Feedback control policies employed by people using intracortical brain–computer interfaces

Francis R Willett, Chethan Pandarinath, Beata Jarosiewicz, Brian A Murphy, William D Memberg, Christine H Blabe, Jad Saab, Benjamin L Walter, Jennifer A Sweet, Jonathan P Miller, Jaimie M Henderson, Krishna V Shenoy, John D Simons, Leigh R Hochberg, Robert F Kirsch and A Bolu Ajiboye

1 Department of Biomedical Engineering, Case Western Reserve University, Cleveland, OH, USA
2 Louis Stokes Cleveland Department of Veterans Affairs Medical Center, FES Center of Excellence, Rehab. R&D Service, Cleveland, OH, USA
3 Department of Neurosurgery, Stanford University, Stanford, CA, USA
4 Department of Electrical Engineering, Stanford University, Stanford, CA, USA
5 Stanford Neurosciences Institute, Stanford University, Stanford, CA, USA
6 Department of Neuroscience, Brown University, Providence, RI, USA
7 Center for Neurorestoration and Neurotechnology, Rehabilitation R&D Service, Department of Veterans Affairs Medical Center, Providence, RI, USA
8 Brown Institute for Brain Science, Brown University, Providence, RI, USA
9 School of Engineering, Brown University, Providence, RI, USA
10 Department of Neurology, University Hospitals Case Medical Center, Cleveland, OH, USA
11 Department of Neurosurgery, University Hospitals Case Medical Center, Cleveland, OH, USA
12 Department of Neurobiology, Stanford University, Stanford, CA, USA
13 Department of Bioengineering, Stanford University, Stanford, CA, USA
14 Howard Hughes Medical Institute, Stanford University, Stanford, CA, USA
15 Department of Neurology, Massachusetts General Hospital, Boston, MA, USA
16 Department of Neurology, Harvard Medical School, Boston, MA, USA

E-mail: frw9@case.edu

Received 20 July 2016, revised 13 October 2016
Accepted for publication 10 November 2016
Published 30 November 2016

Abstract

Objective. When using an intracortical BCI (iBCI), users modulate their neural population activity to move an effector towards a target, stop accurately, and correct for movement errors. We call the rules that govern this modulation a ‘feedback control policy’. A better understanding of these policies may inform the design of higher-performing neural decoders. Approach. We studied how three participants in the BrainGate2 pilot clinical trial used an iBCI to control a cursor in a 2D target acquisition task. Participants used a velocity decoder with exponential smoothing dynamics. Through offline analyses, we characterized the users’ feedback control policies by modeling their neural activity as a function of cursor state and target position. We also tested whether users could adapt their policy to different decoder dynamics by varying the gain (speed scaling) and temporal smoothing parameters of the iBCI. Main results. We demonstrate that control policy assumptions made in previous studies do not fully describe the policies of our participants. To account for these discrepancies, we propose a new model that captures (1) how the user’s neural population activity gradually declines as the cursor approaches the target from afar, then decreases more sharply as the cursor comes into contact with the target, (2) how the user makes constant feedback corrections even when the cursor is on top of the target, and (3) how the user actively accounts for the cursor’s current velocity to avoid
overshooting the target. Further, we show that users can adapt their control policy to decoder dynamics by attenuating neural modulation when the cursor gain is high and by damping the cursor velocity more strongly when the smoothing dynamics are high. **Significance.** Our control policy model may help to build better decoders, understand how neural activity varies during active iBCI control, and produce better simulations of closed-loop iBCI movements.

**Keywords:** brain–computer interface, motor cortex, motor control

[Online supplementary data available from stacks.iop.org/JNE/14/016001/mmedia](http://stacks.iop.org/JNE/14/016001/mmedia)

(Some figures may appear in colour only in the online journal)

---

**Introduction**

Intracortical brain–computer interfaces (iBCIs) can restore movement and communication to people with paralysis by recording movement–related neural activity from the cortex and translating it into command signals for an external device, such as a computer cursor or a robotic arm [1–8]. Much of the literature to date has focused on improving iBCI performance by changing how neural activity is translated into motion (i.e. ‘decoding’) [2, 4, 7–16]. However, less effort has been spent on understanding the details of how people modulate their neural activity to actively control the brain–computer interface. In other words, what rules govern how a person’s neural activity varies to enable them to move an effector towards their target, stop accurately, and correct for movement errors? Here, we investigated how three participants in the BrainGate2 pilot clinical trial modulated their neural activity, using imagined or attempted movement, to control a 2D computer cursor. Our goal was to identify the user’s ‘control policy’, a mathematical function describing how the user’s neural population activity changes as a function of the target position and the state of the computer cursor (its position and velocity). Note that when we use the phrase ‘control policy’ we do not mean to imply that the user has a conscious, explicit strategy for controlling the iBCI.

There are many possible control policies the user could take to move the cursor towards the target, several of which have been proposed for the purposes of computer modeling and decoding [1, 4, 7, 8, 14, 16–20]. A number of studies have proposed linear control policies based on models of optimal control to describe the user’s control policy [15, 17–20]. While these models have been proven useful for decoding algorithms and can reproduce some aspects of iBCI control, their core assumption of linearity has not, to our knowledge, been rigorously validated and compared to a wider class of potentially non-linear policies.

Other studies have made a more limited set of heuristic assumptions for the purposes of decoder calibration. For example, researchers have assumed that (on average) the user’s neural population activity is zero when the cursor is on top of the target (ReFit) [4, 8, 16], or that, when the cursor is far from the target, the population activity is equal to a unit vector pointing straight from the cursor to the target [7, 14]. While these are useful heuristics for calibrating decoders, it is not entirely clear that these assumptions accurately describe the user’s neural activity. Moreover, these heuristics do not fully model the direction and magnitude of the user’s population activity both near and far from the target. On the whole, we still lack a complete picture of the user’s control policy that has been rigorously validated against competing alternatives. Since recent reports have showed significant performance gains enabled by using better assumptions about the user’s control policy to calibrate decoders [4, 8, 16, 17, 21], we expect that more refined knowledge of the user’s control policy could continue to improve performance.

Additionally, the extent to which users can adapt their control policy to account for different end effector dynamics is not well known. It has been shown that non-human primates can adapt to perturbations in the decoder that cause a visuomotor rotation when using an iBCI [22, 23]. Are similar adaptations possible to different cursor gains and to the acceleration dynamics caused by high amounts of velocity smoothing? Human operator studies have demonstrated that people can adapt to these kinds of dynamics when controlling devices with natural movements [24–26]. However, it is unknown if the same adaptability occurs when people are controlling a system directly with motor cortical activity without somatosensory feedback.

To gain a deeper understanding of how people control and adapt to iBCIs, we performed offline analyses of the neural population activity recorded while study participants controlled a cursor to acquire targets using our iBCI. Our approach was to use a flexible control policy model to describe the neural activity. The model makes limited assumptions about the participant’s control policy and can converge to other policies proposed in the literature if they are consistent with the data. After fitting the model, we examined the resultant parameters to discover which aspects of the participants’ control policies matched previous assumptions and which did not. We used the same model to understand how each participant adapted their control policy to different decoder dynamics.

For this study, we used a steady-state Kalman filter to enable real time control of cursor velocity. The steady-state Kalman filter is essentially a linear decoder with exponential smoothing dynamics; since this is a very common decoding approach [1, 2, 4–8, 27], our results should have broad applicability. To facilitate our investigation, we reparameterized the Kalman filter to explicitly separate it into two pieces: a dimensionality reduction step that maps the user’s high-dimensional neural activity to a 2D ‘control’ vector, plus a second-order dynamical system that defines how the cursor moves in
response to the control vector (figure 1). The dimensionality
reduction done by the Kalman filter implies that only neural
modulation lying within a 2D ‘output-potent’ subspace has
any effect on the motion of the cursor. We therefore focused
our investigation on understanding this lower dimensional
projection of the neural activity, the control vector.

The effect of the control vector on cursor movement has
an intuitive physical interpretation: it applies a ‘force’ to the
cursor which pushes it in the direction of the control vector.
The gain (speed scaling) and smoothing dynamics of the
Kalman filter determine how the cursor moves in response to
that force. A high amount of smoothing gives the cursor more
‘inertia’ and makes it take longer to accelerate and decelerate,
while a low amount of smoothing makes it quick and responsive
(at the cost of higher frequency noise in the movement).

To understand how the user pushed the cursor towards
the target, we fit the relationship between the online decoded
control vectors, the target position, and the cursor state using
piecewise linear functions that can express a wide class of pos-
sibly non-linear control policies. To validate our results, we
compared the accuracy of the piecewise model to previously
proposed control policies in terms of their ability to predict
the online decoded control vector. We show that the piecewise
model has the highest cross-validated prediction accuracy.
We also inserted the piecewise control policy model into a
subject-specific, feedback control model of 2D cursor move-
ments to demonstrate that it simulates movements accurately
matching those of the participant. The simulation results give
confidence that the piecewise model is capturing the function-
ally significant elements of the user’s control policy.

Lastly, we note that while one motivation for better under-
standing the user’s control policy is to enable more accurate
decoder calibration, decoder performance is not the main focus
of the study. Our study is not about testing and improving
decoder performance, it is about better understanding the
user’s control policy. Thus, all of the results we present are
from an offline analysis of what was decoded during online,
closed-loop iBCI operation.

**Methods**

Permission for these studies was granted by the US
Food and Drug Administration (Investigational Device
Exemption #G090003) and the Institutional Review Boards
of University Hospitals (protocol #04-12-17), Stanford
University (protocol #20804), Partners Healthcare/
Massachusetts General Hospital (2011P001036), Providence
VA Medical Center (2011-009), and Brown University
(0809992560). All participants were enrolled in a pilot
clinical trial of the BrainGate Neural Interface System (www.
clinicaltrials.gov/ct2/show/NCT00912041). Informed consent,
including consent to publish, was obtained in writing from the
participants prior to their enrollment in the study.

**Participants**

Participants were implanted with one (T6) or two (T7, T8)
96 channel intracortical microelectrode arrays (Blackrock
Microsystems, Salt Lake City, UT) in the hand area of domi-
nant motor cortex (1.0 mm electrode length for T6, 1.5 mm
length for T7 and T8). Participant T6 is a 52 year-old woman
with tetraplegia and diagnosed with ALS. We collected 7 ses-
sions with T6 that took place between T6’s trial days 699–
788. Participant T7 is a 59 year-old man with tetraplegia and
diagnosed with ALS. We collected 2 sessions with T7 that
took place on T7’s trial days 406 and 409. Participant T8 is
a 53 year-old man with tetraplegia due to high cervical spinal
cord injury. We collected 5 sessions from T8 that took place
between T8’s trial days 57–418. More surgical details can be
found in [2, 8, 28].

**Study design and center-out-back task**

Participants T6, T7, and T8 completed 7, 2, and 5 sessions
(respectively) of neural control of a computer cursor in a 2D
center-out-back task (supplemental videos 1–3 (stacks.iop.org/JNE/14/016001/mmedia)). Each session lasted between 2 and 4 h and contained an average of 1228 cursor movements. To control the cursor, participants were instructed to imagine moving the thumb and index finger (T6), imagine moving a computer mouse placed under the hand (T7), or attempt to make arm movements (T8).

Each session began with an ‘open-loop’ block in which participants watched the cursor automatically move to the targets while imagining or attempting to control the cursor. We used the neural data collected during this block to initialize the decoding matrix. Then, participants completed a series of ‘closed-loop’ neural control blocks with computer assistance that were used to recalibrate the decoder (similar to Jarosiewicz et al [14]). Finally, the decoding matrix was fixed and participants completed a series of 5 min closed-loop blocks with no computer assistance, each block with a different cursor speed and/or smoothing setting.

Gain and smoothing settings were selected to span a variety of possible cursor dynamics (slow movements, fast movements, jittery but responsive movements, smooth but less responsive movements with long acceleration and deceleration times, etc). In sessions testing a variety of parameters, we used our feedback control model before the session to select a cloud of points that were predicted to be centered on an optimally performing gain and smoothing setting. In high speed or high smoothing sessions, we varied either gain or smoothing dynamics more widely to span a large range of values. Figure 2 illustrates example impulse response functions of the cursor dynamics under different gain and smoothing settings and illustrates the gain and smoothing settings chosen for each block.

Participants acquired targets by holding the cursor in constant contact with the target region for a specified dwell time (ranging from 0.15 to 2 s). If the target was not acquired within 8 or 10 s (depending on the session), the trial was considered unsuccessful and the cursor was reset to the target position. After a target was

Figure 2. Cursor dynamics. (A)–(B) Impulse responses of the cursor dynamics to a control vector impulse of unit magnitude applied at the dashed vertical line. Colored lines illustrate impulse responses under different smoothing (A) and gain (B) parameters and give a sense of how the gain and smoothing parameters alter the cursor dynamics. The smoothing parameter \( \alpha \) controls the amount of first-order, low-pass filtering applied to the decoded control vector and the gain parameter \( \beta \) scales the decoded control vector. (C) Imposed cursor dynamics for all 114 blocks reported in the study (N = 50 for T6, 20 for T7, and 44 for T8) plotted on the gain/smoothing parameter surface. Repeated blocks with identical settings are marked with + symbols.
acquired, the next target appeared after a 200–300 ms delay (T6, T7) or immediately afterwards (T8). Supplemental videos 1–3 show the task in action. Table S1 in the supplement summarizes the task parameters used for each session.

Online decoding framework

Dimensionality reduction. Our signal processing and neural feature extraction methods followed closely those of Jarosiewicz et al [7]. Every 20 ms time step, threshold crossing counts and power in the spike frequency band (250–5000 Hz) were computed for each channel. Threshold crossing counts were defined as the number of times that the voltage time series on a given channel crossed a channel-specific threshold from above (−3.25 × RMS for T6, who did not have many discernable single-unit spikes, and −4.5 × RMS for T7 and T8, who did). High frequency power was defined as the root mean square of the signal in the spike band. These neural features were then concatenated into a 2N×1 feature vector (where N is the number of microelectrode array channels).

At each time step, we mapped the (z-scored) neural features to a decoded control vector with the equation

\[ u_t = Df_t, \]

where \( f_t \) is a 2N×1 neural feature vector, \( D \) is a 2×2N decoding matrix, and \( u_t \) is a 2×1 decoded control vector. The \( u_t \) vector that we decoded online was the control vector, the main object of our investigation. Below, we describe how we modeled \( u_t \) as a function of target position and cursor state in offline analysis to retrospectively characterize the user’s control policy during online control. Note that we did not use a separate decoding matrix for offline analysis; the decoding matrix for offline analysis and online control was identical.

To account for changes in the mean value of neural features over time that can cause an offset (bias) to appear in \( u_t \) [29], we updated our estimate of the feature means over the course of the session using methods similar to Jarosiewicz et al [7]. We also used a ‘bias corrector’ during blocks of closed-loop control to detect and remove biases in \( u_t \) not accounted for by updating the feature means. See section 9 in the supplement for a full account of online and offline methods used to mitigate the effects of non-stationary feature means.

Decoder calibration. The matrix \( D \) was solved for using either ‘reverse regression’ with T6 and T7 [30], or ‘full OLE’ with T8 [11]. When solving for \( D \), we made the assumption that \( u_t \) pointed straight from the cursor to the target at each time step with a constant magnitude of 1. Note that this assumption did not influence the coefficients of the decoder so as to cause the decoder output to be biased towards that assumption, nor did it cause the constant magnitude control policy to be more optimal for operating the decoder (section 10 of the supplement).

To build the decoder, we used all open loop and closed-loop calibration data (2–7 blocks, each 5 min long). To exclude data where the user’s control vector may have decreased in magnitude near the target (violating the assumption that the control vector is of constant magnitude), we excluded time steps after the first 1.8 (T6, T7) or 1.5 (T8) s of the trial. We also excluded time steps that occurred during each participant’s reaction time period (supplementary figure S1).

Cursor dynamics. The decoded control vector influenced cursor motion at each time step through the following dynamical equation

\[ \mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t, \]

where \( \mathbf{x}_t \) is the cursor state data and \( \mathbf{u}_t \) is the decoded control vector. The \( \mathbf{u}_t \) vector that we decoded online was the control vector, the main object of our investigation. Below, we describe how we modeled \( \mathbf{u}_t \) as a function of target position and cursor state in offline analysis to retrospectively characterize the user’s control policy during online control. Note that we did not use a separate decoding matrix for offline analysis; the decoding matrix for offline analysis and online control was identical.

To account for changes in the mean value of neural features over time that can cause an offset (bias) to appear in \( \mathbf{u}_t \) [29], we updated our estimate of the feature means over the course of the session using methods similar to Jarosiewicz et al [7]. We also used a ‘bias corrector’ during blocks of closed-loop control to detect and remove biases in \( \mathbf{u}_t \) not accounted for by updating the feature means. See section 9 in the supplement for a full account of online and offline methods used to mitigate the effects of non-stationary feature means.

Decoder calibration. The matrix \( D \) was solved for using either ‘reverse regression’ with T6 and T7 [30], or ‘full OLE’ with T8 [11]. When solving for \( D \), we made the assumption that \( \mathbf{u}_t \) pointed straight from the cursor to the target at each time step with a constant magnitude of 1. Note that this assumption did not influence the coefficients of the decoder so as to cause the decoder output to be biased towards that assumption, nor did it cause the constant magnitude control policy to be more optimal for operating the decoder (section 10 of the supplement).

To build the decoder, we used all open loop and closed-loop calibration data (2–7 blocks, each 5 min long). To exclude data where the user’s control vector may have decreased in magnitude near the target (violating the assumption that the control vector is of constant magnitude), we excluded time steps after the first 1.8 (T6, T7) or 1.5 (T8) s of the trial. We also excluded time steps that occurred during each participant’s reaction time period (supplementary figure S1).

Cursor dynamics. The decoded control vector influenced cursor motion at each time step through the following dynamical equation

\[ \mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t, \]

where \( \mathbf{x}_t \) is the cursor state data and \( \mathbf{u}_t \) is the decoded control vector. The \( \mathbf{u}_t \) vector that we decoded online was the control vector, the main object of our investigation. Below, we describe how we modeled \( \mathbf{u}_t \) as a function of target position and cursor state in offline analysis to retrospectively characterize the user’s control policy during online control. Note that we did not use a separate decoding matrix for offline analysis; the decoding matrix for offline analysis and online control was identical.

To account for changes in the mean value of neural features over time that can cause an offset (bias) to appear in \( \mathbf{u}_t \) [29], we updated our estimate of the feature means over the course of the session using methods similar to Jarosiewicz et al [7]. We also used a ‘bias corrector’ during blocks of closed-loop control to detect and remove biases in \( \mathbf{u}_t \) not accounted for by updating the feature means. See section 9 in the supplement for a full account of online and offline methods used to mitigate the effects of non-stationary feature means.
The number of breakpoints was chosen to balance fidelity with overfitting, and we confirmed that the model gives the same result over a wide range of breakpoint numbers (6–48; see section 12 of the supplement). We constrained the coefficients of the forward model function to be negative, assuming that positive values, which correspond to underdamped and possibly unstable control policies, would only arise as an artifact of correlations between decoding error and the cursor state (see section 8 of the supplement for details).

Note that while the entire control vector equation is a function of target position and the IME of cursor position and velocity, the \( f_{\text{arg}} \) and \( f_{\text{vel}} \) functions themselves are functions only of target distance and speed. Thus, the model can only describe a restricted subset of all possible functions of target position, cursor position and cursor velocity. This restriction was intentional and makes the model coefficients easier to interpret, while still being sufficiently flexible to generalize other models in the literature.

In addition to the full model, we also considered special cases of the piecewise model that offer an informative point of comparison to the full model, as well as different control policy models that test assumptions used in previous studies (table 1). In the table, scalar model coefficients for the constant magnitude, linear, and position error models are specified as a and b. Figures (D) and (E) illustrates what \( f_{\text{arg}} \) and \( f_{\text{vel}} \) would look like under these different control policy models.

### Feedback delay and state estimation

We include a visual feedback delay and a forward model that the user employs to estimate the current cursor state \( x_t \). At each time step, the user receives perfect knowledge of the delayed cursor state \( x_{t-\tau} \), where \( t \) is the current time step and \( \tau \) is the visual feedback delay (in \# of time steps). After receiving delayed feedback, the user employs a forward model (matched perfectly to the cursor dynamics), combined with knowledge of previously issued control signals, to estimate the cursor state \( x_t \).

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>The encoded control vector is a linear function of position error and velocity [15, 17, 20]: ( c_t = a(g_t - \bar{p}_t) + b\bar{v}_t )</td>
</tr>
<tr>
<td>Finite</td>
<td>The encoded control vector is a time-varying, linear function of position error and velocity that optimizes a quadratic cost function (as predicted by the finite horizon, linear quadratic Gaussian optimal control model [18, 19]): ( c_t = a(g_t - \bar{p}_t) + b\bar{v}_t )</td>
</tr>
<tr>
<td>LQG</td>
<td>The encoded control vector is equal to a vector of constant magnitude that points from the cursor to the target: ( c_t = a(g_t - \bar{p}_t) + b\bar{v}_t )</td>
</tr>
<tr>
<td>Constant magnitude</td>
<td>The constant magnitude assumption has been used to calibrate decoders from snippets of data when the cursor is far from the target [5, 7, 14].</td>
</tr>
<tr>
<td>Position error</td>
<td>The encoded control vector is a function of position error alone [1]: ( c_t = a(g_t - \bar{p}_t) )</td>
</tr>
<tr>
<td>Piecewise deadzone</td>
<td>The encoded control vector is equal to zero when the cursor is overlapping with the target (where the cursor radius plus target radius is equal to ( r )): (</td>
</tr>
<tr>
<td>Piecewise no velocity</td>
<td>The encoded control vector is a function of target distance only, and does not account for cursor velocity [1]: ( c_t = a(g_t - \bar{p}_t) )</td>
</tr>
</tbody>
</table>

and does not use neural activity recorded during the time bin that is being predicted. The entire model (the user, noise, and cursor dynamics blocks together) can also be used as a generative model that can simulate new cursor movements. We inferred the user’s feedback control policy by examining which control policy assumptions enabled the best predictions and simulations of what occurred during online neural control. Section 13 in the supplement provides more details about model fitting, prediction, and simulation.

**Control policy.** We modeled the decoded control vector \( u_t \) at each time step as the sum of the user’s underlying, ‘encoded’ control vector \( c_t \) and decoding noise \( e_t \):

\[
u_t = c_t + e_t.
\]

To describe the user’s encoded control vector \( c_t \), we used a model that generalizes assumptions made in previous studies (so that it can reduce to those models if they are correct) but also expresses a wider set of possible control policies. Namely, we used a piecewise model that describes the user’s encoded control vector as a non-linear function of target position and the user’s internal model estimate (IME) of cursor position and velocity:

\[
c_t = \frac{g_t - \bar{p}_t}{||g_t - \bar{p}_t||} f_{\text{arg}} ||g_t - \bar{p}_t|| + \frac{\bar{v}_t}{||\bar{v}_t||} f_{\text{vel}} ||\bar{v}_t||,
\]

where \( g_t \) is the target position, \( \bar{p}_t \) is the IME of cursor position, \( \bar{v}_t \) is the IME of cursor velocity, and \( f_{\text{arg}} \) and \( f_{\text{vel}} \) are nonlinear, 1D weighting functions that we fit empirically. Essentially, this equation asserts that \( c_t \) is the sum of a point-at-target vector (weighted by the function \( f_{\text{arg}} \)) and a velocity damping vector (weighted by \( f_{\text{vel}} \)). The meaning of \( f_{\text{arg}} \) and \( f_{\text{vel}} \) is illustrated in figure 4(A) (and discussed more in results).

We parameterized \( f_{\text{arg}} \) and \( f_{\text{vel}} \) as continuous, piecewise linear functions with 12 breakpoints set at the (0, 8.33, 16.66,... 100) percentiles for the IMEs of target distance (\( ||g_t - \bar{p}_t|| \)) and cursor speed (\( ||\bar{v}_t|| \)). The number of breakpoints was chosen to balance fidelity with overfitting, and we confirmed that the model gives the same result over a wide range of breakpoint numbers (6–48; see section 12 of the supplement). We constrained the coefficients of \( f_{\text{vel}} \) to be negative, assuming that positive values, which correspond to underdamped and possibly unstable control policies, would only arise as an artifact of correlations between decoding error and the cursor state (see section 8 of the supplement for details).

Note that while the entire control vector equation is a function of target position and the IME of cursor position and velocity, the \( f_{\text{arg}} \) and \( f_{\text{vel}} \) functions themselves are functions only of target distance and speed. Thus, the model can only describe a restricted subset of all possible functions of target position, cursor position and cursor velocity. This restriction was intentional and makes the model coefficients easier to interpret, while still being sufficiently flexible to generalize other models in the literature.

In addition to the full model, we also considered special cases of the piecewise model that offer an informative point of comparison to the full model, as well as different control policy models that test assumptions used in previous studies (table 1). In the table, scalar model coefficients for the constant magnitude, linear, and position error models are specified as a and b. Figures 4(D) and (E) illustrates what \( f_{\text{arg}} \) and \( f_{\text{vel}} \) would look like under these different control policy models.
The decoding error $e_t$ is characterized with an autoregressive noise model

$$e_t = \Pi_1 e_{t-1} + \Pi_2 e_{t-2} + \ldots + \Pi_p e_{t-p} + \varepsilon_t,$$

where $\Pi_i$ are $2 \times 2$ matrices, $p$ is the number of time lags in the model, and $\varepsilon_t$ is zero-mean, multivariate Gaussian i.i.d. noise. In addition to describing the noise magnitude, the autoregressive model describes the frequency content of the noise by parameterizing correlations and anti-correlations in time.

**Model fitting.** The control policy model and the IMEs were fit together in an iterative process:

1. Initialize the IMEs to a delayed cursor state ($\hat{x}_t = x_{t-\tau}$).
2. Using the current IMEs, fit the control policy model parameters using least squares regression to minimize the error between $c_t$ and $u_t$. When fitting the finite horizon LQG model, we used an iterative search to find the best fitting cost function parameters (see section 11 of the supplement).

**Figure 3.** Block diagram of the feedback control model used in the study (A) and example data (B)–(D). (A) At each time step, the user receives delayed visual feedback of the cursor state ($p_{t-\tau}, v_{t-\tau}$). The delayed state is used as input to the user’s internal forward model that estimates the current cursor state ($\hat{p}_t, \hat{v}_t$). The user then outputs a control vector $c_t$ as a function of $\hat{p}_t$ and $\hat{v}_t$ (target position) that is intended to drive the cursor towards the target. We model the decoded control vector $u_t$, as the sum of $c_t$ and a corrupting noise vector $e_t$. Finally, the decoded control vector $u_t$ is then passed through the Kalman filter dynamics (i.e. scaled, smoothed with a first order filter, and applied to the cursor as a velocity vector $v_t$). (B)–(D) Example data from one of T7’s cursor movements, where we have used the model to decompose $u_t$ (which we directly observed) into $c_t$ and $e_t$ (which we inferred). Panel (B) shows the cursor trajectory (black line, cursor begins at the top and moves downwards towards the target) with $u_t$ illustrated at each time point as a blue arrow. Panel (C) shows the same movement but with gold arrows illustrating $c_t$. Time series in (D) show the X and Y components of $u_t$, $c_t$, $v_t$, and $e_t$. 

\[ \hat{x}_t = A' x_{t-\tau} + \sum_{i=0}^{\tau-1} A' B c_{t-i-1}, \]

where $\hat{x}_t$ is the user’s internal model estimate of the cursor state ($\hat{x}_t$ is a vector containing $\hat{p}_t$ and $\hat{v}_t$). Since we assume that the user is unaware of the noise added to their control signal for all time steps from $(t - \tau - 1)$ to $(t - 1)$, $\hat{x}_t$ will differ from $x_t$. $\hat{x}_t$ cannot be directly observed but can be estimated from the data in an iterative process. We set $\tau$ equal to each subject’s reaction time (supplementary figure S1).
3. Using the current control policy model, update the IMEs using the forward model equation.
4. Return to step 2 until we have completed five iterations.

Once the control policy model was fit, the noise model was then fit to characterize the error time series $u_t - c_t$ using least squares regression.

**Model prediction.** To predict the decoded control vector time series using a history of target positions and cursor states, we stepped through the data using the control policy model and forward model equation one step at a time:

1. Initialize previous control vector predictions to zero ($c_i = 0$ for $t < 1$) and begin at time step $t = 1$.
2. Compute the current IME for time step $t$ using the forward model equation.
3. Compute the control vector prediction for time step $t$ using the control policy model and the internal model estimate $\hat{x}_t$.
4. Advance the time step forward and return to step 2 if not finished.

Note that this prediction procedure implies that each control vector prediction is a function of all previous cursor states and target positions.

**Cursor trajectory metrics**

We used three metrics to quantify different aspects of observed and simulated cursor movements: translation time, dial-in time, and path efficiency. Translation time was defined as the time between the start of the movement (target appearance) and the first time the cursor touched the target. Dial-in time was defined as the time between when the cursor first touched the target and completion, minus the obligatory dwell time [4]. Dial-in times greater than zero indicate that the cursor exited the target at least once before trial completion. Path efficiency was defined as the path length of a perfectly straight movement from the cursor start position to the target divided by the path length of the actual cursor trajectory. For computing the path efficiency, we ignored the dwell time.
Results

Overview

Participants T6, T7, and T8 used our iBCI to control the motion of a computer cursor in real time under a variety of imposed cursor gain and smoothing settings (figure 2). To infer the user’s feedback control policy, we used the feedback control model presented in figure 3.

Typical form of the user’s control policy

We give the details of our piecewise control policy model in Methods and illustrate it in figure 4(A). We assume that the control vector is the sum of a point-at-target vector (weighted by the function \( f_{\text{targ}} \)) and a velocity damping vector (weighted by \( f_{\text{vel}} \)). \( f_{\text{targ}} \) is a function of target distance and it scales a unit vector that points from the cursor to the target. \( f_{\text{targ}} \) can describe how the user applies a diminishing ‘push’ to the cursor as it approaches the target. \( f_{\text{vel}} \) is a function of cursor speed and it scales a unit vector pointing in the direction of the cursor velocity. \( f_{\text{vel}} \) can describe how the user counteracts velocities that are too fast or that point away from the target.

When smoothing is significant, this ‘damping’ behavior may be necessary to prevent overshooting and oscillating around the target. To estimate \( f_{\text{targ}} \) and \( f_{\text{vel}} \) for each block of data, we made minimal assumptions about their shape and fit them as continuous, piecewise linear functions with 12 breakpoints.

Figures 4(B) and (C) show the average shape of \( f_{\text{targ}} \) and \( f_{\text{vel}} \) for each participant. All three participants have a similar shape for \( f_{\text{targ}} \) that consists of a slow decline in magnitude when approaching the target from far away, followed by a steep decline in magnitude when near the target. \( f_{\text{targ}} \) is significantly non-linear (it saturates when the cursor is far from the target), in contrast to linear control policy models previously used for decoder calibration and simulation [1, 15, 17, 20, 21]. Further, \( f_{\text{targ}} \) is non-zero even when the cursor is in contact with the target (i.e. for target distances to the left of the ‘target edge’ lines in figure 4(B)), meaning that users continually push the cursor towards the center of the target even when the cursor is overlapping the target. This finding is in contrast to the target deadzone assumption made by the ReFIT decoder building process that assumes a control vector of zero when inside the target region [4, 16].

Figure 4(C) shows the average shape of \( f_{\text{vel}} \) for blocks where including \( f_{\text{vel}} \) significantly improved the ability of the piecewise model to predict the user’s decoded control vector when cross-validated (pairwise t-test on mean squared error, with average error within a trial as the statistical unit, \( p < 0.01 \)).

Validation of the piecewise control policy model

Cross-validated prediction accuracy. We used 10-fold cross-validation to compare the accuracy of the full piecewise model to models used in previous studies. For every fold, each model was fit on the training set (containing 90% of the trials) and then used to predict the decoded control vector from a time series of previous target positions, cursor positions and cursor velocities in the test set (containing 10% of the trials). Cross-validation ensures that any apparent performance increase is not caused by overfitting a more complex model. When cross-validating, data was not shuffled before splitting it into folds in order to keep its temporal structure intact. Fold breakpoints were always placed between trials so as not to split a trial between the training and test sets. Figure 5 shows the results of the cross-validation, which indicate that the full piecewise model accounts for more variance in the decoded control vector than any other model tested.

Generative simulation. While measuring cross-validated prediction accuracy is useful for comparing control policy models, it does not indicate whether the best performing model has captured enough aspects of the user’s true control policy to make for a plausibly complete model. In addition, prediction accuracy does not convey the functional significance of any particular feature of the user’s control policy. A model may perform better in terms of prediction error, although it may do so by modeling a certain feature of the user’s control policy that plays only a minimal role in shaping the dynamics of the cursor movements.

To address these concerns, we used the full feedback control model presented in figure 3 as a closed-loop, generative model to simulate cursor movements under different control policy assumptions. We fit both the control policy block.
We simulated 300 cursor movements per block to get a good estimate of the movement dynamics. Figure 6 compares the cursor trajectories made by participant T7 to trajectories simulated by our feedback control model (using the piecewise control policy model) for four example blocks with different cursor gain and smoothing settings. The simulated trajectories are visually similar to the observed trajectories, suggesting that the model is capturing the functionally significant dynamics of interest. Figure 7 quantifies the similarity for all blocks in the study. Though
the model has small biases for movements with low translation times and dial-in times, on the whole the model can simulate trajectories that are a close quantitative match to real trajectories across three different participants and 86 unique cursor dynamics settings. These results suggest that the model is capturing the key elements of the user’s control policy. Supplementary figures S3 and S4 show additional observed versus simulated trajectory comparisons for participants T6 and T8. Supplemental videos 1–3 show simulated movements for three example blocks.

In supplementary figure S7 we examine the simulation accuracy of the feedback control model under different control policy assumptions. The piecewise control policy model has the least absolute error and bias, indicating that it captures elements of the user’s control policy that not only increase prediction performance but also functionally shape the dynamics of the simulated cursor movements. Other candidate models tended to simulate cursor movements with less efficient paths, longer translation times, and longer dial-in times than the actual cursor movements, either because of smaller control vectors near the target (linear and deadzone models) or because of not modeling how the user accounts for cursor velocity to prevent overshooting the target (constant magnitude and no velocity models). Supplementary figures S5 and S6 illustrate these concepts further with example simulated trajectories under each different assumption.

Model fitting consistency. We considered (and ruled out) three issues that could negatively impact the consistency of our model fitting procedure: errors-in-variables (regression dilution), correlations between the decoding error and cursor state, and the effect of the forward model assumption. These are addressed in section 8 of the supplement.

Effect of decoder calibration assumption. We considered (and ruled out) two possible ways that the control policy assumption we made during decoder calibration could have biased our results. When calibrating the decoding matrix $D$, we made the assumption that the control vector pointed straight from the cursor to the target at each time step with a constant magnitude of 1.

First, we confirmed that this calibration assumption did not influence the coefficients of the decoder so as to cause the decoder output to be biased towards that assumption. Offline, we tested three different control policy assumptions for decoder calibration (constant magnitude, linear and piecewise) and found that the output of the decoder was not affected by what was assumed during calibration (the decoder output always took the piecewise linear form shown in figures 4(B) and (C)). A full account of this analysis is given in section 10 of the supplement.

Second, we confirmed that the calibration assumption did not cause the constant magnitude control policy to be more optimal for operating the decoder (thus potentially incentivizing the user to adopt this policy). To demonstrate this, we ran a simulation testing two different decoder calibration assumptions (linear versus piecewise) and two control policies taken by the user (linear versus piecewise). We found that the calibration assumption had no effect on the performance that resulted from using either the linear or piecewise control policy. Regardless of what was assumed during calibration, the piecewise policy led to the best performance (in terms of minimizing movement time). A full account of this simulation is given in section 10 of the supplement.

Changes in control policy as a function of cursor dynamics

Functionally significant adaptation to acceleration-like (second-order) cursor dynamics. When velocity smoothing is high, the cursor dynamics become more second-order and the cursor takes longer to accelerate and decelerate (i.e. it has high ‘inertia’). As inertia increases, it becomes more important for the user to counteract (‘damp’) the effects of velocities that are too fast or that point away from the target; otherwise, the cursor will overshoot and oscillate around the target. To test whether users could adapt their control policy under such conditions by adding a velocity damping component (modeled by $f_{vel}$), we purposefully collected blocks of data with significant second-order dynamics. As demonstrated earlier (figures 4 and 5), T6 and T8’s control policy did in fact show a significant $f_{vel}$ damping component during some subset of the blocks. However, it remains unclear whether this component
was large enough to play a functionally significant role in enabling T6 and T8 to stop accurately on top of the target.

To test the functional effect of the velocity damping component, we simulated trajectories using the piecewise model fit with or without an $f_{vel}$ term and quantified the difference in control quality. Figures 8(A)–(C) illustrate this procedure for a single example block with T6 in which significant second-order dynamics were imposed. With $f_{vel}$ (figure 8(B)), trajectories appear straight and stop accurately on top of the target, closely matching the observed trajectories (figure 8(A)). Without the $f_{vel}$ component (figure 8(C)), the cursor unrealistically overshoots the target and orbits around it, suggesting that this component was functionally critical during T6’s real-time neural control.

Changes in control policy as a function of gain. When making able-bodied movements, it is known that people naturally adapt to higher visuomotor gains by attenuating the magnitude of their motor commands [35–37]. We examined whether or not a similar adaptation occurred in our iBCI users by looking at how $f_{targ}$ (which quantifies how strongly the user pushed on the cursor) changed as a function of cursor gain. Figure 9 illustrates T6 and T8’s $f_{targ}$ functions during sessions where cursor gain was varied over a wide range of settings. For higher gains, the $f_{targ}$ functions were consistently smaller, indicating that users adapted their control policies to command smaller control vectors. In contrast, participants’ velocity damping (quantified by $f_{vel}$) was relatively invariant to cursor gain alone and only increased when both gain and smoothing were high (supplementary figure S8 summarizes the variations in $f_{targ}$ and $f_{vel}$ for all blocks).

Are feedback corrections necessary for accurate cursor movements?

The control policy models we considered assume that the user continuously corrects movement errors caused by unpredictable decoder noise. To confirm this assumption, we took away feedback from the piecewise control policy model (by making the feedback delay infinite) and measured the resulting...
increase in prediction error (section 7 in the supplement). For all three participants, the control policy model predicts the user’s control vector more accurately when feedback corrections are included in the model (i.e. when the model is not purely feedforward).

Are these feedback corrections necessary for accurate cursor movements, or could the user acquire the targets with only feedforward control? To answer this, we used the feedback control model to simulate movements without feedback corrections and measured the resulting decline in success rate. Movements were simulated without feedback by making the feedback delay infinite, forcing the model to rely entirely on feedforward commands based on internal state estimation. For almost all blocks, removing feedback caused a dramatic decline in accuracy that was exacerbated by settings with higher gain parameters (section 7 in the supplement). We confirmed the simulation result with additional data from participant T8. Matching the model’s prediction, T8 was far more accurate at moving towards targets and at holding still on top of targets with visual feedback (section 7 in the supplement). Without feedback corrections, errors due to decoding noise accumulate over the course of the movement, resulting in gross inaccuracies.

Discussion

We used feedback control theory to understand how three participants from the BrainGate2 pilot clinical trial modulated their population neural activity to control iBCI cursor movements (see figures 1 and 3 for an overview of our modeling approach). We called the user’s population neural activity in the two dimensions used by the decoder the ‘control’ vector. To investigate each user’s control policy, we used a piecewise model to describe how the control vector varied as a function of the user’s movement goal (target position) and the state of the cursor (its position and velocity). By varying cursor dynamics from block to block, we also investigated how users adapt their control policy to different cursor gains and smoothing dynamics (figure 2 shows the range of dynamics we imposed).

Departures from previous assumptions

We discovered two significant aspects of the participants’ control policies that were inconsistent with assumptions made in previous work. First, we showed that the magnitude of the user’s control vector gradually declines as the user approaches the target from afar, then decreases more sharply as the user comes into contact with the target (figure 4). When calibrating decoders or simulating iBCIs, previous studies have typically assumed a linear control policy \([1, 15, 17–20]\), which declines too steeply as a function of target distance and does not capture how the magnitude of the control vector saturates when the cursor is far from the target. Second, we found that the user makes constant feedback corrections even when the cursor is on top of the target, in contrast to the ReFIT assumption that assumes the user zeroes their control vector when on top of the target \([4, 8, 16]\). These near-target feedback corrections are important for the user to be able to hold the cursor accurately on top of the target (supplementary figures S7 and S9).

User adaptation to decoder dynamics

We showed for the first time that iBCI users can adjust their control policy to account for different cursor gain and smoothing settings (figures 8–9, supplementary figure S8). This adaptation occurred in two forms. First, when the cursor gain and smoothing settings were chosen to create significant second-order dynamics (such that the cursor takes longer to accelerate and decelerate), we observed a significant velocity damping component to participants T6 and T8’s control policies that was functionally important for preventing target overshoot (figure 8). This is the first direct evidence that iBCI users can adapt to actively account for second-order dynamics of the end effector. Second, we found that users adjusted their control policy to generate smaller motor commands when the cursor gain was increased, similar to how able-bodied people naturally adapt to increased visuomotor gains when making arm movements \([35–37]\) (figure 9, supplementary figure S8).
Models of neural activity during iBCI control

Our control policy results are a description of the user’s population level neural activity in the dimensions extracted by the decoder. The fact that the user’s neural activity can adapt flexibly to match the cursor dynamics provides more evidence that motor cortical activity does not have a rigid relationship to certain variables of movement (e.g. velocity) during iBCI control. Instead, motor cortical activity can be interpreted as the output of a feedback control system that adapts to control the specific end effector that is given to it.

In addition to offering this perspective, the piecewise control policy model can also be used quantitatively to better understand neural activity during iBCI control. For example, when investigating whether temporal dynamics or other such phenomena exist in the neural activity during closed-loop control, a null hypothesis could be linear tuning to the control policy model presented here. As another example, to analyze a neural feature’s preferred direction and depth of modulation during closed-loop control, a tuning model could be fit to the user’s control vector as determined by the control policy model.

Optimal feedback control theory

Optimal feedback control theory has been used to explain certain features of human feedback control policies by assuming that these policies optimize a cost function [38–41]. The cost function typically includes both a measure of task completion and energy expenditure. Recently, researchers have suggested that optimal feedback control theory may also be a good model for iBCI control [17, 19, 20]. These iBCI studies have all applied the most widely used model of optimal feedback control: a plant with linear dynamics, additive white Gaussian noise, and a cost function that is a quadratic function of the control vector and the system state. Under these assumptions, the solution is either a constant linear control policy (if the cost function is integrated over an infinite time horizon) or a time-varying linear control policy (if the cost function is integrated over a finite time horizon).

Interestingly, we found that neither a constant nor a time-varying linear feedback control law described the control policies taken by our participants as well or simulated movements as accurately as the non-linear, piecewise model (figure 5, supplementary figure S7). This was primarily because the control vectors generated by a linear model decline too quickly as the cursor approaches the target, in contrast to the non-linear “saturating” feedback control policies we observed in our participants (figure 4). Explaining iBCI control policies from an optimal control theory perspective may require moving beyond the linear-quadratic-Gaussian model that implies a linear control policy.

Decoder calibration

Recent improvements have been made in decoder calibration by making better assumptions about the user’s control policy [4, 8, 16, 17]. The piecewise control policy model proposed here might be able to improve the decoder calibration process further by providing a more accurate estimation of what movements the user was trying to make at each moment in time. Building decoders to map neural activity to the output of the piecewise model, which outperformed other models as an estimate of the user’s control vector, may lead to a performance benefit. Future work will test this hypothesis directly online.

Offline optimization and evaluation of decoder performance

The modeling framework introduced here may be useful for offline optimization and evaluation of linear velocity decoders with exponential smoothing dynamics. To find the optimal gain and smoothing parameters for a specific user without having to cycle through many candidate values online, the closed-loop feedback control model introduced here could be used to simulate movements under different parameters and select those that are likely to enable the best online performance.

To evaluate the performance of the linear decoding matrix (the D matrix in our notation) without having to use it online, researchers could fit a control policy model to the un-smoothed output of the decoder, effectively separating decoding error from user intent. The magnitude of the decoding error could then serve as a performance metric to optimize what algorithms are used to calibrate the decoder. The decoding error could also be used to fit a noise model that could be combined with the control policy model to enable closed-loop simulation of online performance, enabling researchers to predict the level of performance that would result from using that decoding matrix.

Generalizing to different decoder architectures

The control policy model developed in this study was used solely to examine cursor movements made with a linear decoder that had exponential velocity smoothing dynamics. Care should be taken before applying these results as-is to different decoding architectures. First, if the movement dynamics created by the decoder differ from those studied here (e.g. the decoder maps neural activity to position instead of velocity), the user may adapt their control policy to take a different form. This is to be expected, given our findings that the user adapts their control policy to match the decoder dynamics (figures 8 and 9). Second, if the motor cortex encodes aspects of the user’s feedback control vector in a non-linear way that cannot be captured by a linear decoder, then a non-linear decoder could reveal additional aspects of the user’s control policy not shown here. We deliberately restricted our investigation to the two dimensions of neural activity that were linearly extracted by the decoder, since those are the only aspects of the neural activity that influenced the motion of the cursor. Even so, there may have been aspects of the control vector being represented in the background that were not explored here.

Conclusion

In this study, we asked the question: what feedback control laws govern how an iBCI user’s neural activity modulates to...
drive the cursor towards the target? To answer this question, we proposed a new feedback control model of iBCI cursor movements and tested it extensively against real data. We then used the model to demonstrate that the user can flexibly modulate their neural activity to adapt their control policy to different cursor dynamics. We hope that these results will inform the design of future iBCI systems by providing a more complete picture of how users control iBCIs and how users adapt to end effector dynamics. Also, given that decoding performance has recently been improved by using better assumptions about the user's control policy [4, 8, 16, 17], we hope that the piecewise control policy model will help to improve decoding performance even further.

Acknowledgments

The authors would like to thank participants T6, T7, T8 and their caregivers and families; Brittany Sorice for help with data collection; Daniel Young for informative discussions. This study is funded by the Larry and Pamela Garlick Foundation; National Science Foundation: GRFP DGE-0951783; National Institutes of Health: 1R01HD077220, R01NS066311, R01DC014034, R01DC009899, N01HD53403, N01HD10018; Rehabilitation Research and Development Service: B6453R; Samuel and Betsy Reeves Foundation; MGH-Deane Institute for Integrated Research on Atrial Fibrillation and Stroke.

References

[38] Todorov E and Jordan M I 2002 Optimal feedback control as a theory of motor coordination Nat. Neurosci. 5 1226–35
[40] Liu D and Todorov E 2007 Evidence for the flexible sensorimotor strategies predicted by optimal feedback control J. Neurosci. 27 9354–68