The brain uses invariant dynamics to generalize outputs across movements

Vivek R. Athalye[1]†, Preeya Khanna[2]†, Suraj Gowda[4], Amy L. Orsborn[3], Rui M. Costa[1]*‡, and Jose M. Carmena[4,5,6]*‡

[1] Zuckerman Mind Brain Behavior Institute, Departments of Neuroscience and Neurology, Columbia University; New York, NY, USA

[2] Department of Neurology, University of California, San Francisco; San Francisco, CA, USA

[3] Departments of Bioengineering, Electrical and Computer Engineering, University of Washington, Seattle; Seattle, WA, USA

[4] Department of Electrical Engineering and Computer Sciences, University of California, Berkeley; Berkeley, CA, USA

[5] Helen Wills Neuroscience Institute, University of California, Berkeley; Berkeley, CA, USA

[6] UC Berkeley-UCSF Joint Graduate Program in Bioengineering, University of California, Berkeley; Berkeley, CA, USA

*Corresponding author. Email: rc3031@columbia.edu (RMC); jcarmena@berkeley.edu (JMC)

†These authors contributed equally to this work.

‡Senior author

Abstract:
The nervous system uses a repertoire of outputs to produce diverse movements. Thus, the brain must solve how to issue and transition the same outputs in different movements. A recent proposal states that network connectivity constrains the transitions of neural activity to follow invariant rules across different movements, which we term 'invariant dynamics'. However, it is unknown whether invariant dynamics are actually used to drive and generalize outputs across movements, and what advantage they provide for controlling movement. Using a brain-machine interface that transformed motor cortex activity into outputs for a neuroprosthetic cursor, we discovered that the same output is issued by different activity patterns in different movements. These distinct patterns then transition according to a model of invariant dynamics, leading to patterns that drive distinct future outputs. Optimal control theory revealed this use of invariant dynamics reduces the feedback input needed to control movement. Our results demonstrate that the brain uses invariant dynamics to generalize outputs across movements.

Introduction

On a moment-by-moment basis, our nervous system issues outputs that update movement. Diverse movements are composed from the same repertoire of outputs, just as diverse songs are composed from the same repertoire of musical notes. Thus, a fundamental problem the brain must solve is how to generalize an output, i.e. how to issue the same output in different movements and transition to distinct future outputs.

The instantaneous pattern of neural population activity is thought to underlie both issuing an output and transitioning to the subsequent output. In motor cortex, the activity pattern issues output to update movement through direct connections to the spinal cord[1] and through downstream brain areas such as the basal ganglia, midbrain, and brainstem[2]. Additionally, the present activity pattern is thought to influence future patterns through network connectivity[3]. A recent proposal is that network connectivity constrains activity, such that transitions between patterns follow rules[4–8] that are invariant across different movements[9,10]. We term these rules for how each activity pattern transitions as "invariant dynamics." Models of invariant dynamics explain features of activity unexplained by measured behavior[3,9,11,12], predict activity transitions during different movements[9,10], on single trials[10,13–15], for single neurons' spiking[13], for local field potential features[16], and over many days[15,17], and help predict ongoing behavior from neural activity[10,15,16,18].

While past work characterizes invariant dynamics and its statistical relationship to behavior, it remains unknown if the brain actually uses invariant dynamics to drive and generalize outputs, or if invariant dynamics accompany and predict behavior without directly driving it. Further, it is not well-understood what advantage using invariant dynamics may confer to the brain for controlling movement, especially in behaviors that require integrating ongoing feedback.

Understanding what neural activity patterns and transitions the brain uses to generalize outputs across movements faces a central challenge: precisely identifying the output that updates movement and the causal transformation from neural activity to its output. Neural activity is transformed through downstream synaptic projections into an output, and it is still debated what motor parameters are updated by the output of different neural populations[7,19,20]. Thus, it is difficult to determine when the brain issues the same exact output in different movements. Further, because the transformation from neural activity to its output is unknown, it has been impossible to determine whether transitions of activity due to invariant dynamics drive the output for movement. To address the challenge, we leveraged a brain-machine interface (BMI) [21–24] in which the transformation from neural activity to output for movement is known exactly and determined by the experimenter.

We trained rhesus monkeys to use the activity of 20-151 units in motor cortex (dorsal premotor and primary motor cortex) to move a computer cursor on a screen through a BMI. The BMI approach enabled us to know when a particular output is issued, and to study the distinct transitions following the particular output in different movements. If the current neural population activity pattern not only issues the current output but also influences its transition to future activity and output through invariant dynamics, there are two main predictions for the patterns that the brain uses to control movements. First, in order to issue the same output across different movements, different activity patterns should be used since different movements require distinct transitions. Further, these activity patterns for the same output should be systematically different across movements, predicted by a model of invariant dynamics and past activity patterns. Second, in order for the same output to transition to distinct subsequent outputs, the distinct patterns for the same output should transition through invariant dynamics to the

patterns that causally issue the distinct subsequent outputs. The BMI provides a critical test of this causal contribution of invariant dynamics to output, since invariant dynamics could transition activity in a manner that avoids modifying the components of neural activity that drive output[25–27]. Our evidence shows that invariant dynamics do indeed issue and transition outputs for movement.

Finally, it is unclear what advantage invariant dynamics provide for controlling movement. Past work has presented the view that invariant dynamics could define the activity transitions needed to propagate an initial state of activity [26,28] into a complete activity trajectory which produces movement. However, it is unclear how this view would extend to behaviors that require ongoing input[29] and feedback[27] to execute. We introduce a model based on optimal control theory[30,31] for how a neural population could combine ongoing feedback input with its invariant dynamics in order to control movement in a closed-loop manner. Our model reveals that invariant dynamics reduce the feedback input that a neural population needs to issue the appropriate outputs for movement. Further, the model confirms the principle that using invariant dynamics to control movement leverages systematically different activity patterns to issue the same output across movements.

Results

Task and behavior

We used a BMI to study the dynamics of neural population activity as it causally issues outputs for movement. The BMI transformed high-dimensional neural activity into two-dimensional outputs that updated the computer cursor's velocity[32,33]. Neural activity was recorded using chronically implanted microwire electrode arrays spanning bilateral dorsal premotor cortex and primary motor cortex. Each unit's recorded spike counts at time t were

stacked into a vector of population activity x_t , and the BMI used a "decoder" given by matrix K to linearly transform population activity into a two-dimensional output:

$$\text{output}_t = Kx_t \quad (1)$$

The output linearly updated the velocity of a two-dimensional computer cursor (Fig. 1a and Methods). The space of two-dimensional outputs constituted the output repertoire.

To understand how the brain issues outputs from its repertoire to produce diverse movements, we trained monkeys to perform two different tasks (Fig. 1B). Monkeys performed a center-out task in which the goal was to move the cursor from the center of the workspace to one of eight radial targets and an obstacle-avoidance task with the same goal plus the constraint of avoiding an obstacle blocking the straight path to the target.

Our experimental design allowed us to test how the brain issues outputs for the BMI. Critically, subjects controlled the BMI without using trained overt movement. The BMI was not designed to decode activity during trained overt movement[10] and thus did not demand associated activity. The parameters defining how the BMI transformed neural activity into cursor movement were initialized from a baseline recording block in which subjects passively watched a cursor move through the tasks. Then in a calibration block, the BMI parameters were continuously refined based on closed-loop BMI performance [34]. Following calibration, BMI parameters were fixed. All data subsequently analyzed was from fixed-parameter blocks.

Predictions for how neural activity generalizes outputs for movement

We elucidate what neural population activity must accomplish to generalize outputs for movement. First, in order to issue a particular output, the neural population can use one of many activity patterns[25,26]. This is because there are more neurons than output dimensions, as illustrated with two neurons and a one-dimensional output (Fig. 1C). In high-dimensional

neural activity space where each neuron's activity is one axis, the decoder defined dimensions that causally drive output, which we term the "decoder space," and orthogonal dimensions which have no consequence on the output through the decoder, which we term the "decoder null space." Concretely, neural activity's component in the decoder space determines the output issued. Then, in order to generalize a particular output across movements, the neural population must be able to transition from activity issuing the particular output to activity issuing distinct future outputs (Fig. 1D).

We consider that the transitions of neural population activity x_t are driven by invariant dynamics h and input from the brain[4] (Fig. 1E):

$$x_{t+1} = h(x_t) + \text{input}_t \quad (2)$$

In this "invariant dynamics view", the invariant dynamics h describe how the present activity pattern is influenced by its past activity and influences future activity. Activity transitions due to invariant dynamics can be visualized with a flow field in neural activity space that is invariant regardless of the current output or ongoing movements.

We ask if invariant dynamics are used to generalize output by testing two critical properties of how neural activity issues and transitions outputs. First, a particular output should be issued in different movements by systematically different activity patterns. A model of invariant dynamics should predict these different patterns from past activity. For example, in Fig. 1E the same output is issued by distinct patterns for the blue movement and green movement (blue dot and green dot, respectively), and invariant dynamics predict the transition to these distinct patterns from the previous timepoint's pattern. Second, a model of invariant dynamics should predict the transition to the future patterns that causally issue the subsequent output. In particular, invariant dynamics should predict how the same output undergoes different

transitions in different movements, as illustrated in Fig. 1E. This requires invariant dynamics to predict transitions in the decoder space based on neural activity's coordinate in the decoder null space, which has been proposed from studies on preparation of movement[26,28].

These two properties could arise to some small extent even without invariant dynamics, such as in the simple view that the brain re-uses the same activity patterns to issue a particular output in different movements (Fig. 1F):

$$x_t = f(\text{output}_t) + \text{noise}_t \quad (3)$$

Here, the re-used patterns described by the encoding function f and noise are invariant across movements. The invariant pattern for the present output flexibly transitions to different subsequent outputs' invariant patterns, and thus patterns do not transition by invariant dynamics. Under this invariant pattern view, different activity patterns could be used to issue an output simply due to noise, and transitions between activity patterns could be predictable simply due to the predictability of the behavior they produce. Using the invariant pattern view as a control, we test whether invariant dynamics are used to generalize output for movement.

The same output is issued by different activity patterns in different movements

Given that movements were produced from a repertoire of outputs for cursor velocity, we studied the same output issued in different movements. For analysis, we discretized the two-dimensional, continuous outputs into 32 bins (8 radial x 4 magnitude). Then in each movement, we studied outputs that fell within the same output bin, hereafter referred to as an output. For each trial, the "condition" of movement was defined as the target and task performed and whether the cursor went clockwise or counterclockwise around the obstacle during the obstacle-avoidance task (single trials to 9 example conditions are shown in Fig. 2A). On each session, each output was deemed to be used in a condition if it occurred 15 or more times (Extended Data

Fig. 1, Methods). Each individual output was used with regularity during multiple conditions (Extended Data Fig. 1B). Outputs in different conditions were used within distinct trajectories of outputs and cursor positions (Fig. 2A). We denote the pairing of an output with a condition in which it was used as an "output-condition."

We sought to test whether the same output was issued by different activity patterns during different conditions. We calculated the distance between the average activity for each individual output-condition ("condition-specific output activity") and the average activity for the output pooling over conditions ("condition-pooled output activity"). We then compared this distance to the distances expected due to noise if the brain reuses the same invariant patterns to issue an output (Fig. 1F). To emulate how neural activity for an output would be re-used by sampling an invariant pattern distribution, we constructed shuffled datasets that shuffled each observation of activity that issued a specific output to another observation that issued the same output in a random trial and condition (Fig. 2B, Methods). This method destroyed any systematic differences in condition-specific output activity, leaving differences only due to noise. We tested if the experimentally-observed distances between condition-specific output activity and condition-pooled output activity significantly exceeded the distribution of distances calculated in the shuffled data sets ("shuffle distribution").

Overall, neural activity for a particular output significantly deviated across conditions relative to the shuffle distribution representing the re-use of patterns for an output (Fig. 2C-F). This is illustrated with an example output for many conditions for an example neuron (Fig. 2C) and over the entire population (Fig. 2D). Distances were significant for a large fraction of individual output-conditions (Fig. 2E, Extended Data Fig. 2), outputs pooling over conditions (Fig. 2E), and individual neurons pooling over individually significant output-conditions (Fig.

2F, Extended Data Fig. 2). The average magnitude of the significant deviations between condition-specific output activity and condition-pooled output activity was 20-30% of the magnitude of the condition-pooled output activity pattern (Fig. 2G). Further, these deviations reflected structure in the behavior, as output activity was more distinct for conditions with more distinct past and future outputs (Extended Data Fig. 6E-G). This evidence demonstrates that significantly different activity patterns are used to issue the same output in different conditions.

The different activity patterns used to issue the same output are predicted by a model of invariant dynamics

We next asked whether the distinct activity patterns issuing the same output were predicted by a model of invariant dynamics. We used neural activity across conditions to estimate the dynamics of activity transitions with a linear model (Fig. 3A, top right):

$$x_{t+1} = Ax_t + b \quad (4)$$

We found that dynamics A were low-dimensional (2-4 dimensions) and decaying to a fixed point b (Extended Data Fig. 3), contrasting with rotational dynamics observed during natural motor control [9–11,18,35]. We combined the dynamics model with knowledge of the BMI decoder (Equation 1) to predict the population's activity pattern given the output it issued and its previous activity (Fig. 3A). Further, we tested whether the dynamics model was invariant across outputs and conditions, as expected if dynamics reflect underlying network connectivity. To test invariance, dynamics models were fit on neural activity specifically excluding individual outputs or conditions and then used to predict the activity for left-out outputs or conditions (Fig. 3B, Methods). We evaluated the significance of the dynamics model predictions in comparison to a dynamics model fit on shuffled datasets that preserved behavior and represented the re-use of invariant patterns for an output (Fig. 2B). This shuffled data destroyed the temporal ordering of

neural activity as much as possible while preserving the temporal order of the issued outputs. Thus, the shuffle dynamics model captured the expected predictability in neural activity due to performance of the tasks' movements. As an additional reference, we also computed the prediction of neural activity when only given the output it issued.

We found that the dynamics model significantly predicted neural activity patterns given the outputs they issued and the previous pattern (Fig. 3C). Further, the dynamics model was invariant, predicting activity for outputs and conditions left-out from model fitting (Fig. 3C). The R^2 of model predictions for activity from left-out outputs and conditions on single time points in individual trials was significantly better than predictions of shuffle dynamics and very close to the performance of the model trained on all outputs and conditions (Fig. 3C). Further, the models were invariant even when much larger subsets of outputs and conditions were left-out (Extended Data Fig. 4). We confirmed that the result was not driven by neural activity simply representing behavioral variables in addition to the output that updated velocity, as dynamics models predicted activity beyond models where activity encoded cursor kinematics, target location, and condition (Extended Data Fig. 5). This is consistent with previous work showing that cursor kinematics such as cursor position only weakly affect activity controlling a BMI in the absence of arm movements[36].

The dynamics model also significantly predicted the different condition-specific output activity patterns, as shown for an example output across conditions for the example neuron (Fig. 3D) and for the entire population (Fig. 3E). Activity was significantly predicted relative to shuffle dynamics for almost all output-conditions (Fig. 3F left), outputs pooled over conditions (Fig. 3F middle), and neurons pooled over all outputs and conditions (Fig. 3F right). Across subjects and sessions, the dynamics model predicted 20-40% of the condition-specific

component of activity for a given output (i.e. the difference between condition-specific output activity and condition-pooled output activity) (Fig. 3G, Methods). Finally, the dynamics model predictions preserved structure of activity for a given output between pairs of conditions (Extended Data Fig. S6A-D) and explained the finding that activity patterns for a given output are more distinct between pairs of conditions with more distinct past and future outputs (Extended Data Fig. S6E-I). Altogether, these results show that when the brain generalizes an output, invariant dynamics contribute to what activity pattern is used to issue the output.

The transitions between patterns that drive output are predicted by a model of invariant dynamics

We next asked whether invariant dynamics are actually used to transition activity towards the next output. We were able to determine the consequence of invariant dynamics on the next output by using the dynamics model to predict next activity from current activity and then applying the predicted next activity through the decoder to predict the next output (Fig. 4A). This analysis was possible because the BMI provides a fully-defined causal transformation from activity to output which is not easily measurable during natural motor control. We highlight that invariant dynamics could potentially not contribute to the next output; if invariant dynamics only transitioned activity in the decoder null space (the dimensions of activity that the decoder does not use to transform activity into an output, Fig. 1C), invariant dynamics would not update the issued output (Fig. 4B). To assess this possibility, we compared a dynamics model restricted to the decoder null space ("decoder-null dynamics", see Methods) to the full neural dynamics model.

The decoder-null dynamics model significantly predicted the next activity pattern relative to shuffle dynamics (Fig. 4C), showing that activity transitions followed invariant dynamics in

decoder-null dimensions. As expected, the decoder-null dynamics model provided no prediction for the next output (Fig. 4D), exemplifying that successful prediction of next neural activity does not imply prediction of next output. Critically, the full dynamics model exceeded the decoder-null dynamics model by predicting both the next activity pattern and the next output (Fig. 4CD). Activity and output predictions were significant relative to shuffle dynamics that captured expected predictability due to performance of movements. Importantly, the model was invariant, predicting across outputs and conditions left-out from model fitting (Fig. 4CD).

Previously, we had found that a particular output was issued by distinct activity patterns in different conditions. We thus analyzed whether the dynamics model transitioned these distinct activity patterns towards distinct next outputs appropriate for each condition. The correspondence between the dynamics model's prediction for the next output and the true next output is shown for an example output (Fig. 4E). Indeed, the dynamics model significantly predicted the next output for a given output for the majority of output-conditions and outputs pooling over conditions (Fig. 4F). The predictions preserved the structure of next outputs between pairs of output-conditions (Extended Data Fig. 6J-L). Notably, the dynamics model accurately predicted the direction, i.e. clockwise versus counterclockwise, in which the next output would rotate relative to the current output for almost all output-conditions (Fig. 4G). These results show that invariant dynamics contribute to issuing and transitioning outputs.

Optimal control theory reveals the advantage of using invariant dynamics to control movement based on ongoing feedback

We observe that our dynamics model did not perfectly determine what output was issued. Thus, it is unclear what advantage invariant dynamics provide to the brain for driving and generalizing outputs across movements. Given recent work that motor cortex relies on ongoing

input to control movement[29], we built a model for how a neural population could combine ongoing input with its invariant dynamics in order to control movement in a closed-loop manner. In our model based on optimal feedback control theory, population activity issued outputs to control the BMI cursor to produce center-out and obstacle avoidance movements. Activity followed linear dynamics, received noise, and was driven by input optimized to achieve the task goals (Fig. 5A). Concretely, an optimal linear feedback controller that was specific to each condition computed time-varying inputs to the neural population based on observed population activity and cursor state. Computations used knowledge of the neural population's invariant dynamics and the decoder.

To identify how invariant dynamics contributed to movement, we simulated the brain performing optimal feedback control of the BMI using neural dynamics models fit from experimental data (Invariant Dynamics Model). The Invariant Dynamics Model was compared to simulations that used dynamics models with the transition matrix set to zero (No Dynamics Model). To facilitate comparison, we designed the models to produce matched behavior, resulting in center-out and obstacle-avoidance cursor trajectories with equal success and target acquisition time (Fig. 5B, Methods). We found that the Invariant Dynamics Model required significantly less input than the No Dynamics Model (Fig. 5C). This result illustrates the computational benefit that less input is needed to control neural populations with the invariant dynamics we observed experimentally.

Finally, we confirmed the principle that using invariant dynamics to control movement from feedback makes use of distinct activity patterns to issue a particular output during different movements. For an example output used during different conditions (Fig. 5D), condition-specific output activity was significantly different from the condition-pooled output activity in

the Invariant Dynamics Model but not the No Dynamics Model (Fig. 5E). Analyzing activity over all outputs and conditions, we found the Invariant Dynamics Model deviated from the shuffle distribution representing the invariant pattern view significantly more than the No Dynamics Model (Fig. 5F).

Discussion

Our brain uses a repertoire of outputs to produce diverse movements. While it is thought that neural activity underlying movement is constrained by network connectivity to follow invariant dynamics, it has been untested whether invariant dynamics are actually used to drive and generalize outputs across movements. Using a BMI to define the transformation from activity to output for movement, we discovered that different neural activity patterns are used to issue the same output in different movements. The patterns for the same output vary systemically depending on past activity, and critically, they transition according to invariant dynamics towards patterns that causally drive the subsequent output. Our results provide a conceptual advance beyond previous work that characterized invariant dynamics during behavior[9–11,13] by showing that invariant dynamics are actually used to issue the same output in different movements and transition to the subsequent output.

Further, it has been unclear what advantage invariant dynamics provide for controlling movement based on input[29] and feedback[27]. In our study, optimal control theory reveals that invariant dynamics can help the brain perform feedback control of movement, reducing the input that a neural population needs to issue the appropriate outputs for movement. We verified that that the use of invariant dynamics for feedback control results in issuing the same output across different movements with different activity patterns, as we observed in our experimental data.

These results refute that the brain reuses the same population activity patterns to issue the same output in different movements. This perspective is present in classic motor control studies

that describe populations of motor cortex neurons as being dedicated to representing movement parameters that their activity updates[37,38]. It is still debated what movement parameters are updated by motor cortex populations[7,19,20], as population activity represents diverse movement parameters spanning from low-level muscle-related parameters such as muscle activation [39–41] and synergies [42,43] and force/torque [37,44–46] to high-level movement parameters such as position [47–49], distance [50], velocity [48,49], speed [51], acceleration [52], and direction of movement [46,51,53,54]. Our findings using a BMI strongly suggest that regardless of how motor cortex output updates physical movement, the same neural activity patterns are not re-used to issue the same output across different conditions of movement. Instead, the use of systematically different patterns that transition according to invariant dynamics critically supports the recent proposal that neural activity in motor cortex avoids "tangling": having similar activity patterns undergo dissimilar transitions to control movement[3].

Our results that a neural population's invariant dynamics do not perfectly determine its next issued output contrasts with the view that neural population dynamics evolve an initial state of activity [26,28] into a complete activity trajectory which produces movement [9,55]. Instead, we propose a model based on optimal control theory [30,31,56,57] in which the neural population combines ongoing input [29] with invariant dynamics to control movement. In this view, the invariant dynamics do not need to define the precise activity transitions that perfectly produce movement; they only need to provide useful transitions that inputs can harness to control movement. Our results show that invariant dynamics provide the advantage of reducing the input a neural population needs to issue outputs for movement, adding to previous work identifying that invariant dynamics provide robustness to noise[3].

This feedback control perspective expands the number of behaviors for which invariant dynamics are useful, since invariant dynamics can provide useful activity transitions for driving output across diverse behaviors without specifying the precise transitions needed to completely produce each movement. In our data, simple dynamics (decaying dynamics with different time constants) in a low-dimensional activity space (~4 dimensions) were used to control many conditions of movement (~20 conditions). While our results did not rely on high-dimensional dynamics, they still rely on more dimensions of neural activity than output dimensions. In particular, our results refute a simplistic interpretation of the minimal intervention principle [56] in which neural populations should only control the few dimensions of activity which matter directly for issuing outputs, which are given by the decoder space in these experiments. Instead, we find that invariant dynamics provide constraints in the dimensions of activity which do not directly matter for issuing current outputs, given by the decoder null space (Fig. 4C) [25], so that inputs in these dimensions help produce future outputs (Fig. 5C). This accords with the finding that motor cortex responses to feedback are initially in the decoder null space before transitioning to activity that issues corrective outputs[27]. Broadly, our results provide a feedback control perspective for how invariant dynamics within manifolds of activity enable the brain to generalize outputs across diverse behaviors [4,58,59].

There is almost surely a limitation to the behaviors that particular neural dynamics are useful for. Motor cortex population activity occupies orthogonal dimensions and shows a markedly different influence on muscle activation during walking and trained forelimb movement[60], and follows different dynamics for reach and grasp movements[61]. Notably, our finding of decaying dynamics for BMI control contrasts with rotational dynamics observed during natural arm movement[9–11,18]. We speculate this could be because controlling the BMI

relied more on feedback control than a well-trained natural arm movement and/or because controlling the BMI did not require the temporal structure of outputs needed to control muscles for movement. One intuition would be that behaviors which need particular temporal frequencies of outputs elicit neural dynamics that produce those frequencies in their activity transitions[62,63]. Recent theoretical work shows that cortico-basal ganglia-thalamic loops can switch between different cortical dynamics useful for different temporal patterns of output[64].

The use of invariant dynamics to generalize outputs has implications for how the brain learns new behavior [31,65], enabling the brain to leverage pre-existing dynamics for initial learning [66–68] and to develop new dynamics through gradual reinforcement [69,70]. This learning that modifies dynamics relies on plasticity in cortico-basal ganglia circuits[71–73] and permits the brain to reliably access a particular pattern for a given output and movement [24], even if the same pattern is not used to issue the same output across movements.

Our results suggest that modeling invariant dynamics informs the design of new neuroprosthetics that can generalize outputs to new behaviors [10] and classify entire movement trajectories [74]. We expect that as new behaviors are performed, distinct activity patterns will be used to issue the same output, but that invariant dynamics can predict and thus recognize these distinct patterns as signal for the BMI rather than noise. In addition, our results inform the design of rehabilitative therapies to restore dynamics following brain injury or stroke to recover movement [75,76].

Overall, this study put the output of a neural population into focus, revealing how rules for activity transitions are used to generalize outputs and produce different movements. This was achieved by eliciting the brain to skillfully control the neural population activity we recorded. BMI[31,77–80], especially combined with technical advances in measuring, modeling, and

manipulating activity from defined populations, provides a powerful technique to test emerging hypotheses about how neural circuits generate activity to control behavior.

Online Methods

Surgery, electrophysiology, and experimental setup

Two male rhesus macaques (Macaca mulatta, RRID: NCBITaxon:9544) were bilaterally, chronically implanted with 16 x 8 arrays of Teflon-coated tungsten microwire electrodes (35 mm in diameter, 500 mm separation between microwires, 6.5 mm length, Innovative Neurophysiology, Durham, NC) in the upper arm area of primary motor cortex (M1) and posterior dorsal premotor cortex (PMd). Localization of target areas was performed using stereotactic coordinates from a neuroanatomical atlas of the rhesus brain [81]. Implant depth was chosen to target layer 5 pyramidal tract neurons and was typically 2.5 - 3 mm, guided by stereotactic coordinates.

During behavioral sessions, neural activity was recorded, filtered, and thresholded using the 128-channel Multichannel Acquisition Processor (Plexon, Inc., Dallas, TX) (Monkey J) or the 256-channel Omniplex D Neural Acquisition System (Plexon, Inc.) (Monkey G). Channel thresholds were manually set at the beginning of each session based on 1–2 min of neural activity recorded as the animal sat quietly (i.e. not performing a behavioral task). Single-unit and multi-unit activity were sorted online after setting channel thresholds. Decoder units were manually selected based on a combination of waveform amplitude, variance, and stability over time.

Prior to this study, Monkeys G and J were trained at arm reaching tasks and spike-based 2D neuroprosthetic cursor tasks for 1.5 years. All procedures were conducted in compliance with the NIH Guide for the Care and Use of Laboratory Animals and were approved by the University of California, Berkeley Institutional Animal Care and Use Committee.

Neuroprosthetic decoding

Subjects' neural activity controlled a 2D neuroprosthetic cursor in real-time to perform centerout and obstacle avoidance tasks. The neuroprosthetic decoder consists of two models: 1) A cursor dynamics model capturing the physics of the cursor's position and velocity. 2) A neural observation model capturing the statistical relationship between neural activity and the cursor.

The neuroprosthetic decoder combines the models optimally to estimate the subjects' intent for the cursor and to correspondingly update the cursor.

Decoder algorithm and calibration -- Monkey G

Monkey G used a velocity Kalman filter (KF) [82,83] that uses the following models for cursor state c_t and observed neural activity x_t :

$$c_t = Ac_{t-1} + w_t, w_t \sim N(0, W)$$

$$x_t = Cc_t + q_t, q_t \sim N(0, Q)$$

In the cursor dynamics model, the cursor state $c_t \in R^5$ was a 5-by-1 vector $[pos_x, pox_y vel_x, vel_y, 1]^T$, $A \in R^{5x5}$ captures the physics of cursor position and velocity, and w_t is additive Gaussian noise with covariance $W \in R^{5x5}$ capturing cursor state variance that is not explained by A.

In the neural observation model, neural observation $x_t \in R^N$ was a vector corresponding to spike counts from N units binned at 10 Hz, or 100ms bins. C models a linear relationship between the subjects' neural activity and intended cursor state. The decoder only modeled the statistical relationship between neural activity and intended cursor velocity, so only the columns

corresponding to cursor state velocity and the offset (columns 3-5) in C were non-zero. Q is additive Gaussian noise capturing variation in neural activity that is not explained by Cc_t. For Monkey G, 35-151 units were used in the decoder (median 48 units).

In summary, the KF is parameterized by matrices $\{A \in R^{5x5}, W \in R^{5x5}, C \in R^{Nx5}, Q \in R^{NxN}\}$. The KF equations used to update the cursor based on observations of neural activity are defined as in [83].

The KF parameters were defined as follows. For the cursor dynamics model, the A and W matrices were fixed as in previous studies [84]. Specifically, they were:

$$A = \begin{bmatrix} 1 & 0 & 0.1 & 0 & 0 \\ 0 & 1 & 0 & 0.1 & 0 \\ 0 & 0 & 0.8 & 0 & 0 \\ 0 & 0 & 0 & 0.8 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad W = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

where units of cursor position were in cm and cursor velocity in cm/sec.

For the neural observation model, the C and Q matrices were initialized from neural and cursor kinematic data collected at the beginning of each experimental session while Monkey G observed 2D cursor movements that moved through either a center-out task or obstacle avoidance task. Maximum likelihood methods were used to fit C and Q.

Next, Monkey G performed an "adaptation block" where he performed the center out or obstacle avoidance as the newly initialized decoder parameters were continuously adapted online (closed-loop decoder adaptation", or CLDA). This adaptation block was performed in order to arrive at parameters that would enable excellent neuroprosthetic performance. Every 100ms, decoder matrices C and Q were adapted using the recursive maximum likelihood CLDA algorithm [34]. Half-life values, defining how quickly C and Q could adapt, were typically 300 sec, and adaptation blocks were performed with a weak, linearly decreasing "assist" (re-defining c_t as a

weighted linear combination of user-generated c_t and optimal c_t to drive the cursor to the target). Typical assist values at the start of the block were 90% user-generated, 10% optimal and decayed to 100% user-generated, 0% optimal over the course of the block. Following CLDA, decoder parameters were fixed, and Monkey G completed the center out and obstacle avoidance tasks.

Decoder algorithm -- Monkey J

Monkey J used a velocity Point Process Filter (PPF) [32,33]. The PPF uses the same cursor dynamics model for cursor state c_t as the KF above, but uses a different neural observations model for the spiking $S_t^{1:N}$ of each of N neurons:

$$c_t = Ac_{t-1} + w_t, w_t \sim N(0, W)$$

$$p(S_t^{1:N}|v_t) = \prod_{j=1}^{N} (\lambda_j(t\,|v_t, \phi^j)\Delta)^{S_t^j} \exp(-\lambda_j(t|v_t, \phi^j)\,\Delta)$$

In the neural observations model, neural observation S_t^j is the j[th] neuron's spiking activity, equal to 1 or 0 depending on whether the j[th] neuron spikes in the interval $(t, t + \Delta)$. We used Δt = 5ms bins since consecutive spikes rarely occurred within 5ms of each other. For Monkey J, 20 or 21 units were used in the decoder (median 20 units). The probability distribution over spiking $p(S_t^{1:N}|v_t)$ was a point process with $\lambda_j(t\,|v_t, \phi^j)$ as the j[th] neuron's instantaneous firing rate at time t. $\lambda_j(t\,|v_t, \phi^j)$ depended on the intended cursor velocity $v_t \in R^2$ in the two dimensional workspace and the parameters ϕ^j for how neuron j encodes velocity. $\lambda_j(t\,|v_t, \phi^j)$ was modeled as a log-linear function of velocity:

$$\lambda_j(t\,|v_t, \phi^j) = \exp(\beta_j + \alpha_j^T v_t)$$

where ϕ^j parameters consist of $\alpha_j \in R^2, \beta_j \in R^1$.

In summary, the PPF is parameterized by $\{A \in R^{5x5}, W \in R^{5x5}, \phi^{1:N}\}$. The PPF equations used to update the cursor based on observations of neural activity are defined as in [33].

The PPF parameters were defined as follows. For the cursor dynamics model, the A and W matrices are defined as:

$$A = \begin{bmatrix} 1 & 0 & 0.005 & 0 & 0 \\ 0 & 1 & 0 & 0.005 & 0 \\ 0 & 0 & 0.989 & 0 & 0 \\ 0 & 0 & 0 & 0.989 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad W = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3.7 \times 10^{-5} & 0 & 0 \\ 0 & 0 & 0 & 3.7 \times 10^{-5} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

where units of cursor position were in m and cursor velocity in m/sec.

For the neural observations model, parameters $\phi^{1:N}$ were initialized from neural and cursor kinematic data collected at the beginning of each experimental session while Monkey J observed 2D cursor movements that moved through a center-out task. Decoder parameters were adapted using CLDA and optimal feedback control intention estimation as outlined in [32]. Following CLDA, decoder parameters were fixed, and subjects completed the center out and obstacle avoidance tasks.

Output definition and analysis pre-processing

We define the "output" as the direct influence of subjects' neural activity x_t (binned at 100ms) on the cursor. Concretely, in both decoders, the output was a linear transformation of neural activity that we write as Kx_t which directly updated the cursor velocity.

Output definition -- Monkey G

For Monkey G, the update to the cursor state due to cursor dynamics and neural observation x_t can be written as:

$$c_t = F_t c_{t-1} + K_t x_t$$

where $F_t c_{t-1}$ is the update in cursor state due to the cursor dynamics process and $K_t x_t$ is what we have defined as the neural output: the update in cursor state due to the current neural observation. $K_t \in R^{5xn}$ is the Kalman Gain matrix and $F_t = (I - K_t C)A$. In practice K_t converges to its steady-state form K within a matter of seconds [85], and thus F_t converges to $F = (I - KC)A$, so we can write the above expression in its steady state form:

$$c_t = Fc_{t-1} + Kx_t$$

In this equation, the structure of K is such that neural activity x_t directly updates cursor velocity, and velocity integrates to update position. The following technical note explains the structure of K. Due to the form of the A, W matrices, $Rank(K) = 2$. Specifically, following decoder adaptation that imposed the constraint that the intermediate matrix $C^T Q^{-1} C$ was of the form aI, where $a = mean(diag(C^T Q^{-1} C))$, the rows of K that update the position of the cursor are equal to the rows of K that update the velocity multiplied by the update timestep: $K(1{:}2, :) = K(3{:}4, :) * dt$ [86] (see independent velocity control). Given this structure of K, neural activity's contribution to cursor position is the simple integration of neural activity's contribution to velocity over one timestep.

In summary, since Kx_t reflects the direct effect of the motor cortical units on the velocity of the cursor, we term the velocity components of Kx_t the "output". We analyzed the neural spike counts binned at 100ms that were used online to drive cursor movements with no additional pre-processing.

Output definition -- Monkey J

For Monkey J the cursor state updates in time as:

$$c_t = f_t(c_{t-1}) + K_t x_t$$

where

$$f_t(c_{t-1}) = (Ac_{t-1} - K_t e^{CAc_{t-1}}\Delta), \quad K_t = P_t C$$

Here $f_t(c_{t-1})$ is the cursor dynamics process and $K_t x_t$ is the neural output. $P_t \in R^{5x5}$ is the estimate of cursor state covariance, and $C \in R^{5xN}$ captures how neural activity encodes velocity as a matrix where each column is composed of $[0, 0, \alpha_j^{xvel}, \alpha_j^{yvel}, \beta_j]^T$ for the jth unit.

We define the output for analysis in this study as $K_{est}x_t$, where K_{est} is a time-invariant matrix that almost perfectly approximates K_t. While the PPF's K_t does not necessarily converge in the same way it does in the KF, for all four analyzed sessions, neural activity mapped through $K_{est} \in R^{2xN}$ could account for 99.6, 99.6, 99.5, and 99.8 percent of the variance of the output respectively ($K_t x_t \cong K_{est}x_t$). Thus, when performing any analysis that makes a prediction of neural activity that then must be converted to a prediction of the output, we use the static linear mapping K_{est}. In addition, due to the accuracy of this linear approximation, we also match the timescale of the neural activity and outputs to that of Monkey G. In order to match timescales across the two animals (Monkey G: 100 ms updates, Monkey J: 5ms updates), Monkey J's outputs were aggregated into 100 ms bins by summing $K_{est}x_t$ over 20 consecutive 5ms bins to yield the aggregated output over 100ms. Correspondingly, Monkey J's neural activity was also summed into 100ms bins by summing x_t over 20 consecutive 5ms bins.

Neuroprosthetic tasks

Two neuroprosthetic tasks were performed – a centerout task and an obstacle avoidance task. We define a "condition" as the combination of the task performed, the target position to achieve, and in the obstacle avoidance task, the movement direction around the obstacle (clockwise or counterclockwise). Thus, there were up to 24 different conditions possible (8 centerout conditions,

8 clockwise obstacle conditions, 8 counterclockwise obstacle conditions). In practice, subjects mostly circumvented the obstacles for a given target location consistently in a clockwise or counterclockwise manner (as illustrated in Fig. 1B) resulting in an average of 16-17 conditions per session.

Centerout task:

The centerout task requires subjects to hold their cursor within a center target (Monkey J: radius = 1.2 cm, Monkey G: radius = 1.7 cm) for a specified period of time (Monkey J: hold = 0.25 sec, Monkey G: hold = 0.2 sec) before a go cue signals the animal to move their cursor to one of eight peripheral targets uniformly spaced around a circle. Each target was equidistant from the center starting target (Monkey J: distance = 13cm, Monkey G: distance = 10cm). Monkeys must then position their cursor within the peripheral target (Monkey J: target radius = 1.2cm, Monkey G: target radius = 1.7cm) for a specified period to time (Monkey J: hold = 0.25, Monkey G: hold = 0.2sec). Failure to acquire the target within a specified window (Monkey J: 3-10 sec, Monkey G: 10 sec) or to hold the cursor within the target for the duration of the hold period resulted in an error. Following successful completion of a target, a juice reward was delivered. Monkey J was required to move his cursor back to the center target to initiate a new trial, and Monkey G's cursor was automatically reset to the center target to initiate a new trial.

Obstacle avoidance task:

Monkey G performed an obstacle avoidance task with a very similar structure to the center-out task. The only difference was that a square obstacle (side length 2 or 3 cm) would appear in the workspace centered exactly in the middle of the straight line connecting the center target

position and peripheral target position. If the cursor entered the obstacle, the trial would end in an error and subjects would repeat the trial.

Monkey J's obstacle-avoidance task required a point-to-point movement between an initial (not necessarily center) target and another target. On arrival at the initial target, an ellipsoid obstacle appeared on the screen. If the cursor entered the obstacle at any time during the movement to the peripheral target, an error resulted and the trial was repeated. Target positions and obstacle sizes and positions were selected to vary the amount of obstruction, radius of curvature around the obstacles, and spatial locations of targets. Trials were constructed to include no obstruction, partial obstruction with low-curvature, full obstruction with a long distance between targets and full obstruction with a short distance between targets thus requiring a high curvature. See [33] for further details. Only trials that included partial obstruction or full obstruction were analyzed as "obstacle avoidance" trials.

Number of sessions

We analyzed 9 sessions of data from Monkey G and 4 sessions of data from Monkey J where on each session, monkeys performed both the centerout and obstacle tasks with the same decoder. Only successful trials were analyzed.

Output discretization

Data analysis of outputs relied on discretizing the continuous two-dimensional output. All outputs from rewarded trials in a given experimental session (including both tasks) were aggregated and discretized into 32 bins. Individual outputs were assigned to one of 8 angular bins (bin edges were 22.5, 67.5, 112.5, 157.5, 202.5, 247.5, 292.5, and 337.5 degrees) and one of four

magnitude bins. Angular bins were selected such that the straight line from the center to each of the center-out targets bisected each of the angular bins (Extended Data Fig. 1A). Magnitude bin edges were selected as the 23.75^{th}, 47.5^{th}, 71.25^{th}, and 95^{th} percentile of the distribution of output magnitudes for that experimental session. Outputs falling between the 95^{th} and 100^{th} percentile of magnitude were not analyzed to prevent very infrequent noise events from skewing the bin edges for output magnitude.

Cursor position and output trajectory visualization

Conditions per output

For each of the 32 outputs, the number of times the output was used in a given condition was tabulated. If the output was used >= 15 times for that condition within a given session, that condition was counted as "using" the output. The distribution of the number of conditions that used each output is shown in Extended Data Fig. 1B.

Visualization of cursor position subtrajectories

To visualize the cursor position trajectories for each condition around the occurrence of a given output, we computed the positon "subtrajectory," which is the average trajectory in a window locked to the occurrence of the given output. Cursor positions from successful trials for all conditions were aggregated for a representative session (session 0, Monkey G). A matrix of trajectories was formed by extracting a window of -500ms to 500ms (5 previous samples plus 5 proceeding samples) around each occurrence of the given output for each condition (total of $N_{out\text{-}cond}$ occurrences, yielding a 2 x 11 x $N_{out\text{-}cond}$ array) and averaged to yield a condition-specific output-locked average position trajectory (size: 2 x 11) for each condition. We define a

"subtrajectory" as an average output-locked trajectory. If an output falls in the first 500ms or last 500ms of a trial, its occurrence is not included in the subtrajectory calculation. The position subtrajectories were translated such that the occurrence of the output was set to (0, 0) in the 2D workspace (Fig. 2A, Extended Data Fig. 1C).

Visualization of output subtrajectories

To visualize trajectories of output for each condition around the occurrence of a given output, we computed the output subtrajectory as the output-locked average output trajectory in a window of -500ms to 500ms (Fig. 2A, Extended Data Fig. 1C). Trajectories of output from neural activity were aggregated using the same procedure as described above, yielding a 2x11x $N_{out\text{-}cond}$ size array of output trajectories locked to a given output for a specific condition. We note that this matrix consisted of the continuous, two-dimensional value of the outputs. Averaging over observations yielded the output subtrajectory (size: 2 x 11 array).

Matching the distribution of a given output between condition-specific and condition-pooled data

In many analyses, data (e.g. neural activity or an output-locked cursor trajectory) associated with an output and a specific condition is compared to data that pools across condition for that same output (Figs. 2-4, Extended Data Figs. 2, 4, 6). In a few cases, the precise continuous value of the output within the output's bin may systematically differ between condition-specific and condition-pooled datasets (which we will refer to below as "within-output-bin differences"). To ensure within-output-bin differences are not the source of significant differences between condition-specific and condition-pooled data for an output, we developed a procedure to subselect observations of condition-pooled output to match the mean of the condition-pooled output

distribution to the mean of the condition-specific output distribution when analyzing an output bin. This procedure is performed on all analyses comparing condition-specific data to a condition-pooled distribution of data. The matching procedure is as follows:

1. From the condition-specific distribution, compute the output mean $\mu_{out-cond}$ (size: 2x1) and output standard deviation $\sigma_{out-cond}$ (size: 2x1).

2. Compute the normalized values of each entry in the condition-pooled distribution by subtracting the $\mu_{out-cond}$ and dividing by $\sigma_{out-cond}$.

3. Compute the sum of the squared normalized values across the two dimensions yielding a single value ("cost") for each valid entry in the condition-pooled distribution. This cost corresponds to the deviation from the condition-specific output mean.

4. Remove entries constituting the top 5% of cost values in the condition-pooled distribution. These removed entries are the outputs with normalized values that are furthest from the condition-specific mean.

5. Use a Student's t-test to assess if the remaining entries in the condition-pooled distribution are significantly different than the condition-specific distribution for the first and second dimension (two p-values)

6. If both p-values are > 0.05, then the procedure is complete and the remaining entries in the condition-pooled distribution are considered the "output-matched condition-pooled distribution" for a specific output and condition.

7. If either or both p-values are < 0.05, return to step 4 and repeat.

Comparing condition-specific and condition-pooled output subtrajectories

To assess whether an output is used within significantly different trajectories in different conditions (Extended Data Fig. 1C-D), the following analysis is performed for conditions that have sufficient instances of an output (>=15):

1. The condition-specific output subtrajectory is computed by averaging over $N_{out\text{-}cond}$ single-trial output trajectories, as defined above in "Cursor position and output trajectory visualization".

2. The condition-pooled output subtrajectory is computed: all the single-trial output trajectories are pooled across trials from all conditions that use the given output to create a condition-pooled distribution of single-trial output trajectories. The condition-pooled output subtrajectory is computed by averaging over this distribution.

3. In order to test whether condition-specific output subtrajectories were significantly different from the condition-pooled output subtrajectory, $N_{out\text{-}cond}$ single-trial output trajectories were sampled from a condition-pooled distribution of output trajectories that was output-matched (see above, "Matching the distribution of a given output between condition-specific and condition-pooled data"). These $N_{out\text{-}cond}$ samples were then averaged to create a single subtrajectory, representing a plausible condition-specific subtrajectory if the condition-pooled distribution of trajectories was subsampled and averaged. This procedure was repeated 1000 times and used to construct a bootstrapped distribution of 1000 output subtrajectories. This distribution constituted expected variation in condition-specific output subtrajectories due to averaging over finite data.

4. The true condition-specific output subtrajectory difference from the condition-pooled output subtrajectory was computed (L2-norm between condition-specific 2x11 subtrajectory and condition-pooled 2x11 subtrajectory) and compared to the bootstrapped distribution differences: (L2-norm between each of the 1000 subsampled averaged 2x11 output subtrajectories and the

condition-pooled 2x11 output subtrajectory). A p-value for each output-condition subtrajectory difference was then derived.

The same analysis is also performed using only the next output following a given output (Extended Data Fig. 1E).

Analysis of neural activity issuing a given output

Invariant pattern view and behavior-preserving shuffle

We sought to test two predictions of the "invariant dynamics view": 1) that a particular output is issued by systematically different activity patterns in different movements, and 2) that activity transitions are predictable by invariant dynamics. We tested these predications against the "invariant pattern view", which captures how the same output could be issued by different patterns in different movements due to noise and captures how predictable activity transitions are simply due to the predictable behavior they generate.

Under the invariant pattern view, each observation of neural activity issuing a given output ("output activity") is drawn from an invariant distribution that is independent of condition. We estimated the distribution of neural activity expected by the invariant pattern view using a shuffling procedure (see *Shuffle procedure* below), allowing us to test whether experimental data possesses features that violate the invariant pattern view. Neural activity from each session is shuffled in a manner that preserved the output timeseries yet scrambled neural activity across trials and conditions (Fig. 2B). Thus, the resulting dataset (a single shuffle) has two properties: i) it has the same number of datapoints per output-condition as the original data, and ii) it produces the same behavior (output timeseries) as the original data.

The first property of the shuffle allows us to construct the invariant pattern view's distribution of average condition-specific output activity (activity averaged over observations of a given output in a given condition). Applying this shuffle in Fig. 2C-G, we tested if the difference between average condition-specific and condition-pooled output activity in the real data was significantly larger than the distribution of differences between condition-specific and condition-pooled output activity in the shuffled data.

The shuffled data's second property destroys the temporal ordering of neural activity as much as possible while preserving the temporal order of the issued outputs. Thus, the shuffled data allows us to test the significance of neural dynamics models fit on real neural activity transitions by comparing them to models fit on shuffle data that maintains the predictability in neural activity simply due to behavior but did not have further neural dynamics structure (Fig. 3C-G, Fig. 4CD, Fig. 4FG, Fig. 5EF).

Shuffle procedure

In order to create a shuffle for each animal on each session, all timebins from all trials from all conditions were collated. The continuous output at each timebin was discretized into its output bin. For each of the 32 discretized output bins, all timebins corresponding to a particular discretized output bin were identified. The activity in these identified timebins was then randomly permuted. A complete shuffled dataset was constructed by performing this random permutation for all discretized output bins. This full procedure was repeated 1000 times to yield 1000 shuffled datasets.

For analyses where the condition-specific output activity was computed and compared to the distribution of shuffled condition-specific output activity, an additional precaution was taken to

prevent "within-output-bin differences" from driving significant differences between condition-specific output activity and the shuffle distribution. Before shuffling all discretized outputs across the dataset, we checked whether the distribution of condition-pooled continuous output were significantly different than the distribution of continuous outputs for each condition. If the distributions differed for a particular condition, we output-matched (see above: "Matching the distribution of an output between condition-specific and condition-pooled data") the condition-pooled distribution and then only used the matched data to shuffle for this condition. The end result was a shuffled dataset with condition-specific output distributions that matched the mean of the real data's condition-specific output distributions.

Significantly different output-condition activity

For each session, output-conditions with >= 15 observations were analyzed. We analyzed the difference between condition-specific output activity and condition-pooled output activity, both for individual neurons and for the population's activity vector (Fig. 2E-G). Analysis of individual neurons for a given output-condition:

1. Compute the mean activity of individual neurons ($\mu_{out-cond} \in R^N$).

2. Compute the condition-pooled mean activity of individual neurons for the given output ($\mu_{out-pooled} \in R^N$). The output-matching procedure is used to account for within-bin output differences if needed.

3. Compute the absolute value of the difference between the condition-specific and condition-pooled means: $d\mu_{out-cond} = abs(\mu_{out-cond} - \mu_{out-pool}) \in R^N$.

4. Repeat steps 1-3 for each shuffled dataset *i*, yielding $d\mu_{shuff-i-out-cond}$ for $i = 1{:}\,1000$.

5. For each neuron *j,* compare $d\mu_{out-cond}(j)$ to the distribution of $d\mu_{shuff-i-out-cond}(j)$ for i = 1:1000. Differences greater than the 95th percentile of the shuffled distribution are deemed to have significantly different neuron *j* activity for an output-condition.

Analysis of population activity for a given output-condition:

The above procedure was repeated, but in step 3 and 4, the distance metric $d\mu_{out-cond}$ was replaced with a population distance metric (L2-norm, normalized by number of neurons): $d\mu_{pop-out-cond} = \left\|\mu_{out-cond} - \mu_{out-pool}\right\|_2/N$, $d\mu_{pop-out-cond} \in R^1$. In step 5, the single value $d\mu_{pop-out-cond}$ is compared to the distribution of $d\mu_{shuff-i-pop-out-cond}$ for i = 1:1000 to derive a p-value for each output-condition. The fraction of output-conditions with significant population activity differences is reported in Fig. 2E *left*.

Significantly different output activity, pooling over conditions

To test whether activity for an output significantly deviated from condition-pooled output activity (Fig. 2E *right*), we aggregated the difference between condition-specific and condition-pooled activity over all $Ncond$ conditions in which the output was used. An aggregate output difference is computed: $d\mu_{pop-out} = \frac{1}{Ncond}\sum_{j=1}^{Ncond} d\mu_{pop-out-j}$, and an aggregate shuffle distribution is computed: $d\mu_{shuff-i-pop-out} = \frac{1}{Ncond}\sum_{j=1}^{Ncond} d\mu_{shuff-i-pop-out-j}$. Then, $d\mu_{pop-out}$ is compared to the distribution of $d\mu_{shuff-i-pop-out}$ for $i = 1{:}1000$ to derive a p-value for each output. The fraction of outputs with significant population activity differences is reported in Fig. 2E *right*.

Significantly different activity for individual neurons, pooling over significant outputs and conditions

To test whether activity for a neuron significantly deviated from condition-pooled output activity (Fig. 2F), we aggregated the difference between condition-specific and condition-pooled output activity over the $Nout - cond$ output-conditions that were individually significant for population activity. An aggregate neuron difference for neuron n was computed: $d\mu(n) = \frac{1}{Nout-cond}\sum_{k=1}^{Nout}\sum_{j=1}^{Ncond} d\mu_{k-j}(n) * I_{sig}(j,k)$ where $d\mu_{k-j}(n)$ is the single neuron difference, and $I_{sig}(j,k)$ is an indicator that evaluates to 1 if (output k, condition j) was significant for population activity, and evaluates to 0 otherwise. This value was compared to the distribution of the aggregated shuffle: $d\mu_{shuff-i}(n) = \frac{1}{Nout-cond}\sum_{k=1}^{Nout}\sum_{j=1}^{Ncond} d\mu_{shuff-i-k-j}(n) * I_{sig}(j,k)$ for $i = 1{:}1000$ to derive a p-value for each neuron. The fraction of neurons with significant activity differences is reported in Fig. 2F.

Activity differences summary

Single neuron activity differences reported in Extended Data Fig. 2A are for individually significant neuron-output-conditions. We report raw differences in neuron activity as $d\mu_{out-cond}(j)$ (Extended Data Fig. 2A, *left*) and fraction differences as $\frac{d\mu_{out-cond}(j)}{\mu_{out-pool}(j)}$ (Extended Data Fig. 2A, *right*).

Population activity differences reported in Fig. 2FG and Extended Data Fig. 2B are for significantly different output-conditions. We report raw differences in population activity as $d\mu_{pop-out-cond}$ (Extended Data Fig. 2B, *left*) and fraction differences as $\frac{d\mu_{pop-out-cond}}{\|\mu_{out-pool}\|_2}$ (Fig. 2G, Extended Data Fig 2B, *right*).

Neural dynamics model and predictions

In order to test whether invariant dynamics could predict the different activity patterns issuing the same output for different conditions, a linear model of neural activity transitions was fit for each experimental session on training data of neural activity from all conditions and assessed on held-out test data. Neural activity at time *t,* x_t, was modeled as a linear function of x_{t-1}:

$$x_t = Ax_{t-1} + b$$

Here $A \in R^{NxN}$ modeled neural dynamics and $b \in R^N$ was an offset vector that allowed the model to identify non-zero fixed points of neural dynamics. Ridge regression was used to estimate the A and b parameters. Prior to any training or testing, data was collated such that all neural activity in bins from t=2:T_{trl} in all rewarded trials were paired with neural activity from t=1:(T_{trl}-1).

Estimation of Ridge Parameter

Data was randomly split into 5 sections, and a Ridge model with a ridge parameter varying from $2.5x10^{-5}$ to 10^6 was trained using 4 of the 5 sections and tested on the remaining test section. Test sections were rotated, yielding five estimates of R^2 for each ridge parameter. The ridge parameter yielding the highest cross-validated mean R^2 was selected for each experimental session. Ridge regression was primarily chosen due to a subset of sessions with a very high number of units (148 and 151 units), thus a high number of parameters needed to be estimated for the A matrix. Without regularization, these parameters tended to extreme values, and the model generalized poorly.

Fitting A, b

Once a ridge parameter was selected, A, b were training using 4/5 of the data. The remaining test data was predicted using the fit A, b . This procedure was repeated using a unique 1/5 of the dataset as the testing set such that predictions for a full test dataset were obtained after 5 iterations.

Predicting $x_{t+1} | \, x_t, A, b$

In Fig. 4C, we predict next activity x_{t+1} based on current activity x_t by taking the expected value according to our model: $E(x_{t+1}|x_t, A, b) = Ax_t + b$.

Predicting $output_{t+1} | \, x_t, A, b, K$

In Fig. 4D-G, we predict the next output $output_{t+1}$ based on current activity x_t by taking its expected value according to our model: $E(output_{t+1} | \, x_t, A, b, K) = K(Ax_t + b)$, where the decoder matrix *K* maps between neural activity and 2D outputs for cursor velocity. This amounts to first predicting next activity based on current activity as above $E(x_{t+1}|x_t, A, b) = Ax_t + b$ and then applying decoder *K*.

Predicting $x_t | \, x_{t-1}, A, b, K, output_t$

In Fig. 3C-G, we predict current activity x_t not only with knowledge of previous activity x_{t-1}, but also with knowledge of the current output $output_t$. We modeled x_t and x_{t-1} as jointly Gaussian with our dynamics model, and $output_t$ is jointly Gaussian with them since $output_t = Kx_t$. We modify our prediction of x_t based on knowledge of $output_t$: $E(x_t|x_{t-1}, A, b, K, output_t)$. Explicitly we conditioned on $output_t$, thereby ensuring that $K * E(x_t|x_{t-1}, A, b, K, c_t) = output_t$. To do this we wrote the joint distribution of x_t and $output_t$:

$$\begin{matrix} x_t \\ Kx_t \end{matrix} \sim N(\begin{pmatrix} \mu \\ K\mu \end{pmatrix}, \begin{pmatrix} \Sigma & (K\Sigma)^T \\ K\Sigma & K\Sigma K^T \end{pmatrix})$$

where $\mu = E(x_t|x_{t-1}, A, b) = Ax_{t-1} + b$, and $\Sigma = cov[x_t - (Ax_{t-1} + b)]$ is the covariance of the noise in the dynamics model. Then, the multivariate Gaussian conditional distribution provides the solution to conditioning on $output_t$:

$$E(x_t|x_{t-1}, A, b, output_t) = Ax_{t-1} + b + \Sigma^T K^T (K\Sigma K^T)^{-1}(output_t - K(Ax_{t-1} + b))$$

This prediction constrains the prediction of x_t to produce the given output $output_t$.

For these predictions, Σ is estimated following dynamics model fitting and set to the empirical error covariance between estimates of $E(x_t) = Ax_{t-1} + b$ and true x_t in the training data.

Predicting $x_t | K, output_t$

In Fig. 3C, as a comparison to the neural dynamics prediction ($x_t | x_{t-1}, A, b, K, output_t$), we predict x_t as its expected value based only the output $output_t$ it issues and the decoder matrix K. The same approach was used as above, except with empirical estimates of μ, Σ corresponding to the mean and covariance of the neural data.

$$\begin{matrix} x_t \\ Kx_t \end{matrix} \sim N(\begin{pmatrix} \mu \\ K\mu \end{pmatrix}, \begin{pmatrix} \Sigma & (K\Sigma)^T \\ K\Sigma & K\Sigma K^T \end{pmatrix})$$

This formulation makes the prediction:

$$E(x_t|output_t) = \mu + \Sigma^T K^T (K\Sigma K^T)^{-1}(output_t - K\mu)$$

Predicting x_t *with* A_{shuff}, b_{shuff}

In order to compare the predictions from the dynamics model to results that would be expected given the shuffled distribution, first, shuffled datasets were created (*Invariant pattern view and behavior preserving shuffle)*. Then, subsets of shuffled data were used to train a dynamics

model yielding parameters A_{shuff}, b_{shuff}. Finally, the shuffled model was applied to unshuffled data to make predictions, as described above.

Generalization analysis to test model invariance:

We assessed the invariance of neural dynamics models by testing if they generalized well when certain categories of activity were not included in the training data. Neural dynamics models were estimated after excluding activity in the following categories (Fig. 3BC, Fig. 4CD, Extended Data Fig. 4):

1. Left-out Output: For each output (total of 32 outputs), training data sets were constructed leaving out activity that transitioned to and from the given output (Fig. 3BC left, Fig. 4CD left, Extended Data Fig 4CD).

2. Left-out Condition: For each condition (consisting of target, task, and clockwise or counterclockwise movement for obstacle avoidance), training data sets were constructed leaving out activity for the given condition (Fig. 3BC right, Fig. 4CD right, Extended Data Fig. 4HI).

Further generalization analyses to test model invariance were performed (Extended Data Fig. 4) excluding larger amounts of data:

3. Left-out Output Angle: For each angular bin (total of 8 angular bins), training data sets were constructed leaving out activity that transitioned to and from the given output angle. This corresponds to leaving out activity for the 4 output bins that have the given angular bin but different magnitude bins (Extended Data Fig. 4B, middle).

4. Left-out Output Magnitude: For each magnitude bin (total of 4 magnitude bins), training data sets were constructed leaving out activity that transitioned to and from the given output

magnitude. This corresponds to leaving out activity for the 8 output bins that have the given magnitude bin but different angle bins (Extended Data Fig. 4B, right).

5. Left-out Classes of Conditions (Extended Data Fig. 4G):

   a. vertical condition class consisting of targets located at 90 and 270 degrees for both tasks,

   b. horizontal condition class consisting of targets located at 0 and 180 degrees for both tasks,

   c. diagonal 1 condition class consisting of targets located at 45 and 215 degrees for both tasks, and

   d. diagonal 2 condition class consisting of targets located at 135 and 315 degrees for both tasks).

For all of the listed categories above, many neural dynamics models were computed – each one corresponding to the exclusion of one element of the category. Each of the trained models was then used to predict the excluded data, together resulting in a full dataset of predictions. The R^2 of this predicted dataset reflected how well neural dynamics models could generalize to types of output-activity that were not observed during training. We note that Monkey J did not perform all conditions in the "diagonal 2" class, and so was not used in the analysis predicting excluded "diagonal 2" conditions.

Neural dynamics prediction of condition-specific output activity

In order to assess whether the neural dynamics model predicted condition-specific output activity better than expected from the neural dynamics model fit on the shuffled data, error between true and predicted condition-specific output activity (single neuron error and population distance)

was computed and compared to the error between the true and predicted condition-specific output activity from the shuffled dynamics model (Fig. 3F). We reported the fraction of output-conditions that were individually significant (Fig. 3F, left). Population activity error aggregated over conditions was used to determine if outputs were individually significant (Fig 3F, middle), and single-neuron error aggregated over outputs and conditions was used to determine if neurons were individually significant (Fig 3F, right).

Neural dynamics prediction of the condition-specific component of output activity:

The dynamics-predicted condition-specific output activity was computed: $\widehat{\mu_{out-cond}}$. The condition-specific component was estimated by subtracting $E(x_t|output_t)$: $\widehat{\mu_{out-cond}} - E(x_t|output_t)$. The variance of $\mu_{out-cond} - E(x_t|output_t)$ explained by $\widehat{\mu_{out-cond}} - E(x_t|output_t)$ is reported in Fig. 3M.

Decoder null-space dynamics:

In order to assess the predictions of the component of neural dynamics that predicts future neural activity but not future output, a dynamics model was fit to the component of neural activity in the decoder null space, yielding model parameters A_{null}, b_{null}. Predictions of neural activity using this model were then computed (Fig. 3B-D): $E(x_t|A_{null}, b_{null}) = A_{null}x_{t-1} + b_{null} + \mu_{decoder-space}$ where $\mu_{decoder-space} \in R^N$ is the mean of the component of neural activity in the decoder space.

Neural dynamics prediction of condition-specific next output:

For each output-condition, the true "next output" $output_{out-cond} \in R^2$ was compared to the dynamics predicted "next output" $\widehat{output}_{out-cond}$. The L2-norm of the difference was computed and compared to the errors obtained from the shuffled-dynamics predictions. "Next outputs" were significantly predicted if the error of the dynamics-predicted "next output" was less than the 5th percentile of the distribution of the errors of the shuffled-dynamics predictions (Fig. 4F, *left*). Outputs were determined to be individually significant if the error aggregated over conditions was significantly less than the shuffled-dynamics error aggregated over conditions (Fig. 4F, *center*).

Prediction of next output's direction relative to current output

To give a more intuitive metric of the neural dynamics model's prediction accuracy for the next output, we reported whether the neural dynamics model's predicted next output rotated in the correct direction (clockwise or counterclockwise) relative to the current output. Specifically, of all output-conditions with significant predictions of the next output, we reported the fraction which correctly predicted the next output's direction relative to the current output (Fig. 4F, *right*).

Eigenvalue characterization

Once *A, b* were estimated, *A* was analyzed to assess which dynamical modes were present (Extended Data Fig. 3). The eigenvalues of A were computed. From each eigenvalue, a frequency and time decay value were computed using the following equations:

Frequency = $\angle\lambda/(2\pi\Delta t)$ Hz if λ is complex, else frequency = 0 Hz

Time Decay = $\frac{-1}{\ln(|\lambda|)}\Delta t$ sec

Dynamical modes contributing substantially to predicting future neural variance will have time decays greater than the BMI decoder's binsize (here, 100ms).

Estimation of behavior-encoding models

To compare the explanatory power of neural dynamics to encoding models that encode more than just the output, we also fit a series of behavior-encoding models (Extended Data Fig. 5). Regressors included cursor state (position, velocity), target position (x,y postion in cursor workspace), and a categorical variable encoding target number, task, and if an obstacle-avoidance trial, the direction around the obstacle that the trial was performed (clockwise or counterclockwise). The same procedure described above (*Estimation of Ridge Parameter*) was followed with one additional step: prior to estimating the ridge parameter or fitting the regression, variables were z-scored. Without z-scoring, ridge regression may favor giving explanatory power to the variables with larger variances, since they would require smaller weights which ridge regression prefers.

Structure of neural dynamics predictions between pairs of conditions

We sought to assess whether the neural dynamics model predicted the overall structure of neural activity across outputs and movements, beyond the R^2 metric. Thus, we compared the neural dynamics prediction for various quantities across pairs of conditions.

Output activity

The neural dynamics model was used to predict the distance between activity patterns for the same output across pairs of conditions. The correlation between the two conditions' output activity population distance and the neural dynamics model's prediction of output activity population distance was computed and reported (Extended Data Fig. 6A-D).

Next output

The neural dynamics model was used to predict the distance between "next outputs" for the same given output across pairs of conditions. The correlation between the two conditions' next output distance and the neural dynamics model's prediction of next output distance was computed and reported (Extended Data Fig. 6J-L).

Correlation between output activity distance and output subtrajectory distance

We assessed whether the distance between output activity for two conditions was related to the distance between output subtrajectories for the same two conditions (Extended Data Fig. 6E-G), and whether neural dynamics could predict this correlation (Extended Data Fig. 6HI). For every output (that was used in more than five conditions) and pair of conditions that used the output, 1) the distances between condition-specific output activity were computed and 2) distances between output subtrajectories were computed. The output activity distances were correlated with the output subtrajectory distances (Extended Data Fig. 6FG).

To assess whether neural dynamics made predictions that maintained this structure, we performed that same analysis with distances between dynamics-predicted condition-specific output activity (Extended Data Fig. 6HI).

Feedback Control using Neural Population Dynamics

We sought to analyze how a neural population's invariant dynamics influences what inputs are needed to accomplish the centerout and obstacle tasks (Fig. 5). Thus, an optimal linear feedback controller (finite horizon linear quadratic regulator) was designed to control a neural

population to perform the centerout and obstacle avoidance tasks. The controller computed inputs to the neural population based on the current cursor state and current neural activity. The inputs were computed as the solution of an optimization problem that used knowledge of the target/task, decoder, and the neural population dynamics rules. We simulated 20 trials for each of 24 conditions: 8 center-out conditions, 8 clockwise obstacle-avoidance conditions, and 8 counterclockwise obstacle-avoidance conditions. The neural and cursor dynamics processes in the simulation are summarized below:

Neural Dynamics Driven with Input and Noise:

Given N neurons, neural activity $x_t \in R^N$ transitions in time with invariant dynamics $A \in R^{N\times N}$ acting on previous activity x_{t-1}, an activity offset $b \in R^N$, input matrix $B \in R^{N\times N}$ acting on inputs from the feedback controller $u_{t-1} \in R^N$, and noise $\sigma_{t-1} \in R^N$:

$$x_t = Ax_{t-1} + b + Bu_{t-1} + \sigma_{t-1}$$

The input matrix B was set to be the identity matrix such that each neuron has its own independent input. Each neuron also had its own independent noise (see *Noise* section below for how noise level was set).

An offset term was appended to x_t: $\begin{bmatrix} x_t \\ 1 \end{bmatrix} \in R^{N+1}$. This enabled incorporating the offset b into the neural dynamics matrix:

$$\begin{bmatrix} x_t \\ 1 \end{bmatrix} = \begin{bmatrix} A & b \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ 1 \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_{t-1} + \begin{bmatrix} \sigma_{t-1} \\ 0 \end{bmatrix}$$

BMI Cursor Dynamics

The cursor update equations for the simulation matched the steady state cursor update equations in the online BMI experiment (see "Output definition and analysis pre-processing" above):

$$c_t = Fc_{t-1} + Kx_{t-1}$$

As in the experiment, cursor state $c_t \in R^{N_c}$ where $N_c = 5$ was a vector consisting of two-dimensional position, velocity, and an offset: $[pos_x, pox_y vel_x, vel_y, 1]^T$. $K \in R^{N_c \times N}$ was the decoder's steady-state Kalman gain (Monkey G) or estimated equivalent K_{est} (Monkey J).

Neural and Cursor Dynamics Driven with Input and Noise

The feedback controller sent inputs to the neural population which were optimal considering the task goal, the cursor's current state, the dynamics rules of the neural population, and the neural population's current activity. To solve for the optimal input given all the listed quantities, first, the transitions of the input-driven neural and cursor states are jointly defined. Specifically, a full state vector is formed by appending the cursor state c_t to the neural activity state $\begin{bmatrix} x_t \\ 1 \end{bmatrix}$ to form $z_t \in R^{N+1+N_c}$:

$$z_t = \begin{bmatrix} x_t \\ 1 \\ c_t \end{bmatrix} = \begin{bmatrix} A & b & 0 \\ 0 & 1 & 0 \\ K & 0 & F \end{bmatrix} \begin{bmatrix} x_{t-1} \\ 1 \\ c_{t-1} \end{bmatrix} + \begin{bmatrix} B \\ 0 \\ 0 \end{bmatrix} u_{t-1} + \begin{bmatrix} \sigma_{t-1} \\ 0 \\ 0 \end{bmatrix}$$

In words, this expression defines a linear dynamical system where input u_{t-1} influences only the neural activity x_t, x_t evolves by dynamics A with offset vector b, and x_t drives c_t through the BMI decoder K. Finally, noise σ_{t-1} only influences neural activity x_t (see section below regarding noise).

Optimal Feedback Control Solution

The solution of u_t to achieve the desired cursor behavior is as follows. In the finite horizon LQR model, the optimal control sequence $(u_t, t = 0, 1, \dots T-1)$ is computed by minimizing the following cost function:

$$J(u_{0:T-1}) = \left(\sum_{t=0}^{T-1} \left((z_t - z_{targ})^T Q (z_t - z_{targ}) + u_t^T R u_t\right)\right) + \left(z_T - z_{targ}\right)^T Q_T (z_T - z_{targ})$$

Here, z_{targ} is a target state that can be set such that the controller aims to minimize the state error $z_t - z_{targ}$. In this implementation, $Q = 0 \in R^{(N+1+N_c)\times(N+1+N_c)}$, $R = I \in R^{N\times N}$, and $Q_T = \begin{bmatrix} 0 \in R^{N\times N} & 0 & 0 \\ 0 & 0 \in R^1 & 0 \\ 0 & 0 & I * 10^8 \in R^{N_c \times N_c} \end{bmatrix} \in R^{(N+1+N_c)\times(N+1+N_c)}$. Thus, the neural state and cursor state are not penalized throughout the controlled movement. Only the final cursor state error is heavily penalized. The magnitude of the input to neural activity is penalized by setting R as non-zero, causing the controller to send the minimal input to the neural population to produce task behavior.

The optimal control sequence $(u_t, t = 0, 1, \dots T-1)$ is given by $u_t = K_t^{lqr}(z_t - z_{targ})$ where feedback gain matrices $(K_t^{lqr}, t = 0, 1, \dots T-1)$ are computed iteratively solving the dynamic Ricatti equation backwards in time. We note that we computed the LQR solution for u_t using the dynamics of state error $z_t - z_{targ}$, and that the dynamics of state error for non-zero target states are affine rather than strictly linear.

In this implementation, $c_{target} = [targ_{posx}, targ_{posy}, 0, 0, 1]$ for the final target positions to ensure the cursor arrives at the target position. (See section below about simulating cursor movements to circumvent an obstacle.)

Simulations were then run initializing cursor position at $c_0 = [0, 0, 0, 0, 1]$. Target states (z_{targ}) were set to zero, with the exception of the cursor position set to the desired target location.

Targets were positioned 10cm away from the origin (same target arrangement as Monkey G). Target cursor velocity was set to zero to enforce that the cursor should stop at the desired target location.

Exact decoder parameters from Monkey G and linearized decoder parameters from Monkey J were used (F, K) in simulations. The neural dynamics model estimated from data (A, b) from both animals were used in their respective simulations. The horizon for each trial to hit its target state was set to be $T = 40$ (corresponding to 4 seconds if we consider the BMI's timebin of 100ms). This facilitated comparative analysis of behavior, as each trial was of equal length. We verified all of our simulated trials completed their tasks successfully.

Simulations of obstacle-avoidance movements

We performed simulations controlling the cursor around obstacles to a final target, in both clockwise and counterclockwise movements. In order to direct cursor movements around the obstacle, we defined a waypoint as an intermediate state the cursor had to reach enroute to the final target. Concretely, for the first segment of the movement, a controller with a horizon T=20 directed the cursor to the waypoint, and then a controller with horizon T=20 directed the cursor from the waypoint to the final target.

The waypoint was defined relative to the obstacle position as follows. First the vector between the center target and the obstacle position was determined ($v_{obs,center}$). The $v_{obs,center}$ was then rotated either +90 degrees or -90 degrees corresponding to clockwise and counterclockwise movements. The waypoint position was a 6cm distance in the direction of the rotated vector, from the obstacle center. Finally, the desired velocity of the intermediate target was set to be in the direction of $v_{obs,center}$, with a magnitude of 10 cm/s, so that the cursor would

be moving in a direction consistent with reaching its final target in the second segment of the movement after the waypoint was reached.

To compute the inputs to execute these movements, we defined the state error at each time t as $z_{error} = z_{targ} - z_t$, where z_{targ} was the waypoint for the first half of the movement, and z_{targ} was the final target for the second half of the movement. The linear quadratic regulator feedback gain K_t^{lqr} matrices were computed on the appropriate state error dynamics with the shortened horizon T=20.

We note that solving the linear control problem to the final target with the constraint of avoiding an obstacle is not a convex problem, and that this solution for the whole movement (including waypoint and final target) has no certificate of optimality. Our heuristic solution performs optimal control from the start position to the waypoint, and then optimal control from the waypoint to the final target. Importantly, this solution minimizes the amount of input needed to accomplish these goals.

Invariant Dynamics Model: A,b experimentally observed

The "Invariant Dynamics Model" consisted of optimal feedback control simulations where the experimentally-observed dynamics models were used.

No Dynamics Model: A=0

The Invariant Dynamic Model was compared to the "No Dynamics Model," consisting of simulations where *A* was set to zero, but the offset *b* remained non-zero. The No Dynamics Model removed effects of neural dynamics on feedback control.

Noise

In order to simulate feedback control of neural dynamics under noisy conditions, noise with fixed variance was added to each neuron at each timestep: $x_t = Ax_{t-1} + Bu_{t-1} + \sigma_{t-1}$, where $\sigma_t \sim N(0, aI)$, and a was set to the average variance of an individual neuron estimated from simulations of the No Dynamics Model run without noise. Thus, the overall level of added noise (the sum of noise variance over neurons) matched the overall level of signal in the noiseless No Dynamics Model simulation (sum of activity variance over neurons).

Analyzing input magnitude

For each simulated trial, we computed the input magnitude as the L2 norm of the input matrix $u_t \in R^{N \times T}$ (where $T = 39$ was the horizon and thus movement length). We averaged input magnitude over the 20 trials for each of 24 conditions. We then compared input magnitude between the Invariant Dynamics Model and the No Dynamics Model (see Statistics below).

Analyzing activity issuing a given output

In the feedback control model, we sought to verify if distinct activity patterns were used to issue the same output across different conditions. We applied the same analysis steps from experimental neural data to the feedback control simulations and then compared results across the simulations for the Invariant Dynamics Model and No Dynamics Model. We defined discretized output bins and calculated the average neural activity for each output and each condition. For output-conditions with >=15 observations of neural activity, we computed the distance between condition-specific output activity and condition-pooled output activity by subtracting the activity, taking the L2 norm, and normalizing by the number of neurons, as in the experimental data

analysis. We then compared activity distances between the Invariant Dynamics Model and the No Dynamics Model (see Statistics below).

To visualize the results (Fig. 5EF), we computed the shuffle distribution of distances between condition-specific output activity and condition-pooled output activity by performing 1000 iterations of the behavior-preserving shuffle. For visualization of activity distances relative to the shuffle chance rate, we subtracted the simulation's activity distances by the mean of the shuffle distribution (same as in Fig. 2D).

Statistics

In many analyses, we assessed whether a condition-specific quantity was significantly larger than expected from the distribution of the quantity expected if it were computed after subsampling the condition-pooled distribution. A p-value was computed by comparing the condition-specific quantity to the distribution of quantities computed from subsampling the condition-pooled distribution. (We note that the subsampling of the condition-pooled distribution is done by performing the "behavior-producing shuffle".)

The following is a summary of these analyses:

- Extended Data Fig. 1D, Quantity: distance between condition-specific output subtrajectory and condition-pooled output subtrajectory, P-value: computed using behavior-preserving shuffle.
- Extended Data Fig. 1E, Quantity: distance between condition-specific next output and the condition-pooled next output, P-value: computed using behavior-preserving shuffle.

- Fig. 2CF: Quantity: distance between condition-specific output activity for a neuron and condition-pooled output activity for a neuron., P-value: behavior-preserving shuffle.
- Fig. 2DE: Quantity: distance between condition-specific output population activity and condition-pooled output population activity, P-value: behavior-preserving shuffle.
- Fig. 3F *right:* Quantity: error between the neural dynamics' prediction of condition-specific output activity for a neuron and the true condition-specific output activity for the neuron. P-value: distribution of prediction errors from neural dynamics models fit on behavior-preserving shuffle (and that made predictions using unshuffled data).
- Fig. 3F *left, middle:* Quantity: error between the neural dynamics' prediction of condition-specific output population activity and the true condition-specific output population activity. P-value: distribution of prediction errors from neural dynamics models fit on behavior-preserving shuffle (and that made predictions using unshuffled data).
- Fig. 4F: Quantity: error between the neural dynamics' prediction of condition-specific next output and true condition-specific next output. P-value: distribution of prediction errors from neural dynamics models fit on behavior-preserving shuffle (and made predictions using unshuffled data).

In the above analyses, we also assessed the fraction of condition-specific quantities that were significantly different from the condition-pooled quantities or significantly predicted compared to a shuffled distribution (Fig. 2EF, Fig. 3F, Fig. 4F, Extended Data Fig. 4DI, Extended Data Fig. 6GIL). In order to aggregate over all data to determine whether condition-specific quantities were significantly different or significantly predicted within a session, we averaged the condition-specific quantity over the relevant dimensions (condition, output, and/or neuron) to yield

a single aggregated value for a session. For example, in Fig. 2F we take the distance between average activity for a neuron-output-condition and condition-pooled average activity for a neuron-output, and we average the quantity over condition and output to yield an aggregated value that is used to assess if individual neurons are individually significant. We correspondingly averaged the shuffle distribution across all relevant dimensions (condition, output, and/or neuron). Together this procedure yielded a single aggregated value that could be compared to a single aggregated distribution to determine session significance. Finally, when we sought to aggregate over sessions, we took the condition-specific quantity that was aggregated within a session and averaged it across sessions and again compared it to a shuffle distribution of this value aggregated over sessions.

When R^2 was the metric assessed (Fig. 3CG, Fig. 4CD, Extended Data Fig. 4BFG) a single R^2 metric was computed for each session and compared to the R^2 distribution from performing the same analysis or prediction using behavior-preserving shuffles.

In some cases, a linear regression was fit between two quantities (Extended Data Fig. 6C-D, J-L) on both individual sessions and on data pooled over all sessions, and the significance of the fit and correlation coefficient were both reported. In other cases where random effects such as session or analyzed output may have influenced the linear regression parameters (Extended Data Fig. 6E-I), a Linear Mixed Effect (LME) model was used with session and/or output modeled as random effects on intercept.

In Extended Data Fig. 5, a paired Student's t-test was used to compare two models' R^2 metric across sessions.

In Fig. 5, we compared data produced by two models of feedback control of BMI. To analyze the magnitude of input that neurons controlling BMI received (Fig. 5C), we compared the magnitude of input used by the Invariant Dynamics Model and the No Dynamics Model for

each experimental session with a paired Wilcoxon test, where each pair consisted of one condition (24 conditions total). To assess significance across sessions, a linear mixed effect model between input magnitude and model category was used, with session modeled as a random effect. To analyze distances between condition-specific output activity and condition-pooled output activity (Fig. 5F), we compared population activity distances produced by the Invariant Dynamics Model and the No Dynamics Model using a 2-sample KS test for each session. We also pooled across sessions for each subject using a LME model between activity distance and model category, with session modeled as a random effect.

Acknowledgements

We thank I. Rodrigues-Vaz, D. Peterka, and I. Papusha for helpful discussions, and the Costa and Carmena labs for their support.

Funding

BRAIN Initiative National Institute of Mental Health postdoctoral fellowship
1F32MH118714-01 (VRA)

BRAIN Initiative National Institute of Mental Health postdoctoral fellowship
1F32MH120891-01 (PK)

NINDS/NIH BRAIN Initiative U19
NS104649 (RMC)

Simons-Emory International Consortium on Motor Control
#717104 (RMC)

NINDS/NIH R01
NS106094 (JMC)

Author Contributions

V.R.A., P.K., R.M.C., and J.M.C. conceived and designed this study. P.K., S.G., and A.L.O. performed the experiments. P.K. and V.R.A. analyzed the data. All authors contributed

materials and analysis tools. V.R.A., P.K., R.M.C, and J.M.C. wrote the manuscript. All authors reviewed the manuscript.

Competing Interests statement

Authors declare that they have no competing interests.

References

1. Porter, R. & Lemon, R. *Corticospinal Function and Voluntary Movement*. *Monographs of the Physiological Society* (Oxford University Press, 1995). doi:10.1093/acprof:oso/9780198523758.001.0001
2. Arber, S. & Costa, R. M. Connecting neuronal circuits for movement. *Science (80-.).* **360**, 1403–1404 (2018).
3. Russo, A. A. *et al.* Motor Cortex Embeds Muscle-like Commands in an Untangled Population Response. *Neuron* **97**, 953-966.e8 (2018).
4. Shenoy, K. V, Sahani, M. & Churchland, M. M. Cortical Control of Arm Movements: A Dynamical Systems Perspective. *Annu. Rev. Neurosci.* **36**, 337–359 (2013).
5. Sussillo, D., Churchland, M. M., Kaufman, M. T. & Shenoy, K. V. A neural network that finds a naturalistic solution for the production of muscle activity. *Nat. Neurosci.* **18**, 1025–33 (2015).
6. Hennequin, G., Vogels, T. P. & Gerstner, W. Optimal control of transient dynamics in balanced networks supports generation of complex movements. *Neuron* **82**, 1394–1406 (2014).
7. Fetz, E. E. Are movement parameters recognizably coded in the activity of single neurons? *Behav. Brain Sci.* **15**, 679–690 (1992).
8. Mastrogiuseppe, F. & Ostojic, S. Linking Connectivity, Dynamics, and Computations in Low-Rank Recurrent Neural Networks. *Neuron* **99**, 609-623.e29 (2018).
9. Churchland, M. M. *et al.* Neural population dynamics during reaching. *Nature* **487**, 51–56 (2012).
10. Kao, J. C. *et al.* Single-trial dynamics of motor cortex and their applications to brain-machine interfaces. *Nat. Commun.* **6**, 7759 (2015).
11. Michaels, J. A., Dann, B. & Scherberger, H. Neural Population Dynamics during Reaching Are Better Explained by a Dynamical System than Representational Tuning. *PLoS Comput. Biol.* **12**, (2016).
12. Liang, K.-F. & Kao, J. C. Deep Learning Neural Encoders for Motor Cortex. *IEEE Trans. Biomed. Eng.* **67**, 2145–2158 (2020).
13. Truccolo, W., Hochberg, L. R. & Donoghue, J. P. Collective dynamics in human and monkey sensorimotor cortex: Predicting single neuron spikes. *Nat. Neurosci.* **13**, 105–111 (2010).
14. Kao, J. C., Ryu, S. I. & Shenoy, K. V. Leveraging neural dynamics to extend functional lifetime of brain-machine interfaces. *Sci. Rep.* **7**, 7395 (2017).
15. Pandarinath, C. *et al.* Inferring single-trial neural population dynamics using sequential auto-encoders. *Nat. Methods* **15**, 805–815 (2018).
16. Abbaspourazad, H., Choudhury, M., Wong, Y. T., Pesaran, B. & Shanechi, M. M. Multiscale low-dimensional motor cortical state dynamics predict naturalistic reach-and-

grasp behavior. *Nat. Commun.* **12**, 607 (2021).
17. Gallego, J. A., Perich, M. G., Chowdhury, R. H., Solla, S. A. & Miller, L. E. Long-term stability of cortical population dynamics underlying consistent behavior. *Nat. Neurosci.* **23**, 260–270 (2020).
18. Sani, O. G., Abbaspourazad, H., Wong, Y. T., Pesaran, B. & Shanechi, M. M. Modeling behaviorally relevant neural dynamics enabled by preferential subspace identification. *Nat. Neurosci.* **24**, 140–149 (2021).
19. Reimer, J. & Hatsopoulos, N. G. The problem of parametric neural coding in the motor system. *Adv. Exp. Med. Biol.* **629**, 243–259 (2009).
20. Omrani, M., Kaufman, M. T., Hatsopoulos, N. G. & Cheney, P. D. Perspectives on classical controversies about the motor cortex. *J. Neurophysiol.* **118**, 1828–1848 (2017).
21. Taylor, D. M., Tillery, S. I. H. & Schwartz, A. B. Direct cortical control of 3D neuroprosthetic devices. *Science (80-.).* **296**, 1829–1832 (2002).
22. Serruya, M. D., Hatsopoulos, N. G., Paninski, L., Fellows, M. R. & Donoghue, J. P. Instant neural control of a movement signal. *Nature* **416**, 141–2 (2002).
23. Carmena, J. M. *et al.* Learning to control a brain-machine interface for reaching and grasping by primates. *PLoS Biol.* **1**, 193–208 (2003).
24. Ganguly, K. & Carmena, J. M. Emergence of a stable cortical map for neuroprosthetic control. *PLoS Biol.* **7**, (2009).
25. Hennig, J. A. *et al.* Constraints on neural redundancy. *Elife* **7**, 1–34 (2018).
26. Kaufman, M. T., Churchland, M. M., Ryu, S. I. & Shenoy, K. V. Cortical activity in the null space: permitting preparation without movement. *Nat. Neurosci.* **17**, 440–8 (2014).
27. Stavisky, S. D., Kao, J. C., Ryu, S. I. & Shenoy, K. V. Motor Cortical Visuomotor Feedback Activity Is Initially Isolated from Downstream Targets in Output-Null Neural State Space Dimensions. *Neuron* 1–14 (2017). doi:10.1016/j.neuron.2017.05.023
28. Elsayed, G. F., Lara, A. H., Kaufman, M. T., Churchland, M. M. & Cunningham, J. P. Reorganization between preparatory and movement population responses in motor cortex. *Nat. Commun.* 13239 (2016). doi:10.1038/ncomms13239
29. Sauerbrei, B. A. *et al.* Cortical pattern generation during dexterous movement is input-driven. *Nature* **577**, 386–391 (2020).
30. Kao, T. & Hennequin, G. Neuroscience out of control : control-theoretic perspectives on neural circuit dynamics. *Curr. Opin. Neurobiol.* **58**, 122–129 (2019).
31. Athalye, V. R., Carmena, J. M. & Costa, R. M. Neural reinforcement: re-entering and refining neural dynamics leading to desirable outcomes. *Curr. Opin. Neurobiol.* **60**, 145–154 (2020).
32. Shanechi, M. M., Orsborn, A. L. & Carmena, J. M. Robust Brain-Machine Interface Design Using Optimal Feedback Control Modeling and Adaptive Point Process Filtering. *PLoS Comput. Biol.* **12**, e1004730 (2016).
33. Shanechi, M. M. *et al.* Rapid control and feedback rates enhance neuroprosthetic control. *Nat. Commun.* **8**, (2017).
34. Dangi, S. *et al.* Continuous closed-loop decoder adaptation with a recursive maximum likelihood algorithm allows for rapid performance acquisition in brain-machine interfaces. *Neural Comput.* **26**, 1811–1839 (2014).
35. Elsayed, G. F. & Cunningham, J. P. Structure in neural population recordings: An expected byproduct of simpler phenomena? *Nat. Neurosci.* **20**, 1310–1318 (2017).
36. Stavisky, S. D. *et al.* Brain-machine interface cursor position only weakly affects monkey

and human motor cortical activity in the absence of arm movements. *Sci. Rep.* **8**, 1–19 (2018).
37. Evarts, E. V. Relation of pyramidal tract activity to force exerted during voluntary movement. *J. Neurophysiol.* **31**, 14–27 (1968).
38. Kalaska, J. F. From intention to action: motor cortex and the control of reaching movements. *Adv. Exp. Med. Biol.* **629**, 139–178 (2009).
39. Fetz, E. E. & Cheney, P. D. Postspike facilitation of forelimb muscle activity by primate corticomotoneuronal cells. *J. Neurophysiol.* **44**, 751–772 (1980).
40. Schieber, M. H. & Rivlis, G. Partial reconstruction of muscle activity from a pruned network of diverse motor cortex neurons. *J. Neurophysiol.* **97**, 70–82 (2007).
41. Morrow, M. M. & Miller, L. E. Prediction of muscle activity by populations of sequentially recorded primary motor cortex neurons. *J. Neurophysiol.* **89**, 2279–2288 (2003).
42. Overduin, S. A., d'Avella, A., Roh, J., Carmena, J. M. & Bizzi, E. Representation of muscle synergies in the primate brain. *J. Neurosci.* **35**, 12615–12624 (2015).
43. Holdefer, R. N. & Miller, L. E. Primary motor cortical neurons encode functional muscle synergies. *Exp. Brain Res.* **146**, 233–243 (2002).
44. Cheney, P. D. & Fetz, E. E. Functional classes of primate corticomotoneuronal cells and their relation to active force. *J. Neurophysiol.* **44**, 773–791 (1980).
45. Ajemian, R. *et al.* Assessing the Function of Motor Cortex: Single-Neuron Models of How Neural Response Is Modulated by Limb Biomechanics. *Neuron* **58**, 414–428 (2008).
46. Sergio, L. E., Hamel-pâquet, C. & Kalaska, J. F. Motor Cortex Neural Correlates of Output Kinematics and Kinetics During Isometric-Force and Arm-Reaching Tasks Motor Cortex Neural Correlates of Output Kinematics and Kinetics During Isometric-Force and Arm-Reaching Tasks. *J. Neurophysiol.* **94**, 2353–2378 (2005).
47. Georgopoulos, A. P., Caminiti, R. & Kalaska, J. F. Static spatial effects in motor cortex and area 5: Quantitative relations in a two-dimensional space. *Exp. Brain Res.* **54**, 446–454 (1984).
48. Wang, W., Chan, S. S., Heldman, D. A. & Moran, D. W. Motor cortical representation of position and velocity during reaching. *J. Neurophysiol.* **97**, 4258–4270 (2007).
49. Paninski, L., Fellows, M. R., Hatsopoulos, N. G. & Donoghue, J. P. Spatiotemporal Tuning of Motor Cortical Neurons for Hand Position and Velocity. *J. Neurophysiol.* **91**, 515–532 (2004).
50. Fu, Q.-G., Suarez, J. I. & Ebner, T. J. Neuronal Specification of Direction and Distance During Reaching Movements in the Superior Precentral Premotor Area and Primary Motor Cortex of Monkeys. *J. Neurophysiol.* **70**, (1993).
51. Moran, D. W. & Schwartz, A. B. Motor cortical representation of speed and direction during reaching. *J. Neurophysiol.* **82**, 2676–2692 (1999).
52. Flament, D. & Hore, J. Relations of motor cortex neural discharge to kinematics of passive and active elbow movements in the monkey. *J. Neurophysiol.* **60**, 1268–1284 (1988).
53. Georgopoulos, A. P., Kalaska, J. F., Caminiti, R. & Massey, J. T. On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex. *J. Neurosci.* **2**, 1527–1537 (1982).
54. Georgopoulos, A. P., Schwartz, A. B. & Kettner, R. E. Neuronal population coding of movement direction. *Science (80-.).* **233**, 1416–9 (1986).

55. Churchland, M. M., Cunningham, J. P., Kaufman, M. T., Ryu, S. I. & Shenoy, K. V. Cortical Preparatory Activity: Representation of Movement or First Cog in a Dynamical Machine? *Neuron* **68**, 387–400 (2010).
56. Todorov, E. & Jordan, M. I. Optimal feedback control as a theory of motor coordination. *Nat Neurosci* **5**, 1226–35. (2002).
57. Kao, T. C., Sadabadi, M. S. & Hennequin, G. Optimal anticipatory control as a theory of motor preparation: A thalamo-cortical circuit model. *bioRxiv* 1–15 (2020). doi:10.1101/2020.02.02.931246
58. Gallego, J. A. *et al.* Cortical population activity within a preserved neural manifold underlies multiple motor behaviors. *Nat. Commun.* **9**, 1–13 (2018).
59. Gallego, J. A., Perich, M. G., Miller, L. E. & Solla, S. A. Neural Manifolds for the Control of Movement. *Neuron* **94**, 978–984 (2017).
60. Miri, A. *et al.* Behaviorally Selective Engagement of Short-Latency Effector Pathways by Motor Cortex. *Neuron* **95**, 683-696.e11 (2017).
61. Suresh, A. K. *et al.* Neural population dynamics in motor cortex are different for reach and grasp. *Elife* **9**, (2020).
62. Rokni, U. & Sompolinsky, H. How the brain generates movement. *Neural Comput.* **24**, 289–331 (2012).
63. Churchland, M. M. & Cunningham, J. P. A Dynamical Basis Set for Generating Reaches. *Cold Spring Harb. Symp. Quant. Biol.* **79**, 67–80 (2014).
64. Logiaco, L., Abbott, L. F. & Escola, S. Thalamic control of cortical dynamics in a model of flexible motor sequencing. *Cell Rep.* **35**, 109090 (2021).
65. Mannella, F. & Baldassarre, G. Selection of cortical dynamics for motor behaviour by the basal ganglia. *Biol. Cybern.* **109**, 575–595 (2015).
66. Vyas, S. *et al.* Neural Population Dynamics Underlying Motor Learning Transfer. *Neuron* **97**, 1177-1186.e3 (2018).
67. Perich, M. G., Gallego, J. A. & Miller, L. E. A Neural Population Mechanism for Rapid Learning. *Neuron* **100**, 964-976.e7 (2018).
68. Sadtler, P. T. *et al.* Neural constraints on learning. *Nature* **512**, 423–426 (2014).
69. Athalye, V. R., Ganguly, K., Costa, R. M. & Carmena, J. M. Emergence of Coordinated Neural Dynamics Underlies Neuroprosthetic Learning and Skillful Control. *Neuron* **93**, 955–970 (2017).
70. Athalye, V. R., Santos, F. J., Carmena, J. M. & Costa, R. M. Evidence for a Neural Law of Effect. *Science (80-.).* **359**, 1024–1029 (2018).
71. Athalye, V. R., Santos, F. J., Carmena, J. M. & Costa, R. M. Evidence for a neural law of effect. *Science (80-.).* **359**, 1024–1029 (2018).
72. Koralek, A. C., Jin, X., Long II, J. D., Costa, R. M. & Carmena, J. M. Corticostriatal plasticity is necessary for learning intentional neuroprosthetic skills. *Nature* **483**, 331–335 (2012).
73. Neely, R. M., Koralek, A. C., Athalye, V. R., Costa, R. M. & Carmena, J. M. Volitional Modulation of Primary Visual Cortex Activity Requires the Basal Ganglia. *Neuron* **97**, 1356–1368 (2018).
74. Willett, F. R., Avansino, D. T., Hochberg, L. R., Henderson, J. M. & Shenoy, K. V. High-performance brain-to-text communication via handwriting. *Nature* **593**, 249–254 (2021).
75. Khanna, P. *et al.* Low-frequency stimulation enhances ensemble co-firing and dexterity after stroke. *Cell* **184**, 912-930.e20 (2021).

76. Ramanathan, D. S. *et al.* Low-frequency cortical activity is a neuromodulatory target that tracks recovery after stroke. *Nat. Med.* **24**, 1257–1267 (2018).
77. Shenoy, K. & Carmena, J. Combining decoder design and neural adaptation in brain-machine interfaces. *Neuron* **84**, 665–680 (2014).
78. Golub, M. D., Chase, S. M., Batista, A. P. & Yu, B. M. Brain-computer interfaces for dissecting cognitive processes underlying sensorimotor control. *Curr. Opin. Neurobiol.* **37**, 53–58 (2016).
79. Orsborn, A. L. & Pesaran, B. Parsing learning in networks using brain-machine interfaces. *Curr. Opin. Neurobiol.* **46**, 76–83 (2017).
80. Moxon, K. A. & Foffani, G. Brain-Machine Interfaces beyond Neuroprosthetics. *Neuron* **86**, 55–67 (2015).
81. Paxinos, G., Huang, X.-F. & Toga, A. W. The Rhesus Monkey Brain in Stereotaxic Coordinates. (2013).
82. Gilja, V. *et al.* A high-performance neural prosthesis enabled by control algorithm design. *Nat. Neurosci.* **15**, 1752–1757 (2012).
83. Wu, W., Gao, Y., Bienenstock, E., Donoghue, J. P. & Black, M. J. Bayesian Population Decoding of Motor Cortical Activity Using a Kalman Filter. *Neural Comput.* **18**, 80–118 (2006).
84. Dangi, S., Orsborn, A. L., Moorman, H. G. & Carmena, J. M. Design and Analysis of Closed-Loop Decoder Adaptation Algorithms for Brain-Machine Interfaces. *Neural Comput.* **25**, 1693–1731 (2013).
85. Malik, W. Q., Truccolo, W., Brown, E. N. & Hochberg, L. R. Efficient decoding with steady-state kalman filter in neural interface systems. *IEEE Trans. Neural Syst. Rehabil. Eng.* **19**, 25–34 (2011).
86. Gowda, S., Orsborn, A. L., Overduin, S. A., Moorman, H. G. & Carmena, J. M. Designing dynamical properties of brain-machine interfaces to optimize task-specific performance. *IEEE Trans. Neural Syst. Rehabil. Eng.* **22**, 911–920 (2014).

Figures

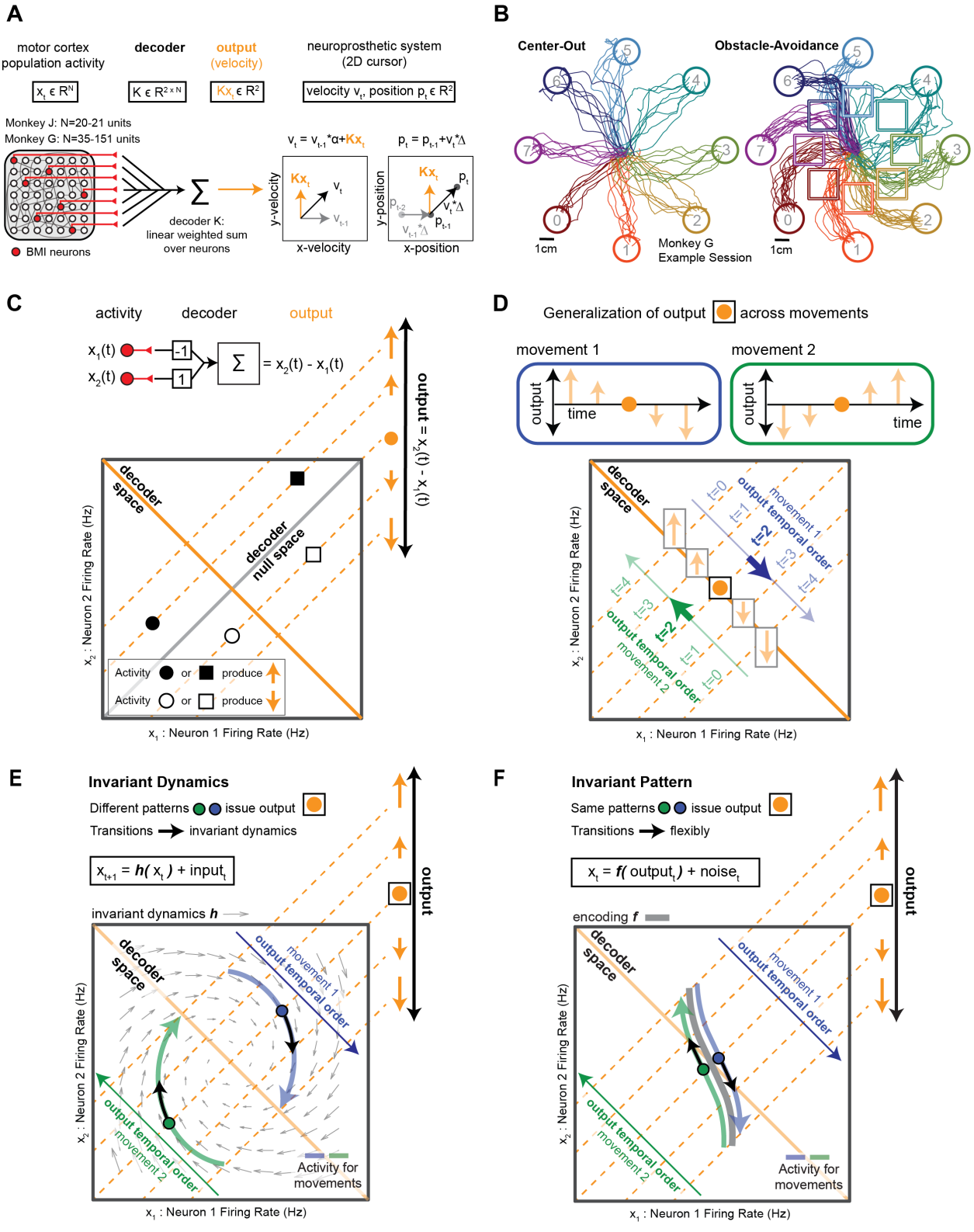

Figure 1. Brain-machine interface paradigm to study how an output is generalized across movements.

(A) Schematic of neural population control of a brain-machine interface (BMI). The experimenter defines an observed population of neurons in motor cortex to directly control a two-dimensional neuroprosthetic cursor. Outputs are the behavior-controlling signal which the BMI decoder reads out from the motor cortex population by performing a weighted linear summation over the neurons. Outputs update the velocity of the neuroprosthetic cursor which updates the cursor's position.

(B) Illustration of single-trial BMI cursor movements from an example session from Monkey G. Conditions of movement are defined by the target (gray target numbers 0 through 7), task (center-out or obstacle avoidance), and the direction the cursor moves around the obstacle during the obstacle avoidance task (clockwise or counterclockwise). The center-out task elicits straight movements and the obstacle-avoidance task elicits curved movements.

(C) Illustration of the relationship between neural activity and outputs, using 2 neurons and a one-dimensional output. Many population activity patterns can issue the same output because the output integrates over the activity of many individual neurons. For simplicity, the illustration defines the output as neuron 2 activity minus neuron 1 activity, symbolized with orange arrows (top right) indicating the output's magnitude and sign. We visualize the relationship between activity and outputs in neural activity space, where each axis is one neuron's firing rate and a dot in the space corresponds to a population activity pattern. Outputs correspond to the projection of neural activity onto the solid orange axis (decoder space). Components of neural activity in the decoder null space (solid gray axis) do not affect outputs. Producing a given output (top right) can be achieved by any activity pattern along the corresponding dotted orange line. Two example

patterns (shown in black) produce an "up" output, and two example patterns (shown in white) produce a "down" output.

(D) Illustration of movements composed of outputs in different temporal orders. Issuing a particular output, for example the output corresponding to zero output (orange dot), in different movements requires activity for the output to take different transitions to issue different next outputs. This is highlighted by activity issuing the zero output taking the blue transition in movement 1 and the green transition in movement 2.

(E) Illustration of the Invariant Dynamics view for activity producing movements from (D). Under this view, different activity patterns are used to issue the same output in different movements. The patterns transition to different future patterns according to invariant dynamics h (gray arrows), following trajectories that issue the outputs in the appropriate temporal order.

(F) Illustration of the Invariant Pattern view for activity producing movements from (D). Under this view, an output is issued by an invariant distribution of activity, regardless of the movement. The invariant pattern transitions flexibly to different future patterns. The gray curve shows the (arbitrary) encoding f defining the average invariant pattern for each output.

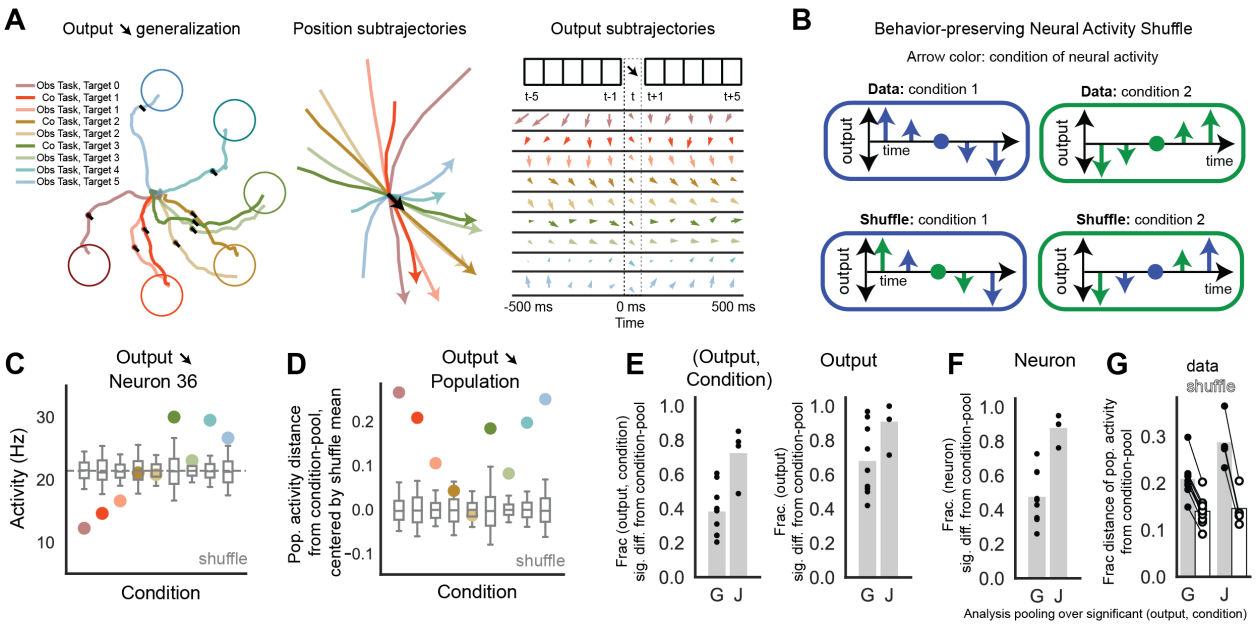

Figure 2. The same output is issued by different activity patterns in different movements.

(A) *Left:* Illustration of the occurrence of an example output (smallest magnitude bin, -45 degree direction) during single trials for a variety of conditions. *Center:* For the example output, each condition's mean position subtrajectory (average position trajectory from -500ms to 500ms locked to output occurrence for a condition). Subtrajectories are centered and superimposed at output occurrence to highlight trajectory differences. The output's average usage is different for different conditions. *Right:* For the example output, each condition's mean output subtrajectory (average output trajectory from -500ms to 500ms locked to output occurrence for a condition).

(B) Illustration showing a shuffling procedure used for testing if variation in neural activity for an output across conditions is significant. *Top row*: Outputs (arrows) producing 2 movements, each from a distinct condition (blue, green). *Bottom row*: In the shuffling procedure, the neural activity corresponding to a particular output and condition is shuffled to an occurrence of the same output in a random condition.

(C) Mean activity of an example neuron for the same output and conditions illustrated in (A) is shown. Dotted gray line shows the average condition-pooled output firing rate and gray boxplots show the distribution of expected condition-specific activity computed from the shuffle ("condition-pooled activity"). Condition-specific output activity is significantly different from condition-pooled output activity aggregating over all neurons, outputs, and conditions: Monkey G [J]: pv < 0.001 for 9/9 [4/4] sessions, pv < 0.001 pooled over sessions (mean difference between condition-specific and condition-pooled output activity = 1.108 Hz [1.223 Hz], mean (95^{th} percentile) of differences for shuffled distribution = 0.946 (0.951) Hz [0.722 (0.734) Hz].

(D) For the population's activity vector, the distance from average condition-specific output activity to the average condition-pooled output activity pattern for the same output and conditions illustrated in (C) is shown. The distribution of distances computed from the shuffle is shown with the gray boxplot. For each condition, the data and the boxplot are centered by the mean of the shuffle distribution of differences. Condition-specific output activity is significantly different from condition-pooled output activity aggregating over all outputs and conditions: Monkey G [J]: pv < 0.001 for 9/9 [4/4] sessions, pv < 0.001 for pooled over sessions (mean of condition-specific and condition-pooled population activity distances = 0.223 [0.392], mean (95^{th} percentile) of shuffled distribution output-condition population activity distances = 0.183 (0.184), [0.219 (0.224)].

(E) Fraction of output-conditions with population activity significantly different from condition-pooled activity (Monkey G [J]: n=9 [4] sessions) (*left*). Fraction of outputs with condition-specific activity significantly different from condition-pooled activity, aggregating over conditions (Monkey G [J]: n=9 [4] sessions) (*right).*

(F) Fraction of neurons with condition-specific output activity significantly different from condition-pooled output activity (Monkey G [J]: n=9 [4] sessions), aggregating over output-conditions that were individually significant when analyzing population activity.

(G) Normalized population activity distance between condition-specific output activity and condition-pooled output activity for data (dark bars) and median of shuffled distribution (light bars) (Monkey G [J]: n=9 [4] sessions). The normalization is to the magnitude of the condition-pooled output activity pattern.

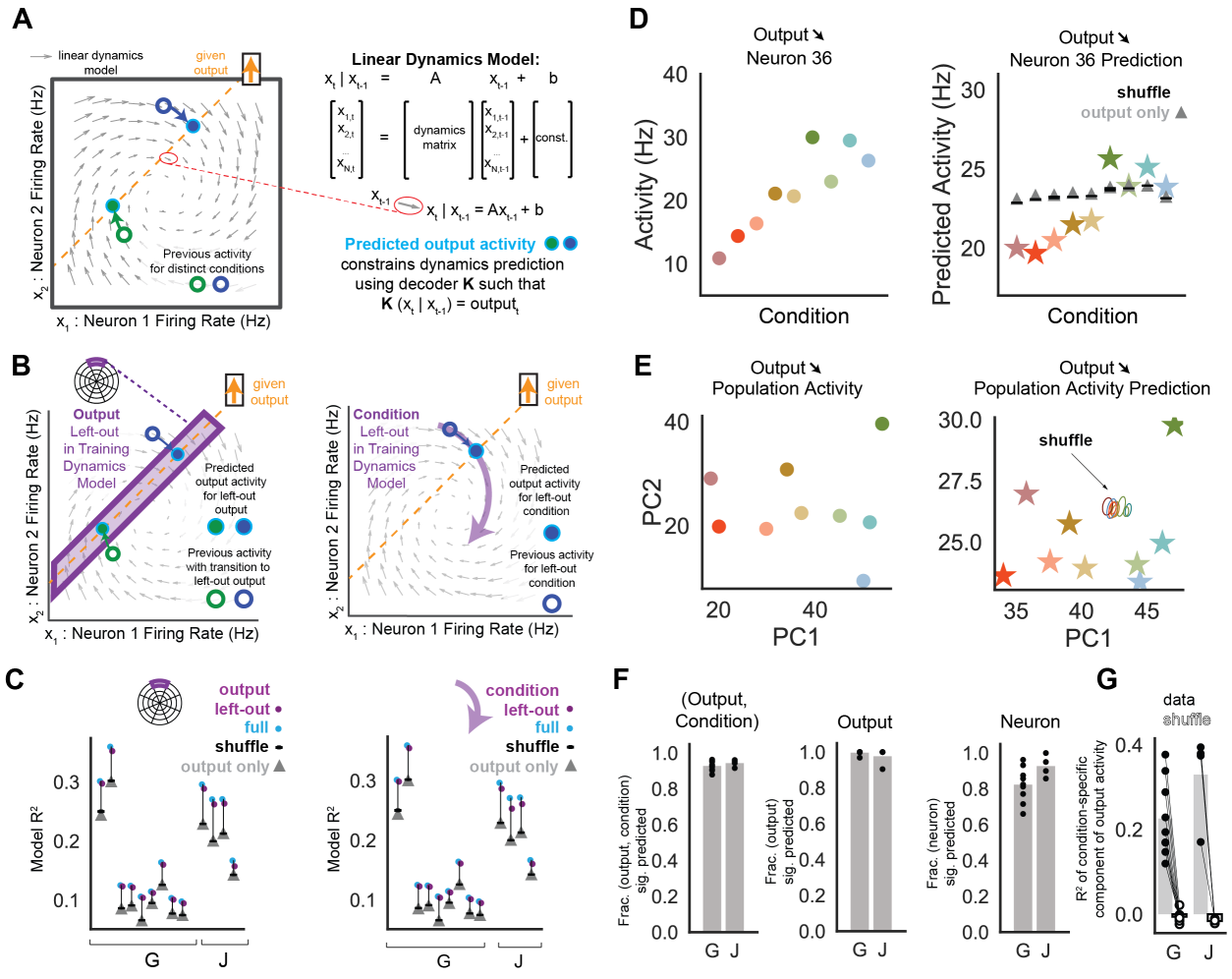

Figure 3. The different activity patterns used to issue the same output are predicted by a model of invariant dynamics.

(A) Schematic illustrating the question of whether invariant dynamics predict activity given its output and previous activity. (*Left)* The given output is the orange arrow, and all activity along the dotted orange line would produce the output. The filled circles with cyan outlines are the predicted output neural activity, and the empty circles are previous neural activity (color indicates the movement). The gray arrows are the predictions of activity transitions due to the model of dynamics rules. (*Right*). Linear model of invariant dynamics. Current neural activity (x_t) is modeled as a linear function of previous neural activity (x_{t-1}) and an offset (b). In order to

predict current neural activity that issues the known current output, we model current neural activity, current output, and previous neural activity as jointly Gaussian and compute the conditional distribution of ($x_t \mid x_{t-1}$, $output_t$) using knowledge of the BMI decoder K (Methods).

(B) Schematic illustrating the question of whether activity for a given output is still accurately predicted by invariant dynamics if transitions to and from that output or transitions within an entire condition are left out of the dynamics model training data. Purple indicates left out training data (*left:* an output, *right*: a condition).

(C) R^2 of the dynamics model trained using a complete dataset and predicting held-out data (cyan), dynamics model trained with an output left out and predicting the left-out output (*left, purple)* or dynamics model trained with a condition left out and predicting the left-out condition (*right, purple)* (Monkey G [J]: n=9 [4] sessions). All models are compared to predictions made by a model fit on shuffled datasets, "shuffle dynamics" (black distribution, spread is too narrow to see) and an estimate of x_t given $output_t$ (gray triangle). Complete dataset dynamics model (cyan), dynamics model with output left out (purple *left*), and dynamics model with condition left out (purple *right*) all predict activity significantly better than shuffled dynamics: Complete dataset (cyan): Monkey G [J]: $pv < 0.001$ for 9/9 [4/4] sessions, $pv < 0.001$ for sessions pooled, mean $R^2 = 0.167$ [0.252], mean (95^{th} percentile) R^2 of shuffle = 0.130 (0.130) [0.196 (0.196)]. Output left-out (purple *left*): Monkey G [J]: $pv < 0.001$ for 9/9 [4/4] sessions, $pv < 0.001$ for sessions pooled, mean $R^2 = 0.163$ [0.243], mean (95^{th} percentile) R^2 of shuffle = 0.130 (0.130) [0.196 (0.196)]. Condition left-out (purple *right*): Monkey G [J]: $pv < 0.001$ for 9/9 [4/4] sessions, $pv < 0.001$ for sessions pooled, mean $R^2 = 0.163$ [0.240], mean (95^{th} percentile) R^2 of shuffle = 0.130 (0.130) [0.196 (0.196)].

(D) *Left*. Condition-specific output activity for neuron 36 from example session (same as Fig. 2C). *Right.* Condition-specific output activity predicted by dynamics (stars), shuffled dynamics (black distribution), and only conditioning on output (gray triangle). Overall, neurons were significantly predicted with the dynamics model. Monkey G [J]: pv < 0.001 for 9/9 [4/4] sessions, pv < 0.001 for pooled sessions (mean error between dynamics-predicted and true condition-specific output activity: 1.234 Hz [1.185 Hz], mean (5^{th} percentile) of error between shuffle dynamics-predicted and true condition-specific output activity: 1.352 (1.352 Hz) [1.443 (1.442)].

(E) *Left.* Condition-specific output population activity plotted in PC space for the same output and conditions as in Fig. 2A. *Right.* Dynamics-predicted condition-specific output population activity plotted in the same PC space (stars), and dynamics-predicted activity from shuffled dynamics (ellipses show 3 standard deviations of spread of predictions). Overall, output-condition population activity was significantly predicted with the dynamics model: Monkey G [J]: pv < 0.001 for 9/9 [4/4] sessions, pv < 0.001 for pooled sessions (mean error between dynamics-predicted and true condition-specific output activity: 0.244 [0.365], mean (5^{th} percentile) of error between shuffle dynamics-predicted and true condition-specific output activity = 0.276 (0.276), [0.451, (0.451)].

(F) *Left.* Fraction of output-conditions with population activity predicted significantly better with the dynamics model than shuffled dynamics (Monkey G [J]: n=9 [4] sessions). *Center.* Fraction of outputs, aggregated over conditions, with population activity predicted significantly better than shuffled dynamics (Monkey G [J]: n=9 [4] sessions). *Right.* Fraction of neurons, aggregated over all output-conditions, with activity predicted significantly better than shuffled dynamics (Monkey G [J]: n=9 [4] sessions).

(G) Variance of the condition-specific component of output activity (condition-specific output activity minus the activity predicted from output alone ($x_t \mid output_t$)) explained by the dynamics model was significantly greater than the shuffle-dynamics (Monkey G [J]: n=9 [4] sessions). Monkey G [J]: pv < 0.001 for 9/9 [4/4] sessions, pv < 0.001 for pooled session (mean R^2 for dynamics model predicted activity 0.211 [0.315], mean (95^{th} percentile) of R^2 of shuffled dynamics -0.006 (-0.005) [-0.014 (-0.013)].

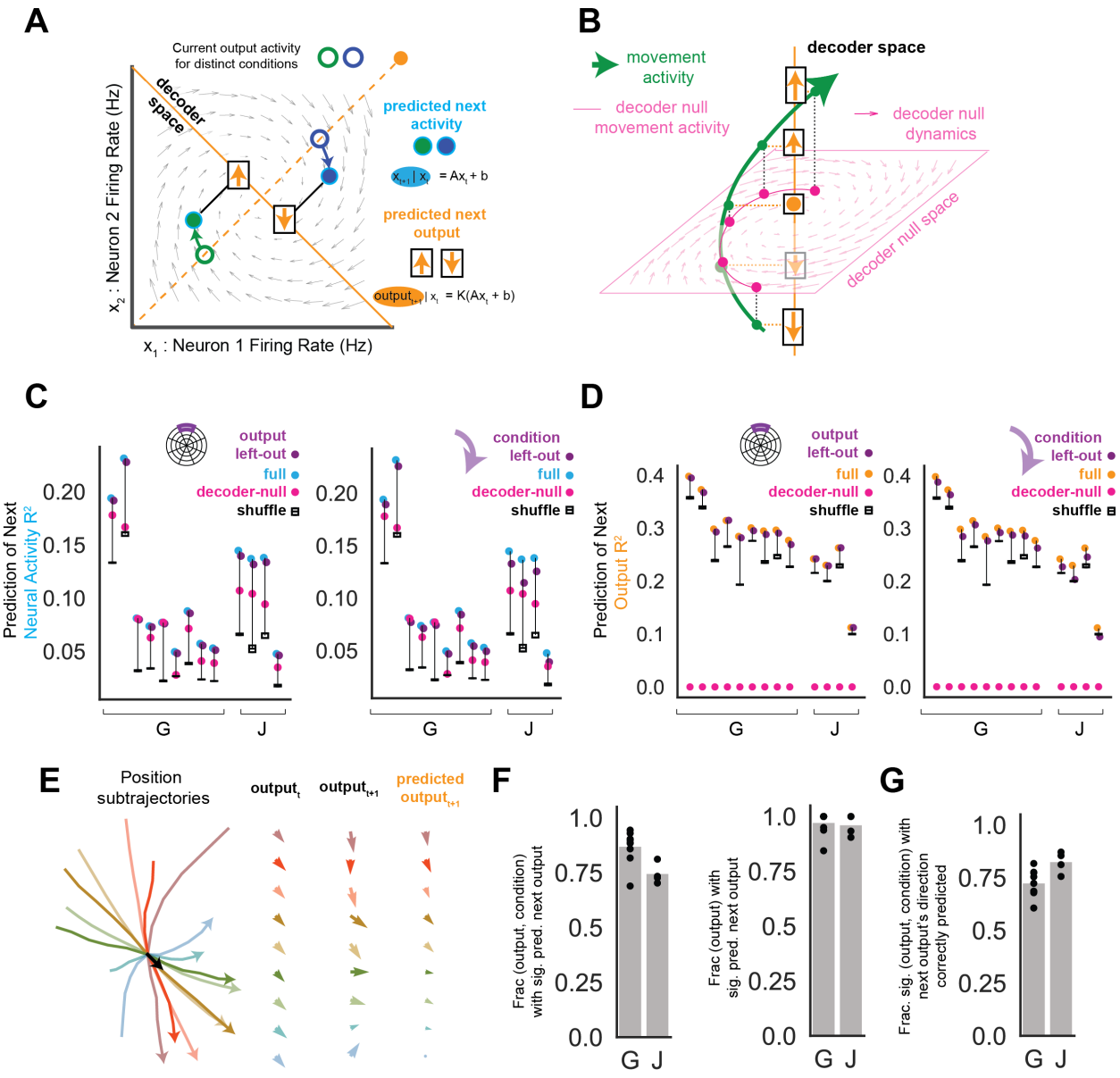

Figure 4. The transitions between patterns that drive output are predicted by a model of invariant dynamics

(A) Schematic illustrating the question of whether invariant dynamics are used to transition activity towards the next output needed for each movement. Instead of predicting activity issuing

a given output, now current activity is used to predict the future activity (cyan) and the future output (orange).

(B) Schematic illustrating the possibility that invariant dynamics may only define activity transitions in the neural space null to the BMI decoder. If this were the case, invariant dynamics may predict the next neural activity in the decoder null space, but the predicted transition would not change the next output. To highlight this possibility, invariant dynamics were fit on the component of activity null to the decoder space (pink).

(C) R^2 of predictions for future activity using the dynamics model trained using a complete dataset and predicting left-out test data (cyan), dynamics model trained with output left out and predicting left-out outputs (*left, purple)*, dynamics model trained with conditions left out and predicting left-out conditions (*right, purple)*, or null dynamics (pink) (Monkey G [J]: n=9 [4] sessions). All models are compared to the shuffle dynamics (black distribution, spread is too narrow to see). Complete dataset (cyan): Monkey G [J]: pv < 0.001 for 9/9 [4/4] sessions, pv < 0.001 for sessions pooled, mean R^2 = 0.100 [0.117], mean (95^{th} percentile) R^2 of shuffle = 0.055 (0.055), [0.051 (0.053)]. Output left-out (purple *left*): Monkey G [J]: pv < 0.001 for 9/9 [4/4] sessions, pv < 0.001 for sessions pooled, mean R^2 = 0.099 [0.113], mean (95^{th} percentile) R^2 of shuffle = 0.055 (0.055), [0.051 (0.053)]. Condition left-out (purple *left*): Monkey G [J]: Monkey G [J]: pv < 0.001 for 9/9 [4/4] sessions, pv < 0.001 for sessions pooled, mean R^2 = 0.097 [0.103], mean (95^{th} percentile) R^2 of shuffle = 0.055 (0.055), [0.051 (0.053)]. Null dynamics (pink): Monkey G [J]: Monkey G [J]: pv < 0.001 for 9/9 [4/4] sessions, pv < 0.001 for sessions pooled, mean R^2 = 0.083 [0.085], mean (95^{th} percentile) R^2 of shuffle = 0.055 (0.055), [0.051 (0.053)].

(D) R^2 of predictions for future outputs using the dynamics model trained using a complete dataset and predicting held-out data (orange), dynamics model trained with outputs left out and predicting left-out outputs (*left, purple)*, dynamics model trained with conditions left out and predicting left-out conditions (*right, purple)*, or null dynamics (pink) (Monkey G [J]: n=9 [4] sessions). All models are compared to the shuffle dynamics (black distribution, spread is too narrow to see). Complete dataset (orange): Monkey G [J]: pv < 0.001 for 9/9 [4/4] sessions, pv < 0.001 for sessions pooled, mean R^2 = 0.315 [0.212], mean (95^{th} percentile) R^2 of shuffle = 0.264 (0.266) [0.186 (0.188)]. Output left-out (purple *left*): Monkey G [J]: pv < 0.001 for 9/9 [4/4] sessions, pv < 0.001 for sessions pooled, mean R^2 = 0.310 [0.211], mean (95^{th} percentile) R^2 of shuffle = 0.264 (0.266) [0.186 (0.188)]. Condition left-out (purple *right*): Monkey G [J]: pv < 0.001 for 9/9 [2/4] sessions, pv < 0.05 for 9/9 [3/4] sessions, pv n.s. for 0/9 [1/4] sessions, pv < 0.001 for sessions pooled, mean R^2 = 0.305 [0.193], mean (95^{th} percentile) R^2 of shuffle = 0.264 (0.266) [0.186 (0.188)]. Null dynamics (pink): Monkey G [J]: Monkey G [J]: pv n.s. for 9/9 [4/4] sessions, pv n.s. for sessions pooled, mean R^2 = 0.00 [0.00], mean (95^{th} percentile) R^2 of shuffle = 0.264 (0.266) [0.186 (0.188)].

(E) *Left.* Position subtrajectories for example output. *Right.* Example output (column 1), condition-specific next output (column 2), and dynamics-predicted condition-specific next output (column 3). Rows correspond to different conditions. Overall, condition-specific next outputs are significantly predicted: Monkey G: pv < 0.001 for 9/9 [4/4] sessions, pv < 0.001 for pooled sessions (mean predicted output error = 3.956 [7.324], mean (5^{th} percentile) shuffle dynamics predicted output error = 5.40 (5.38), [9.305 (9.240)]).

(F) *Left.* Fraction of output-conditions where next output is predicted significantly better than shuffle (Monkey G [J]: n=9 [4] sessions). *Right*. Fraction of outputs with significantly predicted next outputs, aggregating over conditions (Monkey G [J]: n=9 [4] sessions).

(G) Fraction of significant condition-specific outputs with next output direction (clockwise or counterclockwise) correctly predicted (Monkey G [J]: n=9 [4] sessions).

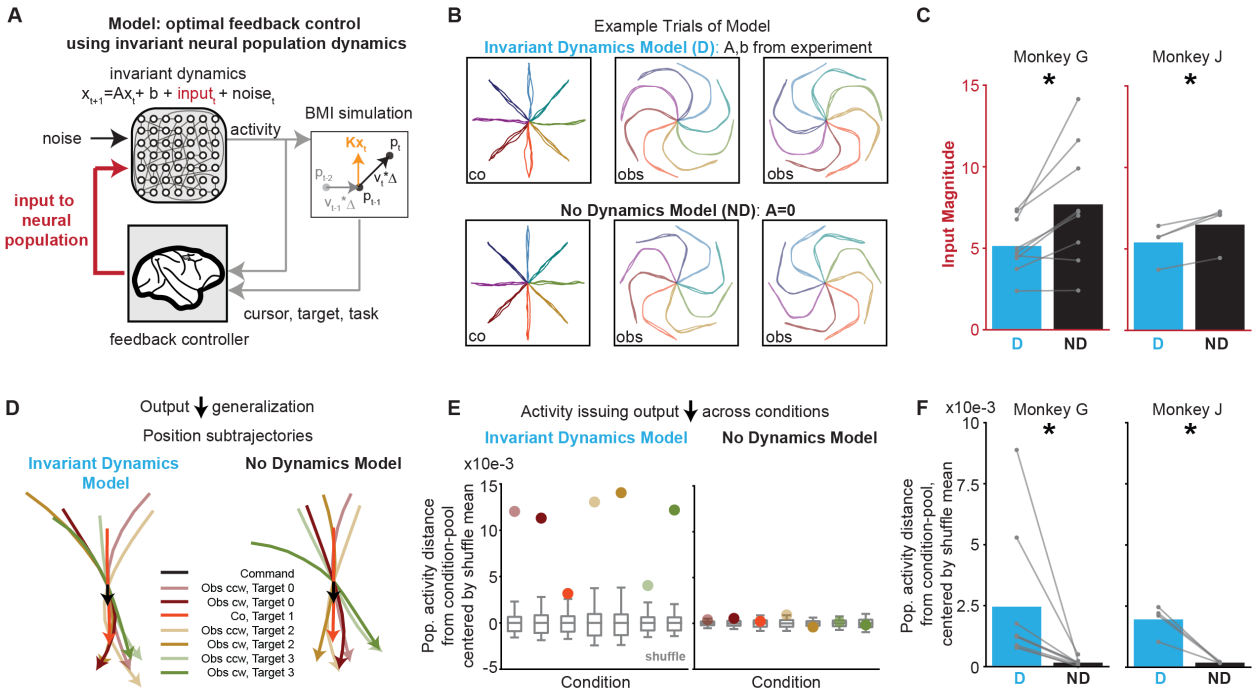

Figure 5: Optimal control theory reveals that a neural population needs less input to control movement if it uses observed invariant dynamics.

(A) Optimal feedback control using invariant dynamics. Neural population activity was driven by a combination of invariant dynamics observed experimentally, input from an optimal linear feedback controller, and noise. The resultant activity issued appropriate outputs to control movement for center-out and obstacle-avoidance conditions.

(B) Example trials (n=3 randomly chosen from 20 trials simulated per condition) of 24 conditions generated by the model. *Top:* model with invariant dynamics (Invariant Dynamics Model). *Bottom:* model without dynamics (No Dynamics Model), such that the dynamics matrix A was set to 0.

(C) The Invariant Dynamics Model required significantly less input in order to complete the task (Monkey G [J]: n=9 [4] sessions). (Individual sessions were tested with a paired Wilcoxon test

across the two models comparing trial-averaged input magnitude for each of 24 conditions. Monkey G: pv<0.001 for 8/9 sessions, n.s. for 1/9 sessions. Monkey J: pv<0.001 for 4/4 sessions. Data was pooled over sessions using LME model between input magnitude and model category with session modeled as random effect. Monkey G: N=432, z=9.47, pv=2.83e-21. Monkey J: N=192, z=5.64, pv=1.70e-8.).

(D) Position subtrajectories from the Invariant Dynamics Model and No Dynamics model for an example output used in a variety of conditions.

(E) Distance from condition-specific output population activity to condition-pooled output population activity for the example output in (D), for the Invariant Dynamics Model and No Dynamics Model. Gray boxplots show the behavior-preserving shuffled distribution. Distances are centered by the mean of the behavior-preserving shuffled distribution.

(F) Distances from condition-specific output population activity to the condition-pooled output population activity are significantly larger in the Invariant Dynamics Model versus No Dynamics Model (Monkey G [J]: n=9 [4] sessions). Each dot in bar plot averages over distances for all outputs and conditions with >= 15 observations within a session. (Individual sessions were tested with a 2-sample KS test comparing activity distances across the two models. Monkey G: pv<0.001 for 9/9 sessions. Monkey J: pv<0.001 for 4/4 sessions. Data was pooled over sessions using LME between distance and model category with session modeled as a random effect. Monkey G: N =4795, z=-34.22, pv=1.14e-256. Monkey J: N=2256, z=-24.046, pv=9.16e-128).