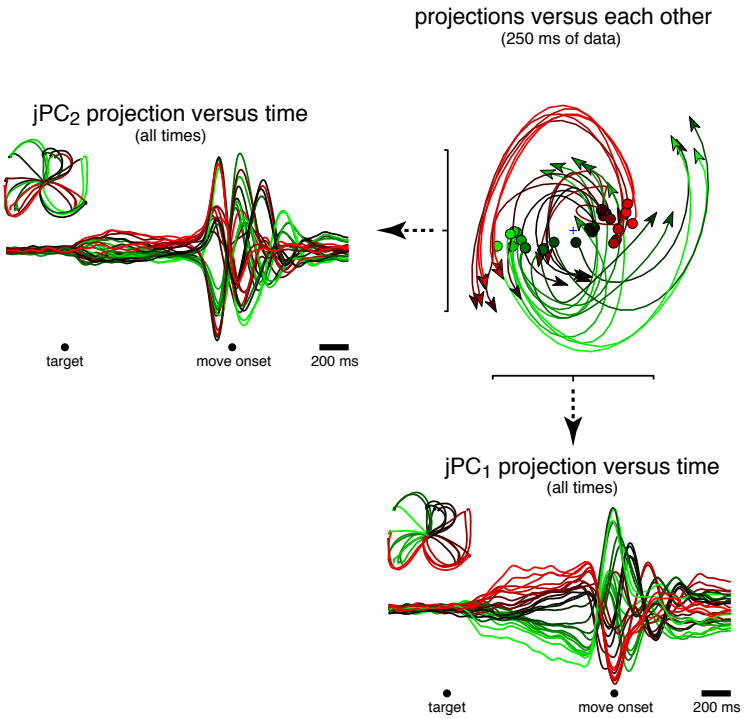
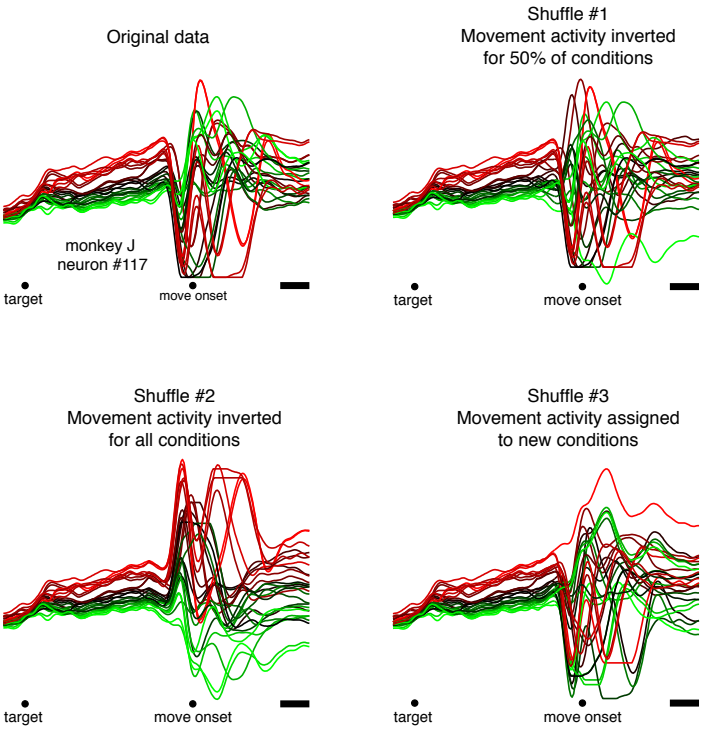
doi:10.1038/nature11129




***Supplementary figure 1.*** Projection onto each jPCA axis as a function of time. Data is shown for monkey J3. The plot of the jPCA plane uses the same format as in figure 3 of the main text. The plots versus time use the same format as for the individual-neuron PSTHs in figure 2 of the main text (though here the vertical units are arbitrary). Traces are colored red to green based on the level of preparatory activity for that projection. This allows visualization of 'tuning' with respect to the reach trajectories (inset). Direction 'tuning' is present but imperfect in the projections, much as it is for the neurons upon which the projections are based.

Indeed, in many ways the projections versus time look as if they could be the responses of single neurons. This is not accidental: the jPCA projections capture responses that are strongly present in the responses of individual neurons. Conversely, the jPCA projections are simply weighted sums of individual-neuron responses. Each jPCA projection can thus be interpreted much as with a traditional 'population average'. The key difference is that the weights used for jPCA are chosen by the algorithm, rather than set by hand according to the 'preferred direction'.

As with the supplementary movies and other plots that show an extended period of time, these projections were based on the top ten PCs rather than the top six. This is not critical but aids in finding a projection that works well across a wide range of times.




***Supplementary figure 2.*** Construction of shuffle controls, illustrated for an example neuron. A potential concern is that the jPCA method might be powerful enough to find state-space rotations for *any* population response that contains diverse and multiphasic responses. This would be a large concern were one analyzing a few conditions in a very high-dimensional space. However, our analyses involved 27-108 conditions, and were applied only after the data dimensionality was reduced using traditional PCA. Nevertheless, it is prudent to empirically evaluate the degree to which rotational patterns might be found by chance simply because responses are diverse and complex.

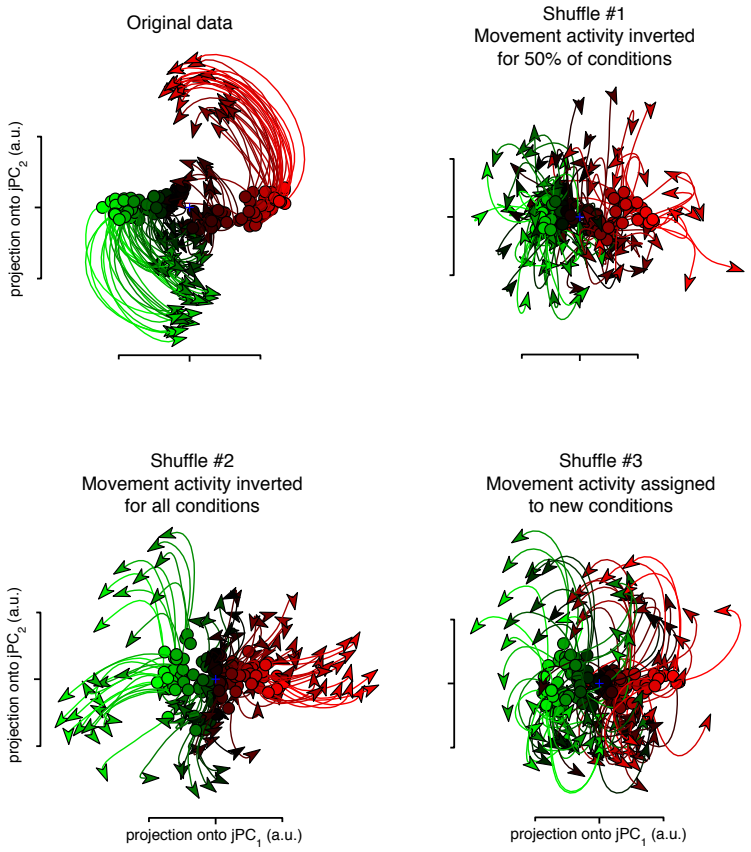
To evaluate this possibility we performed three shuffle controls that preserve response diversity and complexity, but disrupt the deep structure of the data. If robust rotational structure can be seen in the shuffled controls, then this would be a cause for serious concern. It would indicate that rotations can be found by our methods even when not truly present.

All shuffle controls are based on the distinction between preparatory activity (which is left intact) and peri-movement activity (which was shuffled in three different ways). We picked a time-point 50 ms after the go cue as the dividing point between preparatory and peri-movement activity. For the first shuffled control, the pattern of peri-movement activity was inverted for half of the conditions, selected at random. The inversion was performed around the dividing time-point, such that continuity with preparatory activity was preserved. This procedure was performed separately for each neuron.

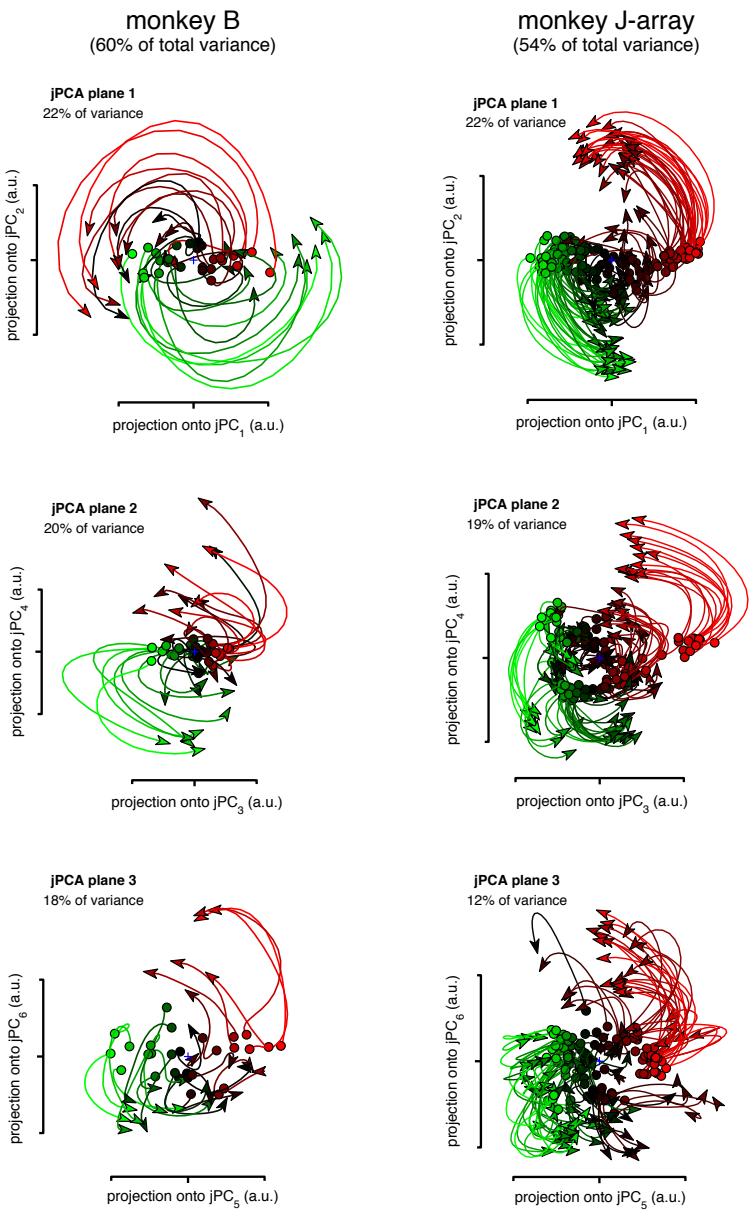
The second shuffle control was similar to the first, but inverted the peri-movement activity pattern for all conditions. This manipulation is not expected to remove all rotational structure (most such structure is merely sign-inverted). However, this manipulation is expected to largely remove any consistent relationship (assuming there is one) between the preparatory state and the phase of subsequent oscillations. Thus, for the time period of interest, this shuffle control is expected to greatly reduce the consistency of any true rotations, especially the relationship between phase and initial state.

The third shuffle control randomly reassigned the peri-movement activity from one condition to the preparatory activity from another. The beginning of the peri-movement pattern was simply appended to the final firing rate during the preparatory state, such that there was no discontinuity The same reassignment was performed for all neurons. As with the second control, this third shuffle control is not expected to remove all rotational structure (many of the peri-movement activity patterns are altered only modestly by this manipulation). However, any true relationship between rotation phase and initial state is expected to be disrupted. Thus, for the time period of interest, this shuffle control is expected to greatly reduce the consistency of any true rotations, especially the relationship between phase and initial state.

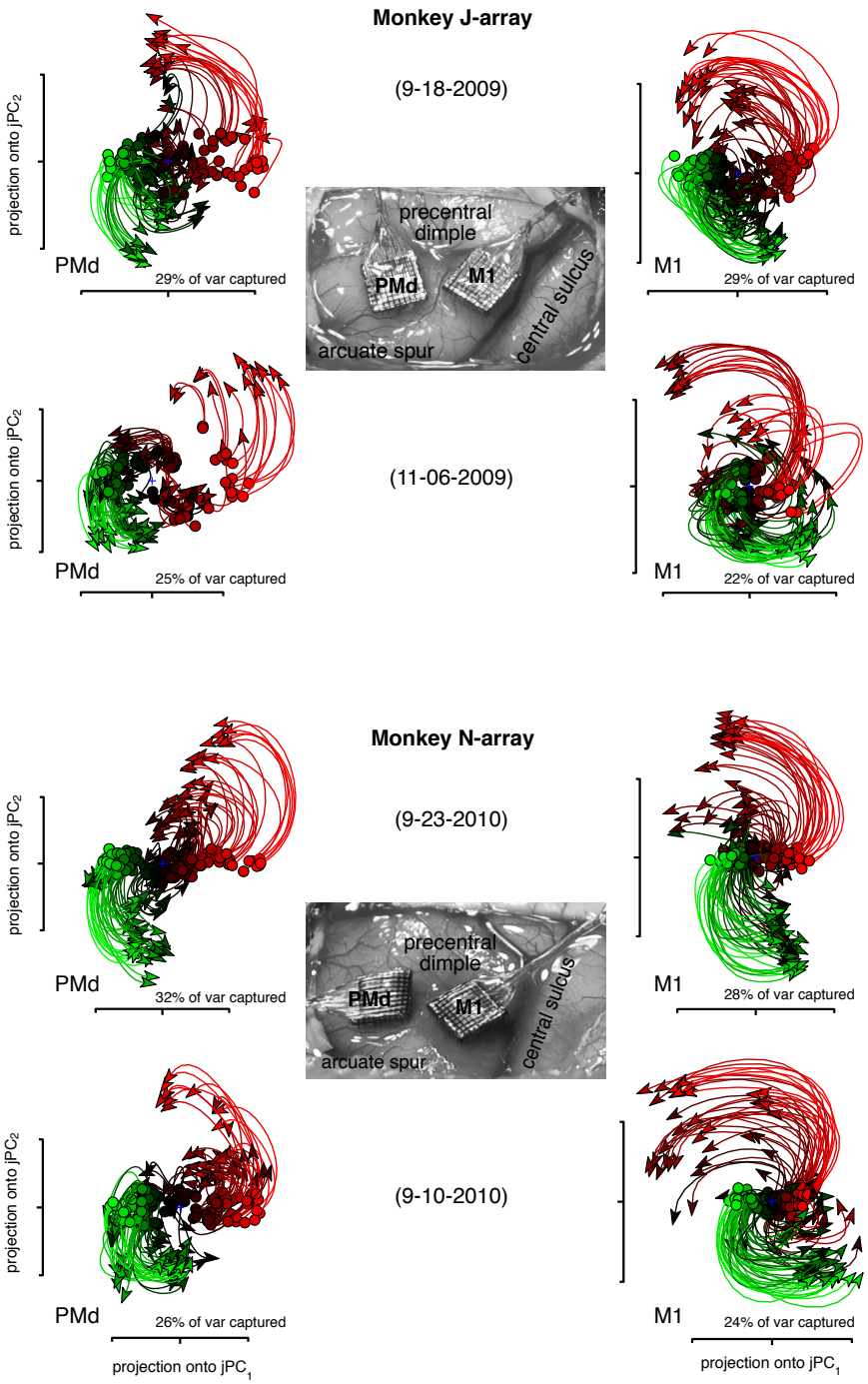
For each control, if strong rotational structure survives the shuffle procedure, then that will be taken as evidence that the jPCA method can erroneously extract such structure even when it is absent or weakly present. This would make the central results in figure 3 impossible to interpret. If strong rotational structure is lost following the shuffle, then that will be taken as evidence that such structure was in fact present in the original data to a much greater degree than expected by chance.




***Supplementary figure 3.*** Effect of the shuffle controls illustrated in supplementary figure 2. Each panel plots the jPCA projection of the population response for the J-array dataset (108 reach conditions). The top-left panel plots the projection for the original, un-shuffled data (same as figure 3*e* of the main text). The other panels plot the projections when jPCA was applied following the three shuffle controls. While many individual trajectories remain curved, the overall robustness of the rotational structure, and the relationship between phase and initial state, is largely lost following shuffling. This indicates that the pattern seen in the top-left panel reflects real underlying structure in the population response, rather that the ability of our method to find such structure by chance. Shuffle controls had similar effects across all datasets.



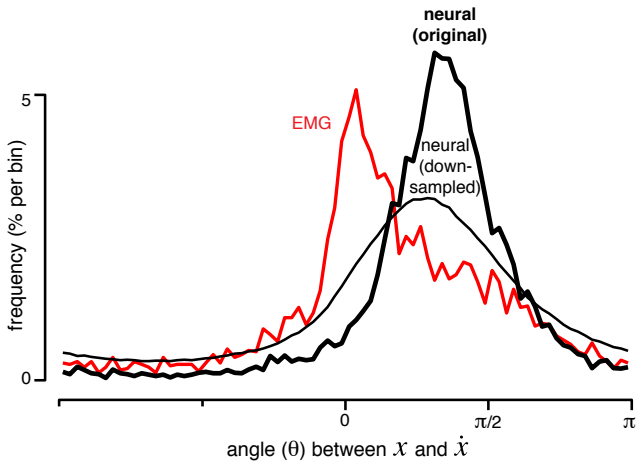

***Supplementary figure 4.*** The data contain multiple planes with rotations. Data are shown for two datasets (monkey B and monkey J-array). Very similar findings were obtained for all datasets. Each column plots the first three jPCA planes (the top six jPCs) found within the top 10 PCs. All three planes contained rotational structure that was coherent (in the same direction and at a similar angular velocity) across conditions. However, rotations were slower (as expected) and less orderly for the higher-numbered jPCA planes. The top three jPCA planes (spanning six dimensions) captured 60% (monkey B) and 54% (monkey J-array) of the total variance in the data. For comparison, the top six PCs (which by definition capture the most variance possible) captured 72% and 64% of the variance. These findings were typical: all datasets contained two, and usually three, planes that captured rotations. Together these planes captured 50-70% of the data variance (range is across datasets). The planes shown were found by applying jPCA to the top 10 PCs, rather than the top 6 as in the main text. This is necessary; it is unlikely that the six dimensions with the strongest rotations would fortuitously align with the top 6 PCs.




***Supplementary figure 5.*** Analysis segregated by area (caudal PMd versus M1). Analysis employed data from two monkeys, each implanted with a pair of electrode arrays. Each array spanned ~4-6 mm of anterior-posterior distance, with ~1-2 mm of separation at the closest point. We refer to the two arrays as the 'PMd array' and the 'M1 array', though it should be kept in mind that border between these two areas is not sharp. Shown are the J-9-18-2009 and N-9-23-2010 datasets employed in the main text, plus two additional datasets collected on different dates: J-11-06-2009 and N-9-10-2010. The additional datasets employed the same design as those shown in the main text, and were added to give a sense of the consistency of the effects described below.

Both PMd and M1 exhibited clear rotations of the neural state. However, there were two subtle but noticeable differences between the projections for the PMd and M1 arrays. First, for PMd the preparatory state tended to be better separated across conditions. This is consistent with the long-reported fact that PMd tends to exhibit stronger preparatory activity. Second, the angular velocity of the rotations was in each case slightly higher for M1. We suspect that this effect may be real. It is consistent with our informal observations, notable in every dataset we have inspected so far, that neurons recorded in posterior sites are more likely to exhibit high-frequency response features. Still, we wish to stress that PMd-like neurons are frequently found in M1 and *vice versa*. And of course both arrays exhibited overall patterns that were qualitatively very similar: the neural state rotated away from the initial preparatory state, with a direction and angular velocity that was reasonably consistent across all 108 reach conditions.

Variance captured refers to the proportion of total data variance that lies within the jPCA plane. This was typically 20-35%. Capturing more than this using two dimensions is unlikely, given the high-dimensional nature of the neural data.

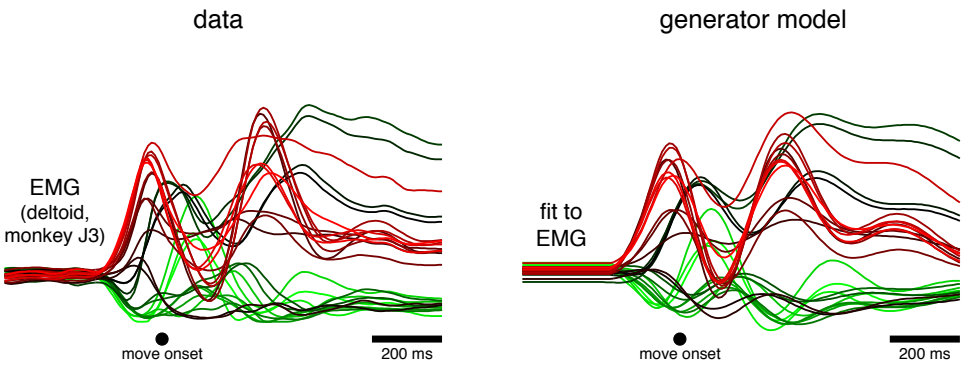



***Supplementary figure 6.*** Effect of down-sampling the population of recorded neurons to the same size as the population of recorded muscles. As reported in the main text (fig. 4*f,j* and fig. 6*a,b*) the recorded populations of muscles did not exhibit strong rotations in the jPCA plane. A potential concern is that the population of muscle recordings might have shown weak rotations (relative to the neural population) simply because fewer muscles were recorded than neurons. Across datasets, 6-12 muscles were recorded, compared with 50-218 neural isolations.

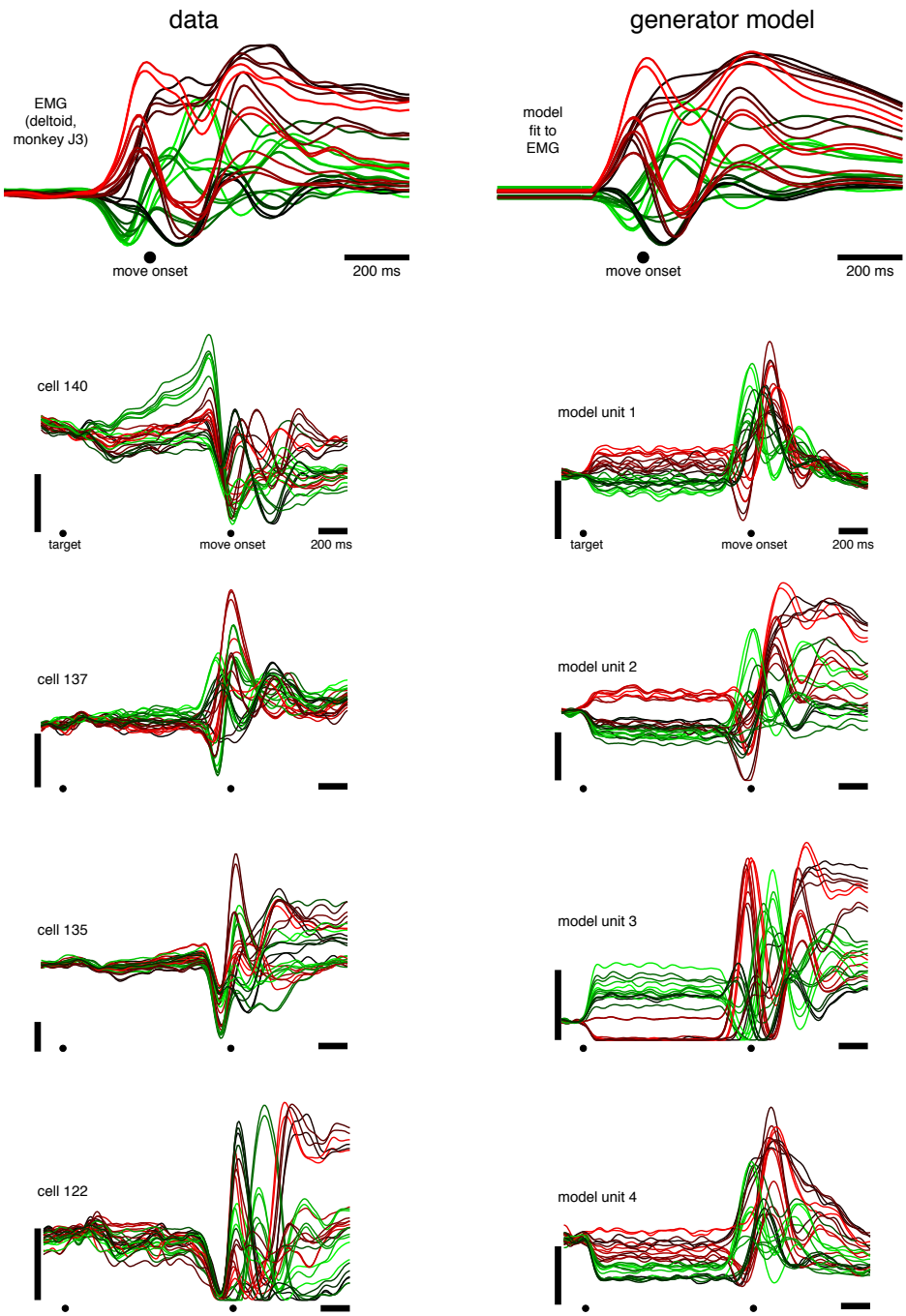
Before describing the control shown here, two points are worth making. First, for all analyses in the main text, the dimensionality of both the muscle and neural populations was reduced to six before applying jPCA. Thus, for both muscles and neurons, the jPCA method attempts to find a plane with rotations within a six-dimensional space. Specifically, it was not the case that jPCA was allowed to 'search' within a larger space for the neural data. Second, the central hypothesis of this study holds that the output of cortex reflects a subset of the patterns present in the neural data. This is consistent with the finding that muscle activity is typically found to be low dimensional[43,44]. Thus, it *should* in some sense be true that the muscle population shows weaker rotations because it is lower dimensional. This is certainly true of the generator model, where the single output is constructed from underlying rotations, but does not itself contain any rotations (main text; fig. 5). Nevertheless, it is important to rule out the statistical concern that the higher-dimensional neural space is somehow given an 'unfair' advantage.

To control for this possibility, for each dataset we down-sampled the neural population to equal the size of the muscle population. This was repeated 100 times for each dataset. Individual-neuron recordings are often much noisier than individual EMG recordings; we therefore restricted analysis to neurons where the strength of peri-movement activity was better than average. The down-sampled data was then analyzed as in fig. 6 of the main text. The distribution for the down-sampled data is broader and shifted slightly to the left of the distribution for the original data. This is expected: the increase in sampling error will broaden the distribution. Furthermore, rotations will be weakened on those draws where the sample of neurons does not contain both relevant phases to roughly equal degrees. Nevertheless, the distribution for the down-sampled neural data peaks well to the right of the distribution for the muscle recordings.

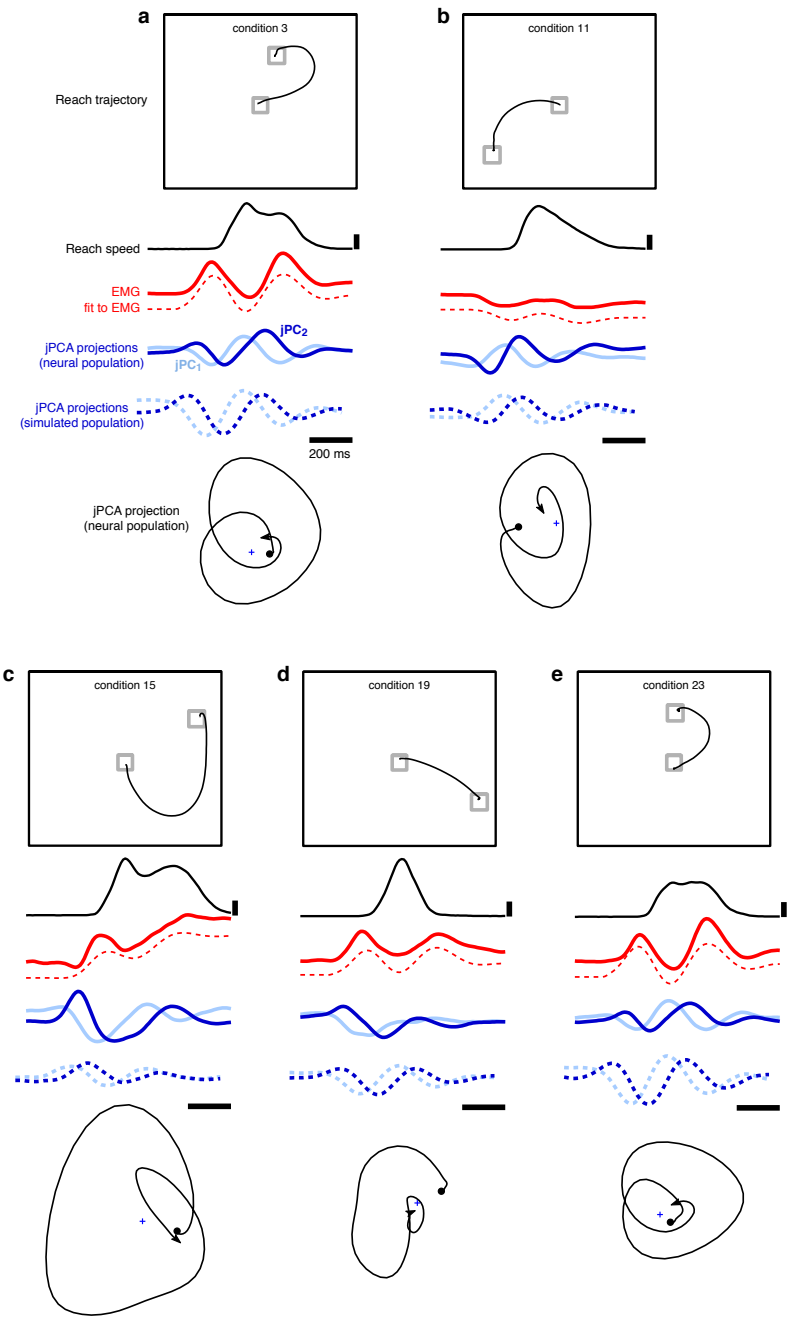
In summary, populations of neurons contain considerably stronger rotations of their state than do populations of muscles, even when the two populations are matched in size.




***Supplementary figure 7.*** Fits of the generator model (right) to deltoid EMG (left). Fits are the same as those in figure 5*b,c* of the main text, but are shown for all 27 conditions (dataset J3). Color-coding is based on the initial strength of deltoid EMG, just before movement onset.




***Supplementary figure 8.*** Further examples (from dataset J2) of how the generator model fits EMG (*top*). Also shown are example responses of both real (*left column*) and simulated units from the generator model (*right column*). Note that although the real and simulated data share a number of interesting general features, the individual units/neurons are not meant to (and do not) map directly onto one another. Vertical scale bars indicate 20 spikes/s. So that the same color coding can be used for all panels, color-coding is based on the initial strength of deltoid EMG, just before movement onset. Note that for the model, neural responses directly reflect the underlying rotational patterns used to generate EMG. Nevertheless the responses of most model neurons do not match the EMG profile.

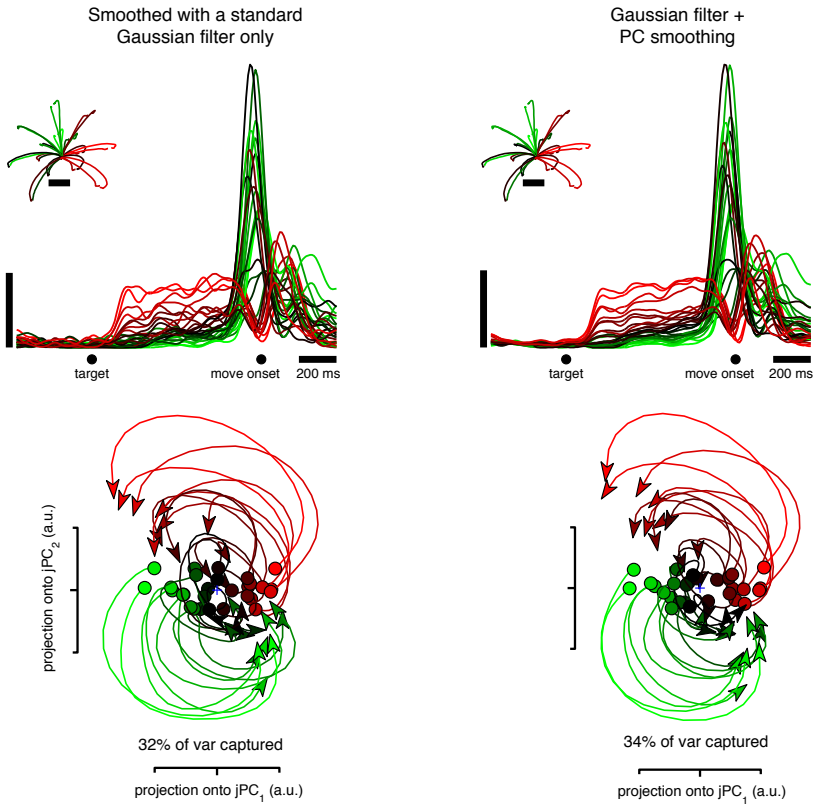



***Supplementary figure 9.*** Example data for individual conditions, allowing one to compare kinematics, EMG, jPCA projections of neural data, and jPCA projections of generator model 'data' (monkey J3 dataset). The goal is not to make direct quantitative comparisons, but to inform intuition regarding how oscillatory patterns are expected to change across conditions. Data are from monkey J3. Each panel (*a-e*) plots data for one condition. Shown are hand trajectory (*black, top*), reach speed (*black*), deltoid EMG (*red*), EMG fit produced by the generator model (*dashed red*), projections onto the jPCs versus time for real (*blue*) and generator model data (*dashed blue*), and the jPCA plane for the neural data (*black, bottom; jPC1 and jPC2 are the horizontal and vertical axes*). The jPCA projections for the generator model have been advanced in time by 50 ms to emulate a lag between neural and EMG signals.

A comparison of kinematics, EMG, and neural data makes two points. First, there is no straightforward relationship between reach duration and oscillation duration. For example, compare panel *d* with panel *a*. Although reach durations differ, the multiphasic muscle and neural patterns have a similar duration in both panels. Second, there is no obvious relationship between reach duration and oscillation frequency. These two observations were true by design for the generator model, but were not guaranteed to be true for the neural data or for the EMG.

Comparing EMG with the generator model illustrates why the generator model is able to provide such good fits. The EMG contains multiphasic features that differ, across conditions, in their amplitude and phase. Yet those multiphasic features are reasonably consistent in their frequency. The generator model is thus able to contribute those features using a short-lived oscillatory pattern whose phase and amplitude differs across conditions. The lower-frequency features are contributed by the second (and slower) of the two oscillations provided by the model (not shown).

These data are inadequate to address the possibility that neural activity directly drives muscle activity. Indeed, one presumes that reasonably strong dynamics are contributed by the spinal cord. Yet the examples shown illustrate that there is nothing paradoxical regarding some of the salient features of the neural state-space rotations. Such rotations have a relationship with kinematics that is sometimes counter-intuitive, but they form a natural basis for driving the muscle activity that results in those kinematics.




***Supplementary figure 10.*** Effect of the pre-processing step where the PSTH of each neuron was smoothed using the 'PC smoothing' method[45]. This step acts only to remove small amounts of noise from the PSTHs of individual neurons. PC smoothing is done at a single-neuron level and has no relationship to the use of PCA or jPCA at the population level.

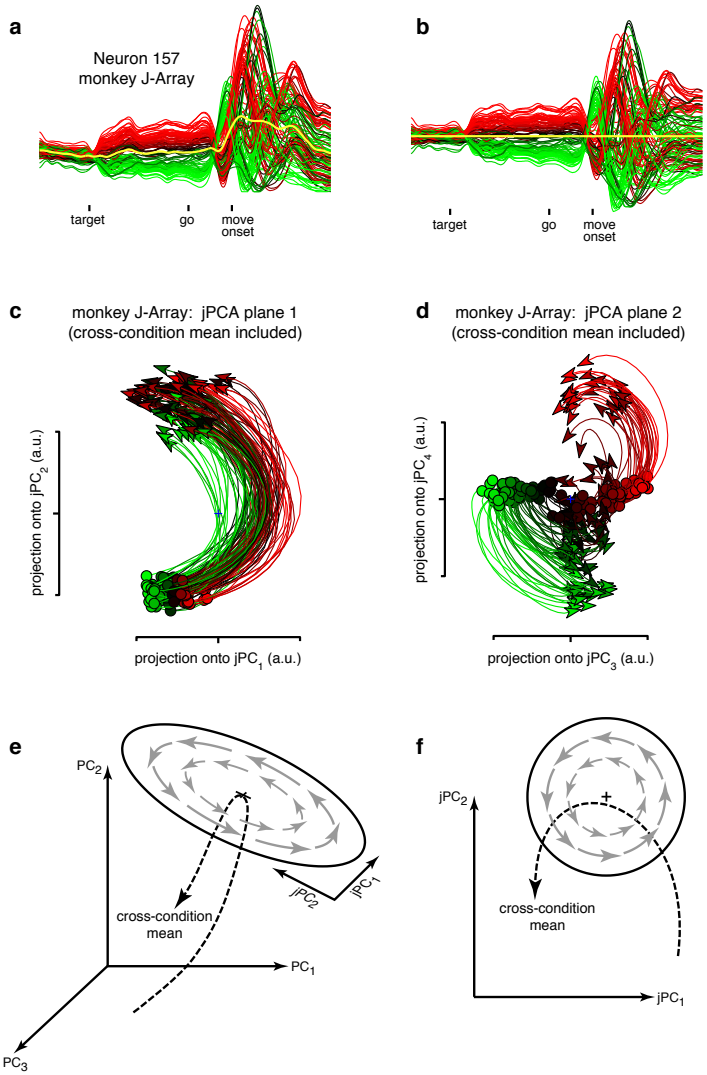
*Top panels*: response of one neuron (B16). The data in the left column were smoothed using a 20 ms Gaussian kernel in the traditional fashion. The data in the right column were further smoothed using the PC smoothing method.

*Bottom panels:* jPCA projections of the population for unsmoothed and smoothed data. The right panel is identical to that in fig. 3 of the main text. PC smoothing has almost no impact at the population level, for reasons described below.

PC smoothing was used to pre-process all data presented in the main text. The details of this method are provided below. There are two key high-level points. First, PC smoothing reduces the noise present in the PSTH of a given neuron. This provides a better visual estimate of that neuron's response. Second, PC smoothing has essentially no impact on the jPCA projections. The removal of small amounts of individual-neuron noise is largely irrelevant at the population level. The jPCA projections are thus virtually identical regardless of whether smoothing was applied. This was true of all datasets, including the EMG recordings.

Although it PC smoothing had essentially no impact on the central results, for completeness we describe below the motivation and methodology. First note that a temporal filter exploits the fact that different times cannot be arbitrarily different from one another. In contrast, PC smoothing exploits the fact that different conditions cannot be arbitrarily different from one another. PC smoothing exploits this fact by using PCA at the level of an *individual neuron* (note that this is very different from the more usual approach of using PCA at the population level). To apply PC smoothing, we compiled a *cxt* data matrix, where each row contained the response of that neuron for one condition across all times. We then decomposed this matrix into its principal components (PCs), and reconstructed the data using the first six principal components. This procedure preferentially discards small high-frequency events that are unique to one of the 27 conditions (and thus likely to result from sampling error). Something similar could of course be obtained by using a broader temporal filter, but that could come at the cost of losing real high-frequency aspects of the response. An advantage of PC smoothing is that it does not remove high-frequency aspects of the response if they are shared among a number of conditions.

To reiterate a technical point: this use of PCA to smooth individual-neuron PSTHs contrasts with the more typical use of PCA at the population level. In the current study we use PCA twice: once for PC smoothing as described above (a minor component of the overall analysis) and a second time at the population level during the computation of the jPCA plane (a key component of our analyses).




***Supplementary figure 11.*** Effects of the pre-processing step where each neuron's response was centered by removing the cross-condition mean. The goal of this pre-processing step was to focus all further analyses on dimensions where activity differed strongly across conditions. The panels below show how the cross-condition mean was removed (*top row*), the consequences of not subtracting the cross-condition mean (*middle row*), and possible relationships between the cross-condition mean and the rotational patterns (*bottom row*). ***a.*** Firing rate as a function of time for one example neuron (monkey J-array dataset; 108 conditions). Each green/red trace plots the average firing rate for one condition, shaded based on the level of preparatory activity. The yellow trace plots the cross-condition mean (the mean of all the other traces). Before subsequent analysis (the application of PCA and jPCA) this cross-condition mean was subtracted. This was done independently for each neuron. ***b.*** Response of the same example neuron after mean subtraction. The cross-condition mean (yellow) is now zero at all times. ***c.*** Application of jPCA to the monkey J-array dataset, with *no subtraction of the cross-condition mean* (*i.e.*, the pre-processing step in *a,b* was *not* applied). This panel can be contrasted with that in figure 3*d* of the main text (for which the cross-condition mean *was* subtracted). When the cross-condition mean is not subtracted, the projection onto the first jPCA plane captures response patterns that vary only weakly across conditions. This is not surprising: many neurons display strong response features that are similar across conditions (the well-known 'non-directional' component of the response[46]). Thus, the top PCs (via standard PCA) often contain large features that are very similar across conditions. The jPCs are simply a projection of what is already contained in the PCs. It was therefore common (as in this example) for the first jPCA plane to contain patterns that were largely condition-independent. Such patterns are consistent with our general hypothesis (they still involve rotations) but are difficult to interpret because there are many potentially trivial explanations for the observation of curved trajectories that are similar across all conditions. ***d.*** The second jPCA plane ($jPC_4$ versus $jPC_3$) for the same analysis as in *c* (again, the cross-condition mean was *not* removed). This pattern is similar to that seen in the original analysis in figure 3*d* (and exists in a plane orthogonal to that in panel *c* of this figure). Thus, for this dataset, the first jPCA plane largely captures the condition-independent response, and the second plane largely captures the condition-dependent response, and is thus very similar to that in the original analysis. However, across the 8 datasets it was not uncommon for both planes to capture a mixture of condition-independent and condition-dependent components. Thus, to allow the first jPC plane to always be the primary focus of analysis, and to allow that plane to capture structure across conditions, all jPCA analyses in the main text were performed after subtracting the cross-condition mean. ***e, f.*** Schematic illustration of possible trajectories for the condition-independent aspect of the response (the cross-condition mean) relative to the pattern of rotations. Those two aspects might be orthogonal (as in *e*) or might lie within the same plane (as in *f*). In both cases, rotations will be best isolated if the cross-condition mean is first removed. These schematics also illustrate a possible role for the condition-independent component of the response.

(As a technical point, the analyses in this figure consider the top 10 PCs, to allow sufficient dimensions to capture both condition-independent and condition-dependent features of the response).

## ***Supplementary Derivation***

jPCA is a dynamical variant of PCA that finds planes of significant rotational structure within data. We consider high dimensional time series data $\mathbf{x}(t) = [x_1(t), ..., x_n(t)]$. We can represent this data as a matrix $X \in \mathbb{R}^{ct \times n}$, where $ct$ is the number of time points in the time series across all conditions, and $n$ is the dimensionality of the data at any time point. This is as in the main text/methods, except here it is convenient to orient $X$ as $ct \times n$ rather than $n \times ct$. Also note that, in practice, jPCA was always applied *after* traditional PCA to $X_{red} \in \mathbb{R}^{ct \times k}$ for $k = 6$ or similar. For the purposes of understanding jPCA at a mathematical level, that preprocessing is a distraction (and has no bearing on the validity of the algorithm itself), so we ignore it here. Hence, hereafter we consider data of the form $X \in \mathbb{R}^{ct \times n}$.

Traditional PCA begins by calculating the data covariance $\Sigma = X^T X$ (assuming $X$ is mean-centered). However, since we are particularly interested in dynamical structure, we seek a different $n \times n$ matrix summarizing the data. The simplest dynamical system that we can fit to that data is a time-invariant linear dynamical system, which has the form $\dot{\mathbf{x}}(t) = \mathbf{x}(t)M$ for any matrix $M \in \mathbb{R}^{n \times n}$. Solving for such an $M$, given our data $X$, reduces to a simple least squares problem. We can write $\dot{X} = XM$, and then the least squares solution solves the problem $M^* = \text{argmin}_{M \in \mathbb{R}^{n \times n}} ||\dot{X} - XM||_F$ (where the subscript $F$ denotes the Frobenius norm). This is solved in simple closed form as $M^* = (X^T X)^{-1} X^T \dot{X}$ (often written as $M^* = X \backslash \dot{X}$). Importantly, this dynamics matrix $M^*$ is a valid summary matrix: whereas the data covariance $\Sigma = X^T X$ describes the ellipsoid that best fits the data (without regard to temporal information), $M^*$ describes the linear dynamical system that best fits the data $X$.

General linear dynamical systems describe both expansions/contractions and rotations, but they make no distinction between these aspects of the data. In this work we seek to investigate the role that rotations play in the dynamics of motor cortical neurons, and hence we are specifically interested in *rotational* linear dynamical systems. Every linear transformation $M$ is some mixture of a symmetric matrix and a skew-symmetric matrix, which we write $M = M_{symm} + M_{skew}$. The symmetric part is, by analogy to functions, the "even" part of the matrix, and is defined as $M_{symm} = (M + M^T)/2$. Such matrices satisfy $M_{symm} = M_{symm}^T$. Accordingly, the skew-symmetric matrix (odd part of the matrix, also called anti-symmetric) is $M_{skew} = (M - M^T)/2$. Such matrices satisfy $M_{skew} = -M_{skew}^T$. Adding $M_{symm}$ and $M_{skew}$ returns the matrix $M$, so indeed these are a general description of any square matrix $M$. Additionally, the symmetric component $M_{symm}$ has purely real eigenvalues and thus describes only expansions and contractions of data. So too, the skew-symmetric component has purely imaginary eigenvalues (in complex conjugate pairs) and describes rotations in the data.

Since we are interested only in rotational linear dynamical systems, we must solve the original least squares problem ($\dot{X} = XM$) not over all matrices $M$ (that describe any linear dynamical system), but instead we should solve the original problem constrained only to the set of rotational linear systems. We call this set of skew-symmetric matrices $\mathbb{S}^{n \times n}$, and we are interested in solutions to the original least squares problem of the form $M_{skew} \in \mathbb{S}^{n \times n}$, namely $M^* = \text{argmin}_{M \in \mathbb{S}^{n \times n}} ||\dot{X} - XM||_F$.

To solve this constrained optimization, we use an equivalent but slightly more cumbersome notation. We begin by noting that in the original unconstrained least squares solution $X \backslash \dot{X}$, the columns of $M$ are solved *independently* of each other. If we write $M^* = X \backslash \dot{X}$ in the expanded least squares form

$M^* = (X^T X)^{-1} X^T \dot{X}$, we see that each column of $\dot{X}$ determines the corresponding column of $M$ independently of other columns. Thus, we can rewrite the original problem as a vector problem instead of a matrix problem. We introduce the vector $\mathbf{m} \in \mathbb{R}^{n^2}$, which is simply the matrix $M \in \mathbb{R}^{n \times n}$ unrolled to a vector (we denote this using the common notation $\mathbf{m} = M(:)$). We can then rewrite the unconstrained least squares problem as $\mathbf{m}^* = \text{argmin}_{\mathbf{m} \in \mathbb{R}^{n^2}} ||\dot{\mathbf{x}} - \tilde{X}\mathbf{m}||_2$, where $\dot{\mathbf{x}} = \dot{X}(:)$, and $\tilde{X}$ is a block diagonal matrix with the matrix $X$ repeated on the $n$ diagonal blocks. These two forms of the least squares problem give identical solutions (it is just a formatting choice: do we write the solution as a vector $\mathbf{m}$ or a matrix $M$).

Now we rewrite the rotational optimization problem in this vector notation, as that will allow us to naturally incorporate the constraint that $M^*$ be a member of $\mathbb{S}^{n \times n}$. Whereas previously we sought to solve the unconstrained $M^* = \text{argmin}_{M \in \mathbb{R}^{n \times n}} ||\dot{X} - XM||_F$, here we want to solve the constrained $M^* = \text{argmin}_{M \in \mathbb{S}^{n \times n}} ||\dot{X} - XM||_F$. For this set of skew-symmetric matrices $\mathbb{S}^{n \times n}$, by virtue of the constraint $M_{skew} = -M_{skew}^T$, these matrices only have $n(n-1)/2$ free parameters (not the $n^2$ of general matrices $\mathbb{R}^{n \times n}$).

The key step is to note that we can represent skew-symmetric matrices as a vector of $n(n-1)/2$ free parameters: we call these vectors $\mathbf{k} \in \mathbb{R}^{n(n-1)/2}$. Further, we can specify a linear map from these vectors onto the space of our vectors $\mathbf{m} \in \mathbb{R}^{n \times n}$. This matrix, which we call $H$, simply places each of the elements of $\mathbf{k}$ into two indices in $\mathbf{m}$. We can consider $\mathbf{k}$ to be the lower triangle of a skew-symmetric matrix. Suppose we have an element of $\mathbf{k}$, $k_r$, that corresponds to the $(i, j)$ element of the matrix for $i > j$ (the lower triangle). $H$ then takes this element $k_r$ and places: (1) the element $k_r$ in the element of $\mathbf{m}$ corresponding to the $(i, j)$ index of a matrix $M$, and (2) the negated element $-k_r$ in the element of $\mathbf{m}$ corresponding to the $(j, i)$ index of $M$. By this construction we define a matrix $H$ that maps from $\mathbb{R}^{n(n-1)/2}$ (the number of free parameters in a skew-symmetric matrix) to $\mathbb{R}^{n^2}$.

Finally, we return to our problem of interest. Since, by our construction, the quantity $H\mathbf{k}$ is a vector in $\mathbb{R}^{n^2}$ and is equivalent to the set of all skew-symmetric matrices $\mathbb{S}^{n \times n}$, our original problem has the equivalence:

$$M^* = \text{argmin}_{M \in \mathbb{S}^{n \times n}} ||\dot{X} - XM||_F \quad \Longleftrightarrow \quad \mathbf{k}^* = \text{argmin}_{\mathbf{k} \in \mathbb{R}^{\frac{1}{2}n(n-1)}} ||\dot{\mathbf{x}} - \tilde{X}H\mathbf{k}||_2.$$

This form can then simply be solved in closed form by grouping $H$ with $\tilde{X}$ and solving $\mathbf{k}^* = (\tilde{X}H)\backslash\dot{\mathbf{x}}$. Since $\mathbf{k}^*$ is a vector of length $n(n-1)/2$ that defines a skew-symmetric matrix $M_{skew}^*$ (via $H$), and since skew-symmetric matrices describe rotational dynamics, this solution is the matrix defining the best-fitting rotational linear dynamical system.

As an implementation note, one might prefer to write the Lagrangian for the constrained optimization problem. Doing so will result in, depending on programming choices, a very similar implementation as above. In any event, this optimization problem has a unique global optimum, so any valid derivation will produce exactly the same result. As a second note, representing the large matrices $H$ and $\tilde{X}$ can be computationally burdensome. Instead, we can avoid the explicit creation of these matrices by solving $\mathbf{k}^* = (\tilde{X}H)\backslash\dot{\mathbf{x}}$ with a gradient based-method, which allows us to manipulate vectors and matrices in the most computationally convenient way. The result is an extremely fast and accurate method for solving

the skew-symmetric least squares problem (only slightly slower than regular unconstrained least squares).

We now have successfully calculated $M^*_{skew}$, the summary matrix that describes rotational dynamics in the data. Traditional PCA takes the summary matrix (data covariance) and does an eigenvalue decomposition, which creates a ranked set of orthonormal vectors that we can use to project our data. Decomposition of a skew-symmetric matrix also produces a ranked set of orthonormal vectors. The eigenvectors produced by the decomposition of $M_{skew}$ come in orthogonal, complex conjugate pairs (these vectors form a unitary matrix), and thus we can naturally think of the decomposition of $M_{skew}$ as producing orthogonal planes (the eigenvalues are purely imaginary). To find a plane - any pair of real-valued projection vectors $\{\mathbf{u}_{i,1}, \mathbf{u}_{i,2}\}$ - each complex conjugate pair of vectors $\{\mathbf{v}_{i,1}, \mathbf{v}_{i,2}\}$ is combined as $\mathbf{u}_{i,1} = \mathbf{v}_{i,1} + \mathbf{v}_{i,2}$ and $\mathbf{u}_{i,2} = j(\mathbf{v}_{i,1} - \mathbf{v}_{i,2})$, which are then suitably normalized. These $\mathbf{u}$ vectors can be used exactly as in traditional PCA to project the high dimensional data to lower dimensionality (e.g., by projecting onto just the first plane, after ordering from largest to smallest pair of imaginary eigenvalues).

The magnitude of the eigenvalues allow us to select the plane (or planes) with the highest frequency and consistency in the *rotational* linear dynamical system. Thus, projecting the data down onto these planes allows us to visualize planes in the data with significant rotations. By connection to the imaginary eigenvalues of $M_{skew}$, we call this algorithm jPCA. To be concrete, all figures showing projections (except those noted otherwise) show the top jPC plane, which is the plane of strongest activity within the rotational dynamical system that best fits the data.

It is important to revisit the fact that jPCA and PCA solve different objective functions and thus produce different results: while PCA finds directions of maximal variance, jPCA finds directions (planes) of significant rotational dynamics. Indeed, the plane with the strongest rotations could in principle capture very little data variance. Thus, to prevent finding significant rotations that capture little variance, jPCA was always applied after first using PCA to find the handful of dimensions ($k = 6$ typically) with the most variance. In this application, jPCA simply rotates the dimensions found by PCA to better reveal dynamical structure (see supplementary movie 2).

As a technical note, we are taking the plane of strongest rotations within the rotational dynamical system that best fits the data. As an alternative approach, one might try to directly fit the plane of strongest rotations from the data, which would in principle do at least as well as the first jPCA plane. Thus our jPC projections (cf. Figure 3 and 4) are theoretically conservative, though our empirical investigation indicates this is an insignificant effect. These two approaches are equivalent when the data is white ($X^T X = I$), and we note that whitening the data also produces very little change to the resulting projections.

As a final point, the jPCA algorithm can be readily modified to a symmetric version 'symmPCA' to focus on directions of largest expansion and contraction. The convenience of being able to eigendecompose a summary matrix and yield orthogonal vectors belongs to the class of normal matrices, which by definition are diagonalizable by a unitary matrix. The class of normal matrices includes symmetric and skew-symmetric matrices, among others. This fact suggests a broader class of PCA variants that are a subject of future work.

## ***Supplementary Movies***

***Supplementary movie 1.*** Neural trajectory in the walking monkey (2× real time). The movie begins with the monkey stationary. After ~8 seconds (16 seconds of real time), the monkey begins walking. After ~25 cycles, the monkey pauses, then begins walking again. The monkey ceases walking once more just before the movie ends. Clear rotations are seen only during epochs of walking (QuickTime 376 KB).

***Supplementary movie 2.*** Illustration of how the PCA axes were rotated to find the jPCA projection. The movie contains clips for three datasets: monkey B, monkey J-array, and monkey N-array. For each clip the opening frame plots the PCA-based projection ($PC_2$ versus $PC_1$). On each subsequent frame the projection is rotated until the jPCA projection is reached. Compare with figure 3*a,e,f* of the main text (QuickTime 3.1 MB).

***Supplementary movie 3.*** The jPCA projections as a function of time. The movie contains clips for four datasets: monkey B, monkey J3, monkey N, and monkey N-array. Time is $1/8^{th}$ real time and starts just as preparatory activity is giving way to movement-epoch activity. The projections differ slightly from those in figure 3 of the main text for two reasons. First, they are based on more time. Second, to allow jPCA to best isolate the plane that worked well across a broad range of times, analysis was based on the top 10 PCs. (The true dimensionality of the data is higher still, but we wished to remain modestly conservative). The shuffled controls in *supplementary movie 4*, and the analysis of EMG in *supplementary movie 5* similarly employ the greater range of times and projections based on the top 10 PCs (QuickTime 2.4 MB).

***Supplementary movie 4.*** Just as for *supplementary movie 3* but after applying the shuffle control. The movie contains clips for the same four datasets: monkey B, monkey J3, monkey N, and monkey N-array. The shuffle that was applied was version #1 (see *supplementary figure 2* and *3*). (QuickTime 1.8 MB).

***Supplementary movie 5.*** The jPCA projections as a function of time for populations of muscle recordings. The movie contains clips for three datasets: monkey A, monkey J3, and monkey N. (EMG data were not recorded for the full 108-condition task used for the array-based datasets). (QuickTime 1.2 MB)

## ***Supplementary References***


43 Bizzi, E., Cheung, V. C., d'Avella, A., Saltiel, P. & Tresch, M. Combining modules for movement. *Brain Res Rev* **57**, 125-133 (2008).

44 Tresch, M. C. & Jarc, A. The case for and against muscle synergies. *Curr Opin Neurobiol* **19**, 601-607 (2009).

45 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).

46 Moran, D. W. & Schwartz, A. B. Motor cortical representation of speed and direction during reaching. *J Neurophysiol* **82**, 2676-2692 (1999).