}^{(t)} + \frac{\beta}{\mathbb{E}\left[||y_{t-1}||^2\right]} \mathbb{E}\left[e(t) y_{t-1}\right],
\end{aligned}$$

where $e(t)$ is the decoding error at time t, estimated as described in Section 2.4. This recursion constitutes one step of a decoder retraining procedure which aims to minimize $\mathbb{E}\left[||e(t)||^2\right]$, the mean squared decoder error. In the final equality above, the step size has been normalized by the "size" of the spikes used for control input, analogous to the normalized least mean squares algorithm [27]. $\bar{A}$ can be similarly updated or it can be kept fixed to designer specified values. Under this estimation procedure, the plant represented by Eq. 8 is a symmetrically dampened velocity linear system (SDVLS) because it no longer utilizes the KF generative spiking model. Closed-loop operation of this algorithm is described entirely by Eq. 8.

We simulated closed-loop neural control of the PVKF, VKF, SDVKF, and SDVLS. Simulations occasionally failed to complete the training set due to poor random initialization of decoder parameters. In these instances, the simulation was simply restarted. This problem was unique to simulations and not present during closed-loop experiments in which initial parameter estimates were better than purely random values. The SDVLS was initialized with $\bar{A}$ as in Eq. 7 and with $\bar{B} = 0$. Each decoding algorithm was simulated 1000 times and for each simulation we calculated the average ME and MV across all trials. Figure 8 shows that both the SDVKF and the SDVLS significantly outperformed the PVKF and the VKF in mean ME and mean MV (KW-ANOVA, $p < 10^{-4}$). The differences in mean ME and mean MV between the SDVKF and the SDVLS were significant ($p < 10^{-4}$), suggesting that the SDVLS may generate more accurate trajectories than any of the other decoding methods.

The SDVLS has three important advantages over the SDVKF: (1) it is computationally simpler both for re-estimating parameters and calculating decoder outputs, (2) it reduces the number of parameters that need to be re-estimated while retaining and (3) each parameter in

the BMI system is easily interpretable. Both the SDVKF and the SDVLS encapsulate a "random walk" model of hand movements in which the kinematic state of the hand is statistically independent of hand kinematics two time steps in the past (or farther) if the kinematic state one time step in the past is known. Several decoding algorithms with more realistic motion models have been developed (e.g. [36, 37, 38]), which, like the KF, are inference algorithms for different types of hidden Markov models (HMMs). The HMM structure conveniently allows the kinematic model of cursor movements to be specified independently of the neural firing model, which means that the kinematic model can change without drastically changing the inference algorithm. The SDVLS is no longer an HMM inference algorithm, and loses this potentially useful mathematical advantage. To our knowledge, there have been no studies assessing the closed-loop performance benefits of more complicated motion models over the standard random walk model used by the KF. Thus, it is unclear whether the SDVKF or the SDVLS is the better solution.

3.6 Closed-loop experimental results: Modifications to plant dynamics improved reach accuracy and endpoint hold performance

Subject J operated both PVKF ($n = 7$) and SDVKF ($n = 4$) decoders to perform the closed-loop BMI center-out task. Parameters for both decoding algorithms were initialized using visual feedback and re-trained using SB-CLDA.

Figure 9 shows two significant changes in task performance across sessions: (1) the median hold error rate was significantly lower for SDVKF sessions than for PVKF sessions (KW-ANOVA, $p < 0.05$) and (2) the median reach time was longer for SDVKF sessions than PVKF sessions ($p < 0.05$). Furthermore, SDVKF reaches were also more accurate than PVKF reaches. We compared the reach accuracy of SDVKF trials versus PVKF trials. When comparing reach accuracy, each target-in trial received equal statistical weight as in earlier analyses. The improvements in median reach accuracy were small but significant (by KW-ANOVA): 7.8% reduction in median ME ($p < 0.002$), 8.4% reduction in median MV ($p < 0.001$), 8.1% reduction in ECD ($p < 0.001$), and 5.1% reduction in VCD ($p < 0.001$).

Alterations to BMI plant dynamics had the expected effect on performance. Significant improvements in the hold error rate and reaching accuracy indicate that applying our BMI design rules to the plant sub-systems defined by T, $\bar{B}_{pos}$, N and S benefits cursor controllability. However, SDVKF reaches were also slower than PVKF reaches, as expected based on our constraints to M. Forcing $M = 0$ prevented the SB-CLDA from learning the correlation between cursor position and velocity inherent to the center-out task, making the slower cursor. In place of faster reaches, we gained greater accuracy in reaching and significantly better hold performance.

4 Discussion

BMI plant dynamics must be carefully crafted so as to maximize closed-loop performance. When PVKF parameters were estimated using MLEs, a diverse set of dynamical effects were created. In closed-loop BMI experiments, several PVKF dynamical properties were either detrimental to control or related to tradeoffs between speed versus reaching accuracy or speed versus holding performance. Position drift and curl dynamics were detrimental to control while the utility of non-uniform velocity offsets or longer control memory depended on the performance measure. These results suggest that BMI performance can be improved by systematically altering BMI plant dynamics to incorporate prior knowledge of the subject's feedback control strategy. The SDVKF and SDVLS decoders were instances of a general framework we developed to precisely shape BMI plant dynamics. In simulation, BMI accuracy improved when the SDVKF and the SDVLS were used to eliminate position drift, non-uniform velocity offsets, and curl dynamics. Using data from closed-loop experiment sessions, we compared the performance of SDVKF and PVKF trials. As expected, SDVKF reaches were slightly slower but more accurate and had far fewer target hold errors.

Our principal finding is that task-specific performance can be optimized by crafting BMI plant dynamics. Our general framework enabled us to shape BMI plant dynamics in order to trade speed for improved accuracy and target hold performance. Ultimately, the optimal tradeoff between speed and accuracy must depend on the BMI's intended purpose. For instance, a communication prosthesis might favor speed over accuracy if the communication mechanism involved reaching to large targets; this tradeoff is also implementable with our methods. Additionally, BMI plant dynamics can be adapted to a particular target layout. For example, the center-out task has a highly structured relationship between cursor position and velocity, and an automatic "return to center" dynamical feature can be built into the plant itself by modeling this correlation. In fact, we showed why this occurs when KF parameters are estimated using standard MLEs. This property may boost center-out performance at the potential cost of generalization to other tasks. Independently, the memory of the plant can be adjusted to compromise between speed and accuracy. More memory made the cursor faster but also made reaches less accurate and made hold errors more frequent. Thus, the memory time constant can be adjusted depending on the target size and hold time requirement, which together determine the minimum accuracy required to accomplish the task. Such adjustments are analogous to Fitts' law, which describes the empirical relationship between reach speed and target size in human hand movements [39]. Our approach provides the BMI designer with the ability to either customize the BMI to a particular task or ensure that it remains general-purpose.

Previous closed-loop BMI studies have used MLEs to model neural spiking [15, 17, 18, 25], but standard MLEs do not provide sufficient precision to craft plant dynamics. In our experiment, standard PVKF MLEs consistently generated detrimental dynamics and inconsistently favored hold performance over speed. To address these problems, the SDVKF models the "neural noise" covariance Q as (1) unrelated to the cursor's position and (2) independent with respect to the cursor's horizontal/vertical velocity. Furthermore, standard MLEs for the KF are likely to create a different $\bar{A}$ if the neural ensemble used for BMI decoding changes, for instance due to a loss of recorded cells from cell death or electrode array shifts [40]. Unlike any previous estimation methods for the KF, the SDVKF can keep

$\bar{A}$ identical across sessions, independent of parameter initialization. The ability to keep $\bar{A}$ the same, independent of the neural ensemble, may prove beneficial for long-term subject learning in BMI.

Previous demonstrations of high performance closed-loop BMIs have used a wide array of algorithms, which have primarily been linear. Our mathematical analyses suggest that these algorithms are more alike than they are different. As noted previously, the linear dynamical system can be used to represent the calculations executed by any linear BMI decoding algorithm. In fact, to optimize accuracy and hold performance, we proposed a plant structure which is very similar to the population vector algorithm [5] with a first-order low-pass filter applied to the decoded velocity. A study comparing the performance of different decoding algorithms found that the most important design choices were related to (1) modeling assumptions about the distribution of neural preferred directions and (2) methods used to smooth cursor kinematics [20]. For linear methods, both of these design choices are reflected in the dynamical system parameters $\bar{A}_t$ and $\bar{B}_t$. MLEs are arguably the most principled approach to estimating parameters, but ultimately closed-loop BMI performance may depend more on plant dynamics and subject learning than on parameter estimation methods.

Ultimately, our results suggest that plant dynamics must synergize with the subject's control policy for optimal results. In several studies, monkey subjects have controlled BMI decoders with rotation bias [19, 20, 21], as well as non-biomimetic linear mappings [11], using the brain's remarkable capacity to learn feedback control strategies. Consistent with our framework, proficient control of these sub-optimal plants required changes to the subject's control policy. Recall that even without control noise, closed-loop optimal control simulations did not produce perfectly straight trajectories when there was a mismatch between the controller's "internal model" and the true plant dynamics. Indeed, the "optimality" of optimal control relies on knowledge of plant dynamics when control policies are designed. Therefore, optimal BMI performance requires that either the subject's control policy conforms to the decoder or that the decoder conforms to the subject's control policy. Our methods provide a way to accomplish the latter.

In summary, we approximated the behavior of a BMI user as a simple feedback controller designed using an internal model of the BMI plant. We used simulations, analysis of experimental data, and *in vivo* closed-loop experiments to gain a better understanding of BMI plant properties and their influence on performance. We developed a framework for 2D BMIs to predict the effects of altered plant dynamics on closed-loop performance. This framework could be used to tailor BMIs to subject or task requirements. Furthermore, our decoder parameter estimation methods generate BMI plant dynamics with less variability than previous methods. This may be important for improving the clinical robustness of BMI decoders by ensuring that plant dynamics remain consistent if and when neural decoder parameters are retrained.

Figures and Figure Captions

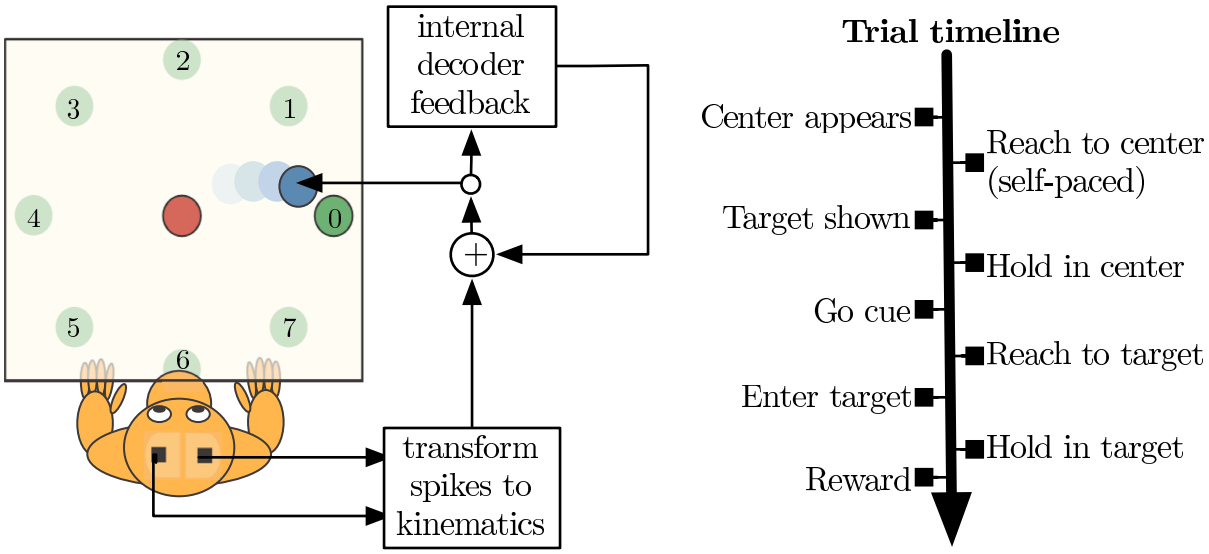

Figure 1: BMI task. While controlling a 2D BMI cursor, the subjects sat in a primate chairs with their arms outside of the cursor workspace. Subjects were required to operate the BMI in a center-out-and-back task. Visual feedback of the cursor's decoded position was presented to the subjects to enable closed-loop BMI operation. Recursive decoders, like the KF, use internal decoder feedback which allows decoded cursor outputs to be generated based on previous decoder outputs.

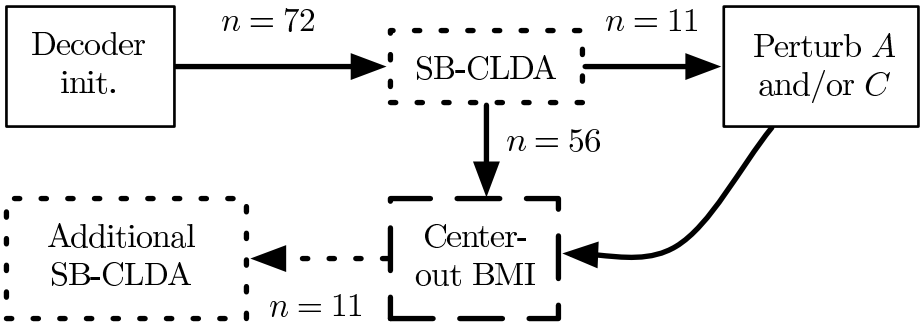

Figure 2: Diagram of SB-CLDA experimental paradigm for subject S. Decoder parameters were initialized without overt contralateral arm movements, e.g. using neural activity captured while the subject passively watched a cursor moving under purely software control. SB-CLDA was applied to improve closed-loop control performance (dotted box). After SB-CLDA was complete, the decoder was used for center-out BMI (lower dashed box). In 11 sessions, we modified the KF matrices A or C to generate a "perturbed" decoder, without using SB-CLDA to improve performance after the perturbation. In a different set of 11 sessions, after the subject had performed center-out BMI, SB-CLDA was utilized again to further improve performance.

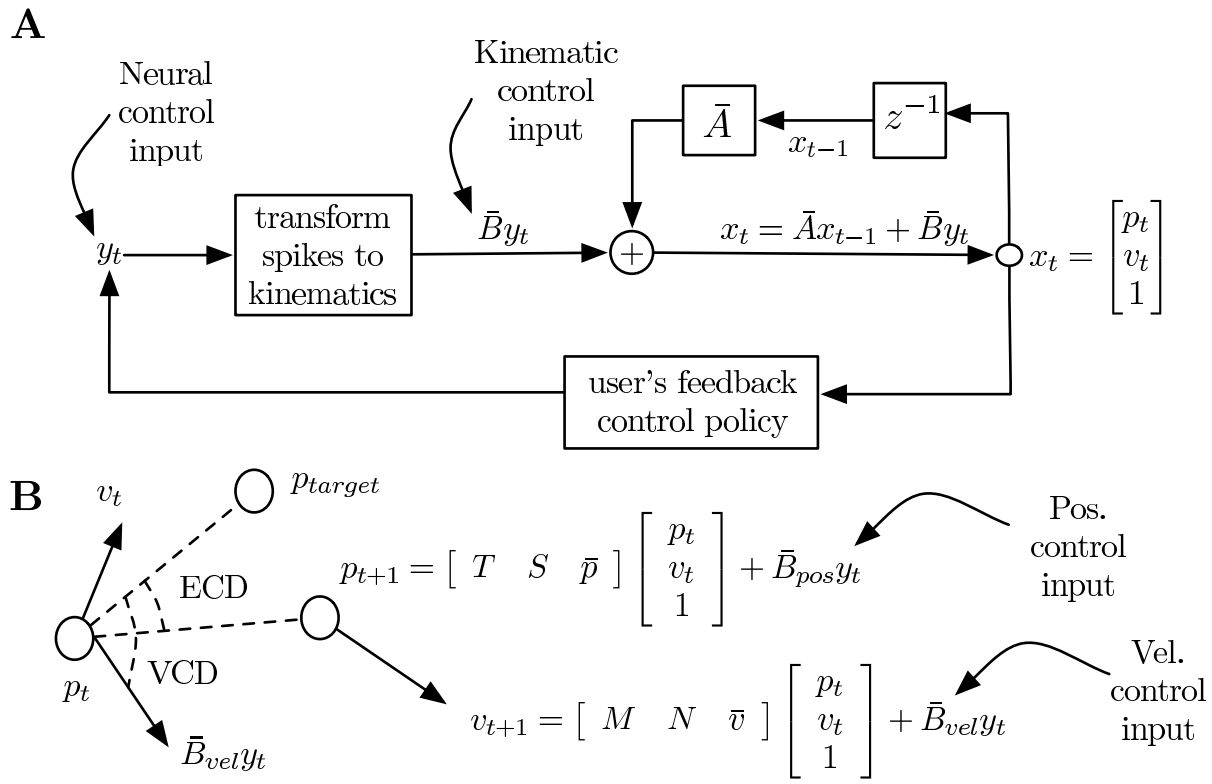

Figure 3: System diagram of a closed-loop BMI plant. (A) The closed-loop BMI block diagram: The BMI is linear if (1) the mapping from spike counts to kinematic control input can be written as a matrix multiplication, and (2) there is only a linear dependence on the cursor's previous kinematic state. (B) Graphical representation of Eq. 4: Mathematical operations describing how the position and velocity of the cursor are updated with a linear position/velocity BMI. Despite the simplicity of the KF state space model, the true underlying dynamics are complicated by the Kalman gain, which involves all KF parameters (A.1). The angular error measures ECD and VCD are defined as well.

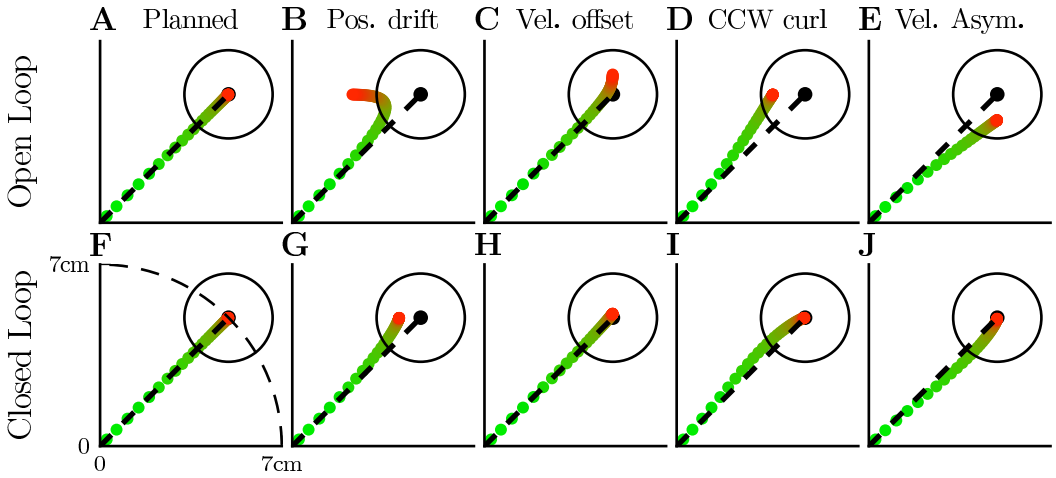

Figure 4: Simulations of open-loop and closed-loop optimal control using a fixed internal model. (A) Open-loop planned reach from the center to the peripheral target, where the trajectory begins as green and transitions to red at the end of the reach. The open-loop control sequence was executed on a plant that was slightly different from the internal model used for planning. Depending on the difference between the internal model and the true plant, 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.

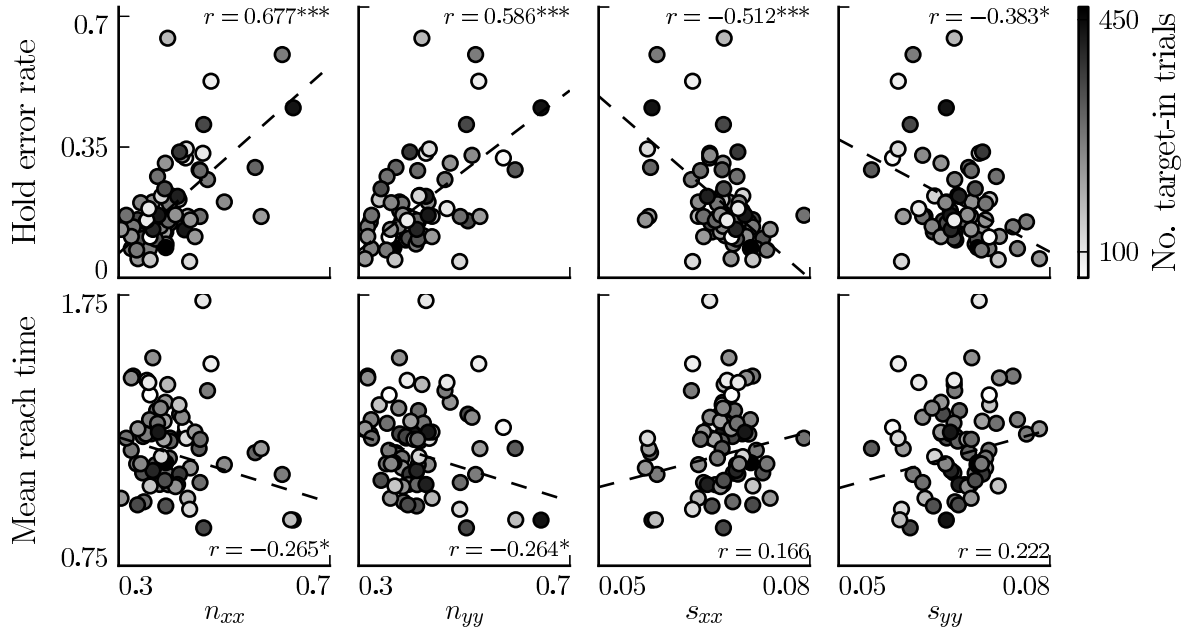

Figure 5: The tradeoff between speed and holding accuracy is reflected in decoder parameters. The figure plots the decoder parameters n_{xx}, n_{yy}, s_{xx}, and s_{yy} versus the hold error rate and mean reach time for 68 closed-loop BMI sessions with subject S. Each scatter point represents one session and the darkness of each point depicts the number of trials initiated in that session. The data shows significant correlations between n_{xx} and the hold error rate/mean reach time, but with opposite signs. A similar trend is shown for and n_{yy}, indicating a natural tradeoff between reaching speed and the ability to accurately stop the cursor. The corresponding trend for s_{xx} and s_{yy} was not significant.

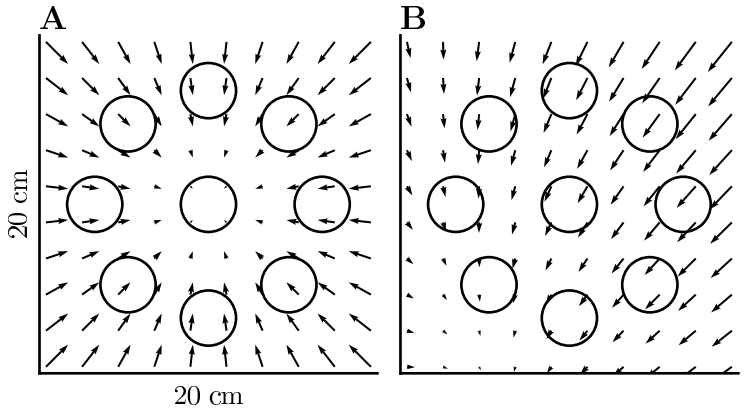

Figure 6: Visualization of non-uniform velocity offsets generated by the PVKF, superimposed onto the workspace targets. (A) Non-uniform velocity offsets learned when training KF parameters using manual control center-out data, as described in Section 3.2. The velocity offsets all point to the workspace center, with increasing strength as the cursor gets farther from the center. (B) Non-uniform velocity offsets as generated by the average experimental decoder for subject S. The symmetry of the left figure is lost in experimental decoders for reasons described in Section 3.2, meaning that each part of the workspace may have significantly different velocity offsets.

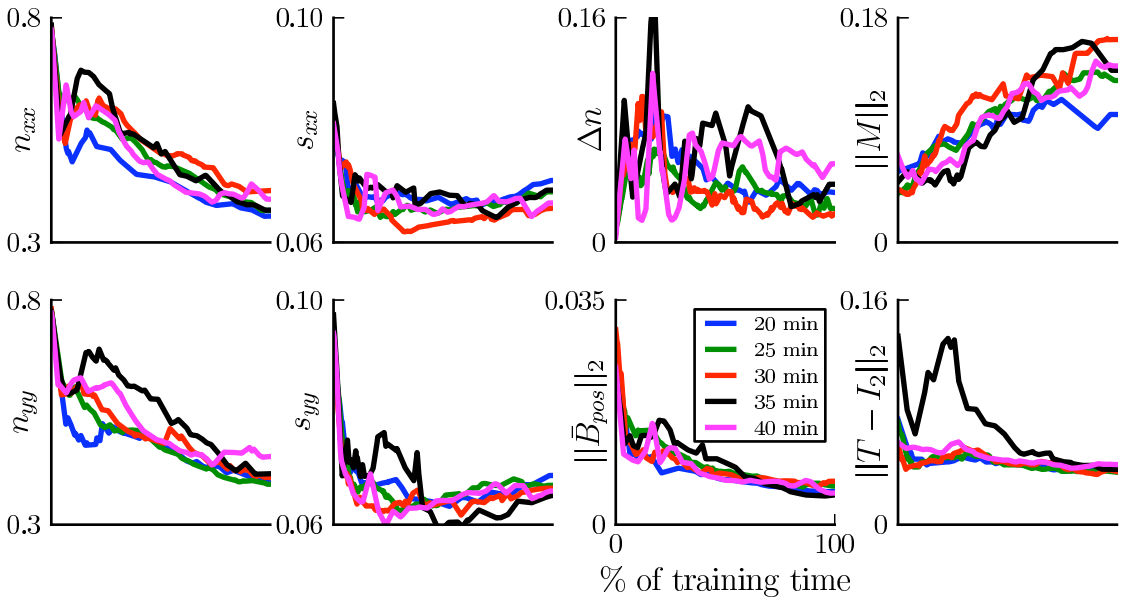

Figure 7: SB-CLDA parameter trajectories. Median parameter trajectories for the 5 most common durations for which SB-CLDA parameter estimation was executed [20 min ($n = 13$), 25 min ($n = 19$), 30 min ($n = 16$), 35 min ($n = 5$), and 40 min ($n = 5$)] are shown for 8 decoder features. The evolution of each decoder feature for each of the 5 training times is shown on a normalized time axis. Briefly, n_{xx}, n_{yy}, s_{xx}, and s_{yy} consistently converge to settings for peak hold performance, whereas Δn does not appear to converge, $\left\|\bar{B}_{pos}\right\|_2$ and $\|T - I_2\|_2$ converge to sub-optimal values, and $\|M\|_2$ converges to values correlated with a loss in hold performance. These results indicate the need for modifications to the parameter estimation procedure.

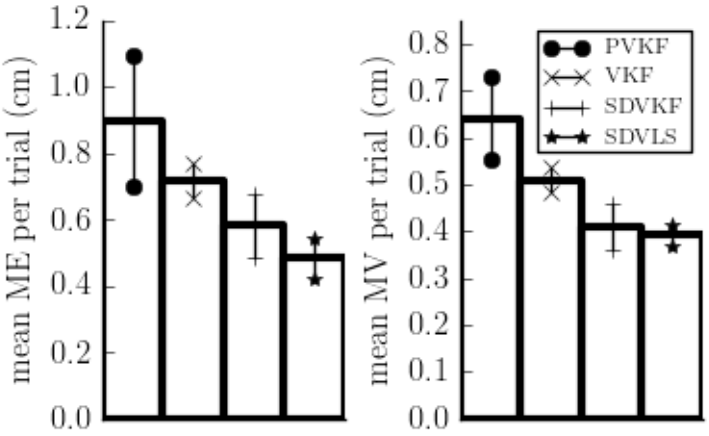

Figure 8: Results of closed-loop BMI simulations. For 1000 independent simulation runs, we trained the VKF, PVKF, SDVKF, and SDVLS using SB-CLDA. The bar plot shows mean performance for each algorithm, with vertical bars indicating the standard deviation of performance across simulation runs. The SDVKF and SDVLS outperformed the VKF and the PVKF with statistically significant reductions in mean ME and mean MV (Kruskal-Wallis ANOVA, $p < 10^{-4}$).

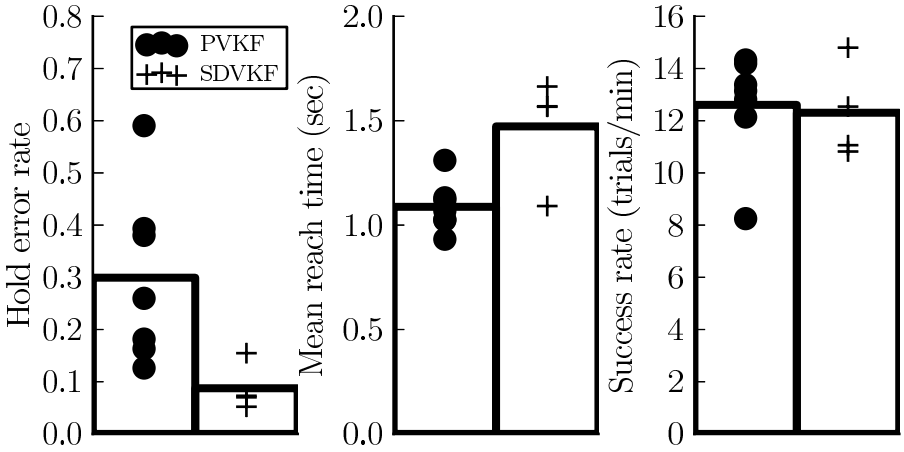

Figure 9: Comparison of PVKF and SDVKF performance in closed-loop BMI experimental sessions. Subject J used both PVKF and SDVKF decoders in separate sessions. For all sessions, decoder parameters were initialized from neural activity while the subject passively observed artificial cursor movements (visual feedback), and retrained using SB-CLDA. SDVKF sessions had significantly higher mean reach times than the PVKF (KW-ANOVA, $p < 0.05$), but also had significantly lower hold error rates, with no net change in the overall success rate.

Tables

	HER	Reach Time		ME		MV		ECD		VCD	
Decoder features		mean	std	mean	std	mean	std	mean	std	mean	std
# units	-0.05	-0.29 *	-0.46 ***	-0.29 *	-0.33 **	-0.42 ***	-0.46 ***	-0.45 ***	-0.45 ***	-0.08	-0.04
$\|\|B_{pos}\|\|_2$	0.397 ***	0.002	0.243 *	0.299 *	0.208	0.313 **	0.150	0.290 *	0.249 *	0.308 *	0.329 **
$\|\|T - I_2\|\|_2$	0.597 ***	-0.24	-0.07	0.145	0.014	0.285 *	0.087	0.181	0.170	0.204	0.181
$min_n\|\|N - nI_2\|\|_2$	0.257 *	0.151	0.106	0.100	0.060	0.232	0.167	0.166	0.205	0.065	0.134
$\|\Delta n\|$	0.181	0.005	0.178	0.097	0.108	0.192	0.249 *	0.253 *	0.301 *	0.015	0.060
$\|\|B_{vel}\|\|_2$	-0.06	0.252 *	0.402 ***	0.218	0.118	0.262 *	0.074	0.274 *	0.155	0.201	0.135
$\|\|S\|\|_2$	-0.52 ***	0.236	-0.16	-0.42 ***	-0.32 **	-0.41 ***	-0.31 *	-0.35 **	-0.30 *	-0.26 *	-0.20
$\|\|N\|\|_2$	0.682 ***	-0.30 *	0.135	0.567 ***	0.428 ***	0.570 ***	0.425 ***	0.440 ***	0.384 **	0.242 *	0.238

Table 1: Significant correlations between decoder properties and performance measures. Hold error rate is abbreviated as HER. Correlations were weighted by the number of target-in trials per session. Asterisks denote p-values: $p < 0.001$ (***), $p < 0.01$ (**), and $p < 0.05$ (*).

A BMI decoding algorithms

A.1 The Kalman Gain

Alternative derivations of the Kalman gain can be found in [27, 26]. We provide a condensed derivation for completeness. Let us define

$$y_{t_0}^{t_1} = \{y_{t_0}, \cdots, y_{t_1}\}.$$

As special cases, let $y^t = y_0^t$ and y^N be the set of all observations.

In inference algorithms, we are concerned with finding the conditional distribution $p(x_t|\, y^s)$. For any s, the distribution $p(x_t|\, y^s)$ is Gaussian. The Kalman *filter* is an efficient algorithm for estimating $p(x_t|\, y^t)$, whereas the Kalman *smoother* provides an estimate of $p\left(x_t|\, y^N\right)$. The Kalman smoother is unsuitable for real-time estimation, as future observations are unavailable to a real-time estimator, and we do not derive it here. We denote estimates of x_t by the mean and variance of the estimator:

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

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

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

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 $p(x_t, y_t|\, y^{t-1})$:

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

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

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

The distribution becomes

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

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

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

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

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

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

A.2 Maximum-likelihood estimation (MLE) for KF parameters

Maximum-likelihood for KF parameter estimation involves estimating the mean and variance of Gaussians. Given samples $\{z_t\}_{t=1}^N$ where $z_t \sim \mathcal{N}(\mu, \Sigma)$, the maximum-likelihood estimator for the mean vector and covariance matrix are

$$\max_{\mu,\Sigma} \qquad -\log\det\Sigma - tr\left(\frac{1}{N}\sum_{t=1}^{N}\left(z_t - \mu\right)\left(z_t - \mu\right)^T\right).$$

A proof that this is the maximum-likelihood problem to solve can be found in [31], which also provides an analytical solution to the problem in the unconstrained case. The KF allows the state space model and the observation model to be fit independently. To find A and W, we solve the optimization problem:

$$\max_{A,W} \qquad -\log\det W - tr\left(W^{-1}\frac{1}{T-1}\sum_{t=1}^{T-1}\left(x_{t+1} - Ax_t\right)\left(x_{t+1} - Ax_t\right)^T\right) \tag{9}$$

$$\text{subj. to} \qquad A = \begin{bmatrix} 1 & 0 & \Delta & 0 & 0 \\ 0 & 1 & 0 & \Delta & 0 \\ 0 & 0 & a_v^{xx} & a_v^{xy} & 0 \\ 0 & 0 & a_v^{yx} & a_v^{yy} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

$$W = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & w_v^{xx} & w_v^{xy} & 0 \\ 0 & 0 & w_v^{yx} & w_v^{yy} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{11}$$

We will use the symbol Δ in place of the dt used in the main text as it makes equations below clearer. The entries of A are restricted so that the state space dynamics reflect physics: cursor velocity is "integrated" to get cursor position. W is fit to model the residual error not captured by the estimated A. W is similarly restricted to have uncertainty only in the velocity terms, leaving position "predictions" to be made deterministically from the estimated velocity.

A similar optimization problem is solved to estimate C and Q:

$$\max_{C,Q} \qquad -\log\det Q - tr\left(Q^{-1}\frac{1}{T}\sum_{t=1}^{T}\left(u_t - Cx_t\right)\left(u_t - Cx_t\right)^T\right). \tag{12}$$

Formulating the parameter estimation procedure as an optimization problem allows easy application of convex parameter constraints.

A.3 Symmetrically damped velocity KF (SDVKF)

A.3.1 Parameter constraints.

In the following math, we will assume that the KF state is represented by

$$x_t = \begin{bmatrix} p_x(t) & p_y(t) & v_x(t) & v_y(t) \end{bmatrix}$$

Suppose that

$$A = \begin{bmatrix} I_2 & dt \cdot I_2 \\ 0 & aI_2 \end{bmatrix}, W = \begin{bmatrix} 0 & 0 \\ 0 & wI_2 \end{bmatrix}, C^T Q^{-1} C = \begin{bmatrix} 0 & 0 \\ 0 & dI_2 \end{bmatrix} \tag{13}$$

and that we start the KF with exact knowledge of the initial state, i.e. $P_{0|0} = 0$. We define a set of matrices where elements are formed by the Kronecker product of a 2×2 matrix with I_2:

$$F = \left\{ M \,\middle|\, M = \begin{bmatrix} a & b \\ c & f \end{bmatrix} \otimes I_2, a, b, c, f \in \mathbb{R} \right\}.$$

An equivalent set definition is

$$F = \left\{ M \,\middle|\, M = \begin{bmatrix} aI_2 & bI_2 \\ cI_2 & fI_2 \end{bmatrix}, a, b, c, f \in \mathbb{R} \right\}.$$

When we multiply or add matrices in F, the resulting product/sum remains in F. First we will show that $P_{t|t-1} \in F$ under the KF model constraints above implies that $P_{t+1|t} \in F$, and then we will use this fact to show that it implies A_t satisfies Eq. 8 for all t. Calculations that are merely matrix algebra we leave to MATLAB's symbolic math engine (Algorithm 1) and include only the final results here.

$P_{t+1|t}$ and $P_{t|t-1}$ are related by the Riccati recursion:

$$\begin{aligned} P_{t|t} &= P_{t|t-1} - P_{t|t-1}C^T(CP_{t|t-1}C^T + Q)^{-1}CP_{t|t-1} \\ P_{t+1|t} &= AP_{t|t}A^T + W \\ &= A\left(P_{t|t-1} - P_{t|t-1}C^T(CP_{t|t-1}C^T + Q)^{-1}CP_{t|t-1}\right)A^T + W \\ &= AP_{t|t-1}A^T - AP_{t|t-1}C^T(CP_{t|t-1}C^T + Q)^{-1}CP_{t|t-1}A^T + W. \end{aligned} \tag{14}$$

Base case. $P_{1|0} = W$, which is not invertible given our constraints. To complete the base case, we must have $P_{2|1} \in F$ and $P_{2|1}$ invertible:

$$\begin{aligned} P_{2|1} &= AP_{1|0}A^T - AP_{1|0}C^T(CP_{1|0}C^T + Q)^{-1}CP_{1|0}A^T + W \\ &= AWA^T - AWC^T(CWC^T + Q)^{-1}CWA^T + W. \end{aligned}$$

The base case is complete if $C^T(CWC^T + Q)^{-1}C \in F$ and $P_{2|1}$ is invertible:

$$\begin{aligned} C^T(CWC^T + Q)^{-1}C &= C^T(Q + CW \cdot I_4 \cdot C^T)^{-1}C \\ &= C^T\left[Q^{-1} - Q^{-1}CW\left(I_4^{-1} + C^TQ^{-1}CW\right)C^TQ^{-1}\right]C, \end{aligned}$$

where the last equality follows from the matrix inversion lemma. Substituting in the quantities W and $C^TQ^{-1}C$ as defined above shows that $C^T(CWC^T + Q)^{-1}C \in F$. Thus, we can evaluate $P_{2|1}$:

$$P_{2|1} = \frac{1}{1+dw}\left(\begin{bmatrix} \Delta^2 w & \Delta aw \\ \Delta aw & a^2w + w + dw^2 \end{bmatrix} \otimes I_2\right).$$

Therefore, $P_{2|1} \in F$ and $P_{2|1}$ is invertible.

Inductive case. Using the matrix inversion lemma we can expand the matrix inverse product in Eq. 14:

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

which we can substitute in to get

$$\begin{aligned} P_{t+1|t} &= AP_{t|t-1}A^T + W \\ &\quad -AP_{t|t-1}C^TQ^{-1}C\left[I - \left(P_{t|t-1}^{-1} + C^TQ^{-1}C\right)^{-1}C^TQ^{-1}C\right]P_{t|t-1}A^T. \end{aligned} \tag{15}$$

To show the general inductive case, let

$$P_{t|t-1} = \begin{bmatrix} eI_2 & fI_2 \\ fI_2 & gI_2 \end{bmatrix} \in F,$$

Algorithm 1 MATLAB symbolic algebra code to compute $P_{2|1}$ and $P_{t+1|t}$ under SDVKF constraints.

```
d = sym('d', 'real');
A = [eye(2), dt*eye(2); zeros(2), a*eye(2)];
W = [zeros(2), zeros(2); zeros(2), w*eye(2)];
C_xpose_Q_inv_C = [zeros(2), zeros(2); zeros(2), d*eye(2)];

% base case
M = C_xpose_Q_inv_C - C_xpose_Q_inv_C*W*inv(eye(4) + ...
C_xpose_Q_inv_C*W)*C_xpose_Q_inv_C;
P_2given1 = A*W*A' - A*W*M*W*A' + W;

% inductive case
f = sym('f', 'real');
e = sym('e', 'real');
g = sym('g', 'real');
P = kron([e, f; f, g], eye(2));
P_tplus1_given_t = A*P*A' + W ...
    - A*P*(C_xpose_Q_inv_C - C_xpose_Q_inv_C*...
        inv( inv(P) + C_xpose_Q_inv_C )*C_xpose_Q_inv_C)*P*A';
KC = P*(C_xpose_Q_inv_C - C_xpose_Q_inv_C*(P^-1 + ...
    C_xpose_Q_inv_C)^(-1)*C_xpose_Q_inv_C);
A_bar = (eye(4) - KC)*A;
```

and let e and g be nonzero, which ensures that $P_{t|t-1}$ is invertible. $P_{t+1|t} \in F$ can be verified by computing Eq. 15:

$$P_{t+1|t} = \frac{1}{gd+1}\begin{bmatrix} deg + e + 2\Delta f + \Delta^2 g - df^2 & (f+\Delta g)a \\ (f+\Delta g)a & a^2 g + wgd + w \end{bmatrix} \otimes I_2.$$

By induction, $P_{t+1|t} \in F$ for all t. Therefore, the 3 conditions on A, W, and $C^T Q^{-1} C$ together imply that, for all t, $\bar{A}_t$ is in the form of Eq. 8:

$$\begin{aligned} \bar{A}_t &= (I - K_t C)\, A \\ &= \left(I - P_{t|t-1}C^T\left(CP_{t|t-1}C^T + Q\right)^{-1} C\right) A \\ &= \begin{bmatrix} I_2 & \frac{\Delta gd + \Delta - fda}{gd+1} I_2 \\ 0 & \frac{a}{gd+1} I_2 \end{bmatrix}. \end{aligned} \tag{16}$$

Alternatively, we can also use $\bar{A}_t$ to calculate $P_{t+1|t}$:

$$\begin{aligned} P_{t|t} &= P_{t|t-1} - P_{t|t-1}C^T(CP_{t|t-1}C^T + Q)^{-1}CP_{t|t-1} \\ &= (I - K_t C)\, P_{t|t-1} = \bar{A}_t A^{-1} P_{t|t-1} \\ P_{t+1|t} &= AP_{t|t}A^T + W = A\bar{A}_t A^{-1} P_{t|t-1} A^T + W. \end{aligned}$$

The last equality above also implies that $P_{t+1|t} \in F$, providing an alternate proof for the inductive case.

A.3.2 Constraining plant sensitivity parameters n and s.

One problem of interest may be to define numerical constraints on d, to narrow the range of search values for n during experimental SB-CLDA.

For the PVKF, we observed that $\lim_{t\to\infty} P_{t+1|t} = P$. In the case of the SDVKF, the limit P does not exist because the position elements of the system are no longer stable. Consider, for example, a constant bounded velocity input to the system. The resulting position is potentially unbounded, meaning that the uncertainty about the cursor's position, $P(t+1|t)_{[0:1,0:1]}$ cannot converge.

Though $P(t+1|t)_{[0:1,0:1]}$ is not convergent, $P(t+1|t)_{[2:3,0:1]} = P(t+1|t)_{[0:1,2:3]}$ and $P(t+1|t)_{[2:3,2:3]}$, which represent uncertainty due to velocity elements, are convergent. In fact, steady-state values of $\bar{A}$ and $\bar{B}$ are independent of $P(t+1|t)_{[0:1,0:1]}$. We solve for the convergent values using the knowledge that

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

This leaves us with the following equations:

$$\begin{aligned} g &= \frac{a^2 g + wgd + w}{1+gd} = \frac{a^2 g}{1+gd} + w \\ f &= \frac{(f+\Delta g)a}{gd+1}. \end{aligned} \tag{17}$$

Solving for g and then f yields:

$$\begin{aligned} g &= \max\left\{ \frac{-(1-a^2-wd) \pm \sqrt{(1-a^2-wd)^2 - 4d(-w)}}{2d} \right\} \\ \beta &= \frac{\Delta g a}{gd+1-a}. \end{aligned}$$

We observe that f and g are fully determined given a, w, and d. Consequently, we can calculate $\lim_{t\to\infty} A_t$ from Eq. 16. This implies that $\bar{A}_t$ and $\bar{B}_t$ are convergent though $P(t+1|t)_{[0:1,0:1]}$ is not, a property which is due to the constraint on $C^T Q^{-1} C$. Let

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

Let $C_{pos} = C_{[:,0:1]}$. Our constraint on $C^T Q^{-1} C$ requires that $C_{pos}^T Q^{-1} C_{pos} = 0$, which in turn means that C_{pos} is in the null space of Q^{-1}, i.e. $C_{pos}^T Q^{-1} = 0$. Let $\Sigma = \left(P_{t|t-1} + C^T Q^{-1} C\right)^{-1}$ which means that $\Sigma \in F$. Then

$$\begin{aligned}
&\left[I - C^T Q^{-1} C \left(P_{t|t-1} + C^T Q^{-1} C\right)^{-1}\right] C^T Q^{-1} \\
&\quad = \left(\begin{bmatrix} 1 & 0 \\ -d\Sigma_{[2:3,2:3]} & 1 - d\Sigma_{[2:3,2:3]} \end{bmatrix} \otimes I_2\right) \cdot C^T Q^{-1} \\
&\quad = \begin{bmatrix} C_{pos}^T \\ -d\Sigma_{[2:3,2:3]} C_{pos}^T + \left(1 - d\Sigma_{[2:3,2:3]}\right) C_{vel}^T \end{bmatrix} Q^{-1} \\
&\quad = \begin{bmatrix} 0 \\ \left(1 - d\Sigma_{[2:3,2:3]}\right) C_{vel}^T Q^{-1} \end{bmatrix}.
\end{aligned}$$

Therefore,

$$K_t = P_{t|t-1} \begin{bmatrix} 0 \\ \left(1 - d\Sigma_{[2:3,2:3]}\right) C_{vel}^T Q^{-1} \end{bmatrix} = \begin{bmatrix} f\left(1 - d\Sigma_{[2:3,2:3]}\right) C_{vel}^T Q^{-1} \\ g\left(1 - d\Sigma_{[2:3,2:3]}\right) C_{vel}^T Q^{-1} \end{bmatrix}.$$

Thus, $P(t+1|t)_{[0:1,0:1]}$ has no impact on the Kalman gain. Note that that this expression for the Kalman gain means that $\bar{B}_{pos} \neq 0$, as implied by Eq. 8, but rather that $\bar{B}_{pos} \propto \bar{B}_{vel}$. Enforcing the requirement $\bar{B}_{pos} = 0$ requires a *causal intervention* to set $f = 0$ every iteration of the KF loop [17].

To find corresponding constraints on n, we utilize Eq. 16:

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

We substitute this into Eq. 17:

$$g = \frac{n}{a} \cdot a^2 g + w \Rightarrow g = \frac{w}{1 - an}.$$

Therefore,

$$d = \frac{1}{g} \frac{a - n}{n} = \frac{(1 - an)(a - n)}{wn}.$$

A.3.3 Implementation details.

To apply the constraints in 13 to the KF model parameters, it is straightforward to apply the constraints for A and W. We need only modify Eq. 10 so that $a_{xy}^v = a_{yx}^v = 0$ and $a_{xx}^v = a_{yy}^v$, and modify Eq. 11 so that $w_{xy}^v = w_{yx}^v = 0$ and $w_{xx}^v = w_{yy}^v$.

We can find an appropriate C and Q by solving the following convex optimization problem:

$$\begin{aligned}
\max_{C,Q} \quad & \log\det Q^{-1} - tr\left(Q^{-1} \frac{1}{T} \sum_{t=1}^{T} (y_t - Cx_t)(y_t - Cx_t)^T\right) \\
\text{subj. to} \quad & C^T Q^{-1} C = \begin{bmatrix} 0 & 0 \\ 0 & dI_2 \end{bmatrix} \\
& d_{min} \leq d \leq d_{max}.
\end{aligned} \tag{18}$$

d_{min} and d_{max} are free parameters which can incorporate prior knowledge of the exact values of the plant, as described in the previous section where we solved for n as a function of a, w, and d. The constraint in the optimization problem above is always satisfiable: $C = 0$ and $Q = E\left[y_t y_t^T\right]$ satisfy the constraints trivially. One approximate solution is to estimate $C_{[:,2:3]}$ the standard MLE and then find an appropriate Q. Using the CVX package for MATLAB (CVX Research Inc, [41]), this problem can be solved in seconds. We can set constraints on d_{min} and d_{max} based on the range of n values we wish to have in $\lim_{t\to\infty} \bar{A}_t$. To re-estimate parameters using SB-CLDA, we must run the re-estimation procedure for Q in a separate thread; although the computation time is short, it is much longer than the BMI loop time of 100 ms.

A faster alternative involves re-parameterizing the KF inference algorithm. We note from Eq. 15 that to run the KF, we must know $C^T Q^{-1} C$ and $C^T Q^{-1}$; it is not necessary to know C and Q if these two products are known. This re-parameterization of the KF is a common way to reduce computation cost when the number of observations is greater than the number of state variables. Recall that the conditional model of neural firing is

$$y_t | x_t \sim \mathcal{N}\left(Cx_t, Q\right).$$

Using Gaussian distribution properties,

$$C^T Q^{-1} y_t | x_t \sim \mathcal{N}\left(C^T Q^{-1} C x_t, C^T Q^{-1} Q Q^{-1} C\right) = \mathcal{N}\left(C^T Q^{-1} C x_t, C^T Q^{-1} C\right).$$

If $C^T Q^{-1} C$ is known, then $C^T Q^{-1}$ can be calculated using pseudo-inverse methods. Let $D = C^T Q^{-1} C$. D is a symmetric positive semidefinite matrix, representing the covariance of the random process $\left\{C^T q_t^{-1}\right\}_{t=1}^{+\infty}$ where $q_t^{-1} \sim \mathcal{N}(0, Q^{-1})$. We can estimate D using the constrained maximum likelihood problem below:

$$\begin{aligned} \max_{d} \quad & \log\det D_{[2:3,2:3]}^{-1} - trace\left(D_{[2:3,2:3]}^{-1} \hat{D}_{[2:3,2:3]}\right) \\ \text{subj. to} \quad & D = \begin{bmatrix} 0 & 0 \\ 0 & dI_2 \end{bmatrix}. \end{aligned}$$

We estimate only $D_{[2:3,2:3]}$ because the constraints are such that D^{-1} does not exist, so the standard maximum-likelihood program is not well-formed. $\hat{D}$ is an estimate based upon the training data, formed by calculating the maximum likelihood estimators $\hat{C}$ and $\hat{Q}$ and then calculating $\hat{D} = \hat{C}^T \hat{Q}^{-1} \hat{C}$. This problem can be solved analytically:

$$D_{[2:3,2:3]}^{*} = \frac{1}{2} \begin{bmatrix} \hat{D}_{[2,2]} + \hat{D}_{[3,3]} & 0 \\ 0 & \hat{D}_{[2,2]} + \hat{D}_{[3,3]} \end{bmatrix}.$$

With this estimate of $C^T Q^{-1} C$, $C^T Q^{-1}$ can be estimated by regression.

References

References

[1] Chapin J., Moxon K., and Markowitz R. Real-time control of a robot arm using simultaneously recorded neurons in the motor cortex. *Nature neuroscience*, 2(7), 1999.

[2] Gage G. J., Ludwig K. A., Otto K. J., Ionides E. L., and Kipke D. R. Naive coadaptive cortical control. *J. Neural Eng.*, 2(2):52–63, June 2005.

[3] Koralek A. C., Jin X., Long J. D., Costa R. M., and Carmena J. M. Corticostriatal plasticity is necessary for learning intentional neuroprosthetic skills. *Nature*, 483(7389):331–5, March 2012.

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

[5] Taylor D. M., Tillery S. I. H., and Schwartz A. B. Direct cortical control of 3D neuroprosthetic devices. *Science*, 296(5574):1829–32, June 2002.

[6] Carmena J. M., Lebedev M. A., Crist R. E., O'Doherty J. E., Santucci D. M., Dimitrov D. F., Patil P. G., Henriquez C. S., and Nicolelis M. A. L. Learning to control a brain-machine interface for reaching and grasping by primates. *PLoS Biology*, 1(2):E42, November 2003.

[7] Musallam S., Corneil B. D., Greger B., Scherberger H., and Andersen R. A. Cognitive control signals for neural prosthetics. *Science*, 305(5681):258–62, July 2004.

[8] Santhanam G., Ryu S. I., Yu B. M., Afshar A., and Shenoy K. V. A high-performance brain-computer interface. *Nature*, 442(7099):195–8, July 2006.

[9] Velliste M., Perel S., Spalding M. C., Whitford A. S., and Schwartz A. B. Cortical control of a prosthetic arm for self-feeding. *Nature*, 453(7198):1098–101, June 2008.

[10] Moritz C. T., Perlmutter S. I., and Fetz E. E. Direct control of paralysed muscles by cortical neurons. *Nature*, 456(7222):639–42, December 2008.

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

[12] Suminski A. J., Tkach D. C., Fagg A. H., and Hatsopoulos N. G. Incorporating feedback from multiple sensory modalities enhances brain-machine interface control. *J. Neurosci.*, 30(50):16777–87, December 2010.

[13] Ethier C., Oby E. R., Bauman M. J., and Miller L. E. Restoration of grasp following paralysis through brain-controlled stimulation of muscles. *Nature*, 485(7398):368–71, May 2012.

[14] Hochberg L. R., Serruya M. D., Friehs G. M., Mukand J. A., Saleh M., Caplan A. H., Branner A., Chen D., Penn R. D., and Donoghue J. P. Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature*, 442(7099):164–71, July 2006.

[15] Kim S.-P., Simeral J. D., Hochberg L. R., Donoghue J. P., and Black M. J. Neural control of computer cursor velocity by decoding motor cortical spiking activity in humans with tetraplegia. *J. Neural Eng.*, 5(4):455–76, December 2008.

[16] Hochberg L. R., Bacher D., Jarosiewicz B., Masse N. Y., Simeral J. D., Vogel J., Haddadin S., Liu J., Cash S. S., van der Smagt P., and Donoghue J. P. Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature*, 485(7398):372–5, May 2012.

[17] Gilja V., Nuyujukian P., Chestek C. A., Cunningham J. P., Yu B. M., Fan J. M., Churchland M. M., Kaufman M. T., Kao J. C., Ryu S. I., and Shenoy K. V. A high-performance neural prosthesis enabled by control algorithm design. *Nature Neuroscience*, 15(12):7–10, November 2012.

[18] Orsborn A. L., Dangi S., Moorman H. G., and Carmena J. M. Closed-loop decoder adaptation on intermediate time-scales facilitates rapid BMI performance improvements independent of decoder initialization conditions. *IEEE Trans. Neural Syst. Rehabil. Eng.*, 20(4):468–77, July 2012.

[19] Jarosiewicz B., Chase S. M., Fraser G. W., Velliste M., Kass R. E., and Schwartz A. B. Functional network reorganization during learning in a brain-computer interface paradigm. *Proc Natl Acad Sci USA*, 105(49):19486–19491, December 2008.

[20] Koyama S., Chase S. M., Whitford A. S., Velliste M., Schwartz A. B., and Kass R. E. Comparison of brain-computer interface decoding algorithms in open-loop and closed-loop control. *J. Comput. Neurosci.*, 29(1-2):73–87, August 2010.

[21] Chase S. M., Schwartz A. B., and Kass R. E. Bias, optimal linear estimation, and the differences between open-loop simulation and closed-loop performance of spiking-based brain-computer interface algorithms. *Neural networks*, 22(9):1203–13, November 2009.

[22] Li Z., O'Doherty J. E., Lebedev M. A., and Nicolelis M. A. L. Adaptive decoding for brain-machine interfaces through Bayesian parameter updates. *Neural computation*, 23(12):3162–204, December 2011.

[23] Shpigelman L., Lalazar H., and Vaadia E. Kernel-ARMA for Hand Tracking and Brain-Machine Interfacing During 3D Motor Control. *Neural Computation*, pages 1–8.

[24] Mahmoudi B. and Sanchez J. C. A symbiotic brain-machine interface through value-based decision making. *Plos One*, 6(3):e14760, January 2011.

[25] Wu W., Black M. J., Gao Y., Bienenstock E., Serruya M., Shaikhouni A., and Donoghue J. Neural decoding of cursor motion using a Kalman filter. In *Advances in Neural Information Processing Systems*, volume 15, page 133. The MIT Press, 2003.

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

[27] Hayes M. H. *Statistical Digital Signal Processing and Modeling*. Wiley, March 1996.

[28] MacKenzie I. S., Kauppinen T., and Silfverberg M. Accuracy measures for evaluating computer pointing devices. *Proc. of SIGCHI Conf. on Human factors in Comp. Sys.*, 2001.

[29] Malik W., Truccolo W., Brown E., and Hochberg L. Efficient decoding with steady-state Kalman filter in neural interface systems. *IEEE Trans. Neural Syst. Rehabil. Eng.*, 19(99):1–1, February 2011.

[30] Wessberg J., Stambaugh C. R., Kralik J. D., Beck P. D., Laubach M., Chapin J. K., Kim J., Biggs S. J., Srinivasan M. A., and Nicolelis M. A. Real-time prediction of hand trajectory by ensembles of cortical neurons in primates. *Nature*, 408(6810):361–5, November 2000.

[31] Boyd S. and Vandenberghe L. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

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

[33] Ganguly K. and Carmena J. M. Neural Correlates of Skill Acquisition with a Cortical Brain-Machine Interface. *Journal of Motor Behavior*, (September 2012):37–41, 2010.

[34] Moran D. W. and Schwartz A. B. Motor Cortical Representation of Speed and Direction During Reaching. *J. Neurophysiol.*, pages 2676–2692, 1999.

[35] Cunningham J. P., Nuyujukian P., Gilja V., Chestek C., Ryu S. I., and Shenoy K. V. A closed-loop human simulator for investigating the role of feedback control in brain-machine interfaces. *J. Neurophysiol.*, pages 1932–1949, 2011.

[36] Srinivasan L., Eden U. T., Willsky A. S., and Brown E. N. A state-space analysis for reconstruction of goal-directed movements using neural signals. *Neural computation*, 18(10):2465–94, October 2006.

[37] Wu W., Kulkarni J., Hatsopoulous N. G., and Paninski L. Neural Decoding of Hand Motion Using a Linear State-Space Model With Hidden States. *IEEE Trans. Neural Syst. Rehabil. Eng.*, 17(4):370–378, 2009.

[38] Yu B. M., Kemere C. T., Santhanam G., Afshar A., Ryu S. I., Meng T. H., Sahani M., and Shenoy K. V. Mixture of trajectory models for neural decoding of goal-directed movements. *J. Neurophysiol.*, 97(5):3763–80, May 2007.

[39] Fitts P. M. The information capacity of the human motor system in controlling the amplitude of movement. 1954. *Journal of experimental psychology. General*, 121(3):262–9, September 1992.

[40] Linderman M. D., Santhanam G., Kemere C. T., Gilja V., O'Driscoll S., Yu B. M., Afshar A., Ryu S. I., Shenoy K. V., and Meng T. H. Signal Processing Challenges for Neural Prostheses. *IEEE Signal Processing Magazine*, (January):18–28, 2008.

[41] Grant M. and Boyd S. Graph implementations for nonsmooth convex programs. In Blondel V., Boyd S., and Kimura H., editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008.