J Neurophysiol 91: 1899–1907, 2004;
10.1152/jn.00438.2003.

Recursive Bayesian Decoding of Motor Cortical Signals by Particle Filtering

A. E. Brockwell,[1] A. L. Rojas,[1] and R. E. Kass[1,2]

[1]*Department of Statistics and* [2]*Center for the Neural Basis of Cognition, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213*

Submitted 26 January 2003; accepted in final form 30 October 2003

Brockwell, A. E., A. L. Rojas, and R. E. Kass. Recursive Bayesian decoding of motor cortical signals by particle filtering. *J Neurophysiol* 91: 1899–1907, 2004; 10.1152/jn.00438.2003. The population vector (PV) algorithm and optimal linear estimation (OLE) have been used to reconstruct movement by combining signals from multiple neurons in the motor cortex. While these linear methods are effective, recursive Bayesian decoding schemes, which are nonlinear, can be more powerful when probability model assumptions are satisfied. We have implemented a recursive Bayesian algorithm for reconstructing hand movement from neurons in the motor cortex. The algorithm uses a recently developed numerical method known as "particle filtering" and follows the same general strategy as that used by Brown et al. to reconstruct the path of a foraging rat from hippocampal place cells. We investigated the method in a numerical simulation study in which neural firing rate was assumed to be positive, but otherwise a linear function of movement velocity, and preferred directions were not uniformly distributed. In terms of mean-squared error, the approach was ~10 times more efficient than the PV algorithm and 5 times more efficient than OLE. Thus use of recursive Bayesian decoding can achieve the accuracy of the PV algorithm (or OLE) with ~10 times (or 5 times) fewer neurons. The method was also used to reconstruct hand movement in an ellipse-drawing task from 258 cells in the ventral premotor cortex. Recursive Bayesian decoding was again more efficient than the PV and OLE methods, by factors of roughly seven and three, respectively.

INTRODUCTION

Information from multiple motor cortical neurons, each broadly tuned to hand velocity, may be combined to predict movement (Georgopoulos et al. 1988). This "population coding" of movement parameters is of interest partly for its possible role in the neural basis of action and also for its potential use in controlling robotic devices (Black et al. 2003; Chapin et al. 1999; Taylor et al. 2002). Decoding of the population signal has been accomplished successfully with the population vector (PV) algorithm (Georgopoulos et al. 1988, 1989; Taylor et al. 2002). The PV method characterizes each neuron's activity by preferred direction and firing rate and performs optimally when the tuning functions are linear and the set of preferred directions are uniformly distributed (e.g., Zhang et al. 1998). To improve performance from a set of neurons the preferred directions of which are not uniformly distributed, Salinas and Abbott (1994) suggested a more general method, which they called optimal linear estimation (OLE). The PV method may be considered a special case of least-squares estimation (reverse regression) when the preferred directions are uniformly distributed. OLE uses least-squares without this restrictive assumption and furthermore can take account of the correlation between firing rates of the many different neurons. OLE estimates can be an order of magnitude better than PV estimates in terms of mean-squared error. However, they are still unable to account for the inherent dynamic behavior of the underlying signal of interest. An arm-velocity signal, for instance, cannot change by a large amount instantaneously—it must change gradually as the arm accelerates or decelerates. Furthermore, both the PV or OLE method assume the firing rates are linearly related to the underlying movement direction.

Bayesian decoding (Brown et al. 1998; Gao et al. 2002; Oram et al. 1998; Sanger 1996; Shoham 2001; Zhang et al. 1998), which is based on specification of a formal statistical model for neuronal spike trains (or spike counts) and its relationship to the behavior of interest (e.g., hand velocity), is much more flexible: it can accommodate not only correlated firing rates (as OLE does) but also nonlinear relationships between the signal and neuron firing rates. Bayesian methods make optimal use of the information available in the data when their modeling assumptions are satisfied (DeGroot 1970), which, in this context, translates into greater decoding accuracy using smaller numbers of neurons (e.g., Oram et al. 1998). Furthermore, using a recursive formulation, they offer a natural framework for a sequence of predictions, as is needed in neuroprosthetic and other dynamically evolving behavioral applications (Brown et al. 1998, 2001; Gao et al. 2002). In the past, recursive Bayesian methods have been difficult to implement without making restrictive assumptions, but a recently developed technique known as "particle filtering" (PF) (see, e.g., Doucet et al. 2001) has largely addressed this problem. We have implemented a particle filter for recursive Bayesian decoding, and investigated it in two studies. The first used numerical simulation where probability modeling assumptions were known to be valid. The second study involved analysis of data collected from monkey motor cortical neurons during ellipse-tracing experiments.

METHODS

We consider three classes of methods for decoding velocities: population vector methods, optimal linear estimation methods, and recursive Bayesian methods. Throughout the paper, velocities are taken to be either two- or three-dimensional vectors encoding both direction and speed (magnitude).

Linear decoding methods

The population vector method predicts velocity at time t as $\hat{v}_t^{(\mathrm{PV})}$, given by

$$\hat{v}_t^{(\mathrm{PV})} = \sum_{j=1}^{N} w_{t,j} d_j \qquad (1)$$

Address for reprint requests and other correspondence: A. E. Brockwell (E-mail: a.brockwell@ieee.org).

The costs of publication of this article were defrayed in part by the payment of page charges. The article must therefore be hereby marked "*advertisement*" in accordance with 18 U.S.C. Section 1734 solely to indicate this fact.

where N is the number of neurons used, d_j denotes the "preferred direction" of the jth neuron, that is, the unit-magnitude vector whose direction maximizes the tuning function $r_j(\cdot)$, and $w_{t,j}$ represents a weight determined by the firing rate of the jth neuron at time t. The weights in *Eq. 1* are determined by a formula such as

$$w_{t,j} = (y_{t,j} - \bar{y}_j)/(y_j^{(\max)} - y_j^{(\min)}) \quad (2)$$

where $\bar{y}_j$, $y_j^{(\max)}$, and $y_j^{(\min)}$ represent the average, maximum, and minimum firing rates of the jth neuron, respectively, over some time window. A scaling transformation must then be applied to the estimates $\hat{v}_t^{(\mathrm{PV})}$ to convert them to physical units of measurement, such as meters per second.

The optimal linear estimation method generalizes the PV method for the case where preferred directions are not uniform; the OLE estimates are given by

$$\hat{v}_t^{(\mathrm{OLE})} = \sum_{j=1}^{N} w_{t,j} d_j^* \quad (3)$$

where the vectors d_j^* are chosen so that the mean-squared error $E[(v_t - \hat{v}_t)^T(v_t - \hat{v}_t)]$ is minimized. Salinas and Abbott (1994) showed that the vectors d_j^* can be obtained as the solution of a particular linear system of equations. Computational details are given in APPENDIX B.

Recursive Bayesian decoding and PF

Recursive Bayesian decoding, described in more detail in APPENDIX A, relies on formal specification of a statistical model, consisting of two parts: a *state model*, for a process $\{v_t\}$, describing the evolution of the state we are trying to predict (here, velocity), and an *observation model* specifying the probability distribution of the data y_t given the underlying state v_t.

The state model should capture probabilistic features of the state (velocity) process. In this paper, we use state models that constrain the sequence of states $\{v_t, t = 0, 1, 2, \ldots\}$ so that they are likely to evolve with some degree of smoothness from one time to the next. Although we here restrict attention to either two- or three-dimensional velocity vectors, more generally the states could involve position, acceleration, or other movement parameters, and the framework may be extended to more sophisticated models, taking into account momentum, relationships between path-curvature and acceleration, etc.

We take $y_t = (y_t^{(1)}, \ldots, y_t^{(N)})$ to be the vector of spike counts for the complete set of N neurons, in the tth time bin, and $\lambda_i(x)$ to be the tuning function specifying the average firing rate for the ith neuron when the hand velocity is equal to x. We then specify the observation model by assuming the spike counts $y_t^{(i)}$ have Poisson distributions

$$y_t^{(i)} | v_{t+\mathrm{lag}_i} \sim \mathrm{Poisson}(\lambda_i(v_{t+\mathrm{lag}_i})) \quad i = 1, 2, \ldots, N \quad (4)$$

where lag_i is an integer-valued neuron-dependent lag measured in time bins. [Note that alternative observation models, such as those developed in Shoham (2001) may be specified.]

The decoding algorithm computes the conditional expectation of v_t, given observations $y_1, y_2, \ldots, y_t$. The underlying theory and our particle-filter implementation are described in APPENDIX A. Note that although we considered velocity in the studies described in this paper and obtained position simply by integrating decoded velocity, it would also be possible to adopt a more sophisticated approach and include position in the state variables $\{v_t\}$ as well as velocity. The conditional expectation of v_t would then include information about *both* velocity and position.

Simulation study

To examine the performance of the particle filtering algorithm, we simulated spike trains for 200 neurons, assuming the two-dimensional velocity v_t traces out the path shown in Fig. 2*A* over the course of 12 s. (The path was defined by $x_t = 6\cos(\pi t/6)$, $y_t = 2\sin(\pi t/2)$, for $t \in [0, 12]$, with velocity being defined by the respective derivatives.) The 12 s in the experiment were divided into 400 time bins, each of length 30 ms. The tuning functions

$$\lambda_i(v) = \max(k_i + m_i v \cdot d_i, 0)$$

were used, where k_i and m_i are positive constants determining, respectively, base firing rate, and directional sensitivity of the neuron, and d_i is a unit-vector representing the preferred direction of the ith neuron. Each of the 200 neurons was assigned a random preferred direction d_i as well as random values of k_i and m_i. Parameters were chosen so that the maximal firing rate over the simulation was ~100 Hertz. These rates are roughly consistent with the ventral premotor cortex data studied in the next section, and also (for instance), with rates shown in Fig. 2 of Kakei et al. (2001).

Given the velocities, the spike counts were taken to have probability distributions specified by *Eq. 4*, with all lags lag_i equal to zero. Following Brown et al. (1998), we took the state model to be a random walk, meaning that velocity at time t is assumed to equal velocity at time $(t - 1)$ plus noise, that is

$$v_t = v_{t-1} + \varepsilon_t \quad (5)$$

where $\{\varepsilon_t\}$ is an independent and identically distributed sequence of bivariate Gaussian random variables with means 0 and covariance matrices 0.03 times the identity matrix. [This is similar to the continuity constraints used in Zhang et al. (1998).] Choice of the coefficient of the covariance matrix here is important. Making it too large reduces the smoothing benefit of the recursive Bayesian decoding scheme, whereas choosing it too small can distort the decoded trajectory by over-smoothing it. The value 0.03 is chosen to give a model for velocity which is somewhat "realistic"—see APPENDIX A for more details. In this case, it means that the change in velocity (in 1 of the 2 dimensions) from one time bin to the next is ~95% likely to lie in the range between plus and minus $1.96\sqrt{0.03} \approx 0.34$. In fact, this is conservatively large because in the simulated trajectory, the greatest difference between successive values is ~0.14. However, because the true trajectory is not generally known in advance, there is always an element of subjective judgment involved in selecting this parameter.

To investigate the ability of the OLE and PF methods to handle nonuniformly distributed preferred directions, we concentrated half of the preferred directions uniformly in the angular range from 0 to $\pi/2$, and the other half uniformly in the range from $\pi/2$ to 2π. The simulation was repeated to generate a total of 60 independent data sets. From the 60 replications, we computed statistical summaries to evaluate the PV, OLE, and PF methods according to criteria described in the following text.

We implemented the PF using 2,500 particles at each time point (see APPENDIX A for details). For the PV method, we chose the linear scaling function to minimize the sum of squared differences between the decoded and actual trajectories. This optimal scaling function would not be possible to determine if the actual trajectory was unknown (as in the application to brain-controlled robotic devices) but was used here to give an additional advantage to the PV method.

Motor cortex data

To investigate the performance of Bayesian decoding in a realistic situation, we considered spike trains from 258 neurons in the subregion of ventral premotor cortex referred to by Gentilucci et al. (1998) as "region F4," collected individually in 258 separate experiments from four rhesus monkeys (described in more detail in Reina and Schwartz 2003). In each experiment, the monkey carried out multiple repetitions of a center-out task followed by multiple repetitions of an ellipse-tracing task.

For each repetition of the center-out task, the monkey reached to the eight corners of a virtual cube. Each of the eight reaching motions was

subdivided into 100 equal-length time bins, which were ~10 ms long, and spike counts for the time bins, as well as hand-position at the beginning and end of each time bin, were recorded. In each repetition of the ellipse-tracing task, monkeys continuously traced five elliptical loops. After each entire repetition was successfully completed, the monkey was rewarded. For each repetition of each experiment, neuron spike times and three-dimensional hand positions were recorded. The ellipses were traced in the *x*-*y* plane, with the *z*-component capturing relatively insignificant deviations of the hand from this plane. The recorded data were then processed as follows. For each repetition, the duration of each of the five loops was divided into 100 equal-sized time bins, yielding 500 measurements of hand-position along with corresponding spike counts. Average time bin width was ~15 ms. We defined "hand-velocity" to be the difference between hand-position in successive bins.

We used PV, OLE, and PF methods to decode the fifth loop of the first repetition of the ellipse-tracing experiment. As is common in the population vector literature, we treated the data from the 258 experiments as if they were collected in a single experiment measuring 258 neurons simultaneously. This raised the problem of deciding what to compare decoded trajectories with because in reality there were 258 with similar but slightly different trajectories. We compared decoded velocities with "actual velocity," which we defined to be the average of observed hand-velocities over the 258 individual experiments.

Preferred directions for the neurons were obtained by analyzing the center-out data, while additional parameters were estimated using only the first three loops of the first repetition of the ellipse-tracing experiment. Thus we maintained strict separation of the data used to determine decoding parameters and the data used to evaluate the performance of the decoding schemes.

Decoding details for the motor cortex data

In implementing the PF for the motor cortex data, we used tuning functions

$$\lambda_i(v) = \exp(k_i + m_i v \cdot d_i + s_i \|v\|)$$

where k_i and m_i, as in the simulation study, represent base firing rate and directional sensitivity, and the additional parameter s_i represents nondirectional sensitivity of the neuron to speed. (see, e.g., Schwartz 1992 for discussion of the importance of speed in this context.) The three-dimensional preferred directions d_i were estimated from center-out data only, and scalar parameters k_i, m_i, and s_i, for neurons $i = 1, 2, \ldots, 258$, were estimated based on the first three loops of the first repetition of the ellipse-tracing ex periment, using standard Poisson-family generalized linear models (McCullagh and Nelder 1989), with the restriction that $m_i \geq 0$. This restriction was imposed to prevent the preferred direction from effectively being reversed. Because the task involved tracing in the *x*-*y* plane, with only minimal movement in the *z*-direction, it would also have been possible to reduce the data and model to only these two dimensions. However, for the sake of testing the algorithm more thoroughly, we chose to develop the full three-dimensional model.

For each neuron, a range of lags from −40 to +40 time bins (corresponding to a range of approximately −600 to +600 ms) was used, and the lag yielding the best-fitting generalized linear model, as determined by comparing deviances of models, was selected. A histogram of the resulting lags for 80 of the neurons is shown in Fig. 1. These neurons were selected as those which spiked ≥10 times during the first five-loop tracing trial, and also had $m_i \geq 0.05$ (m_i can be regarded as a parameter defining directional sensitivity of the neuron, and 0.05 was approximately the median fitted value of m_i over the 258 neurons).

We imposed smooth acceleration on the hand motion by using the state model

$$v_{t+1} - v_t = (v_t - v_{t-1}) + \varepsilon_{t+1} \quad (6)$$

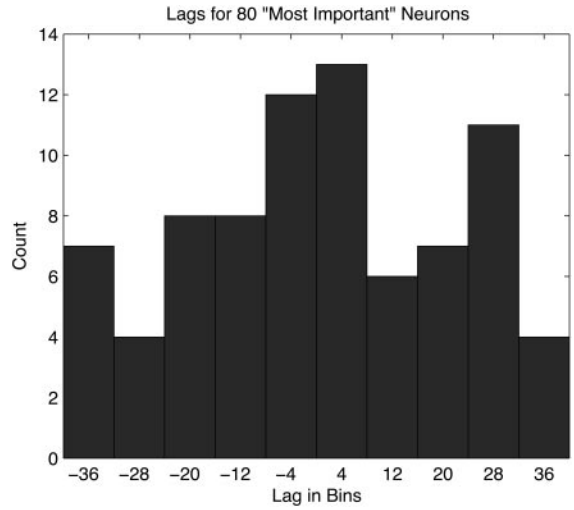

FIG. 1. Histogram of the lags, measured in time bins, of 80 neurons in the ventral premotor cortex data. Neurons were selected as those which spiked ≥10 times during the ellipse-tracing task and for which directional sensitivity was above the median directional sensitivity.

where $\{\varepsilon_t, t = 1, 2, \ldots\}$ is a sequence of independent Gaussian disturbances with mean zero and 3 × 3 diagonal covariance matrix with entries {0.006, 0.006, 0.001}. These values were chosen, as in the simulation example, with the aim of giving a realistic model for the evolution of the velocity over time. In this case, the values can be interpreted as meaning that a 95% confidence interval for the change in the change in *x* and *y* components of velocity over successive time bins is approximately between ±1.96 [racical]0.006, which is roughly consistent with the actual trajectories. (Note that the 0.001 value corresponds to the *z* axis, which is orthogonal to the plane in which the ellipse is traced.) This is a slightly stronger smoothness constraint than used in the simulation exercise—we chose this constraint because in this case we knew a priori that the ellipse tracing experiment would generate trajectories with smoothly varying acceleration. As in the simulation study, we used 2,500 particles at each time point in our implementation of the PF. Furthermore, to cast the state model in the form required in APPENDIX A (*State model*), we defined six-dimensional vectors $v_t^* = (v_t, v_{t-1})^T$, and worked with the equivalent representation of *Eq. 6* given by

$$v_{t+1}^* = \begin{bmatrix} 2I & -I \\ I & 0 \end{bmatrix} v_t^* + \begin{bmatrix} \varepsilon_{t+1} \\ 0 \end{bmatrix}$$

where I denotes the 3 × 3 identity matrix. The algorithm was then implemented exactly as described in APPENDIX A but with v_t replaced by v_t^*.

For PV decoding, preferred directions were taken to be normalized version of the vectors d_i obtained for PF decoding, and lags were taken to be the same as those used for PF decoding. To determine the weights used in *Eq. 1*, we computed, for each neuron, average, minimum, and maximum counts over the time bins in the first three loops of the first repetition of the ellipse-tracing experiment. Weights were then computed using *Eq. 2*. After decoding, results were scaled and offset so as to minimize mean-squared error between the first three loops of the trajectory and the decoded trajectory.

To determine vectors d_j^*(recall *Eq. 3*) in the OLE method, we used a Monte-Carlo procedure, described in detail in APPENDIX B.

Comparison of algorithms

For both the simulation study and the ventral premotor cortex data analysis, we assessed quality of decoded signals by two measures: the integrated squared error (ISE) and the maximum squared error

(MaxSE). For a particular data set, the ISE is the squared difference between the decoded and actual velocities, averaged over all time bins, and the MaxSE is the largest squared difference between decoded and actual velocity among all time bins. Because ISE and MaxSE will vary from data set to data set, it is customary in the statistical literature to use the averages of such criteria across multiple simulated data sets, the averages being estimates of their theoretical expected values. For the simulation study, we computed the averages (arithmetic means) of ISE and MaxSE across the 60 simulated data sets and refer to them as MISE and MMaxSE, respectively. For typical statistical methods of estimation, these measures will tend to decrease proportionally to the inverse of the number of neurons. In this context squared error thus has the following very useful interpretation: the accuracy of PF based on N_{PF} neurons will be comparable to the accuracy of PV based on N_{PV} neurons when $N_{PV} = N_{PF} \times R$ and R is the ratio of the MISE for PV to the MISE for PF.

RESULTS

Simulation study

For the simulation study, the MISE and MMaxSE for each of the three methods is given in Table 1. In this nonuniform and nonlinear case, OLE performs better than the PV algorithm, and the PF is clearly superior to both: the MISE for the PF is ~10 times smaller than that for the PV algorithm and ~5 times smaller than that for OLE. In large samples, expected squared error is inversely proportional to sample size. Therefore these results may be interpreted as saying that ~10 times more neurons would be needed when using the PV algorithm (5 times when using OLE) to obtain the same error, averaged across time, as the PF (e.g., 250 neurons would be needed with PV to obtain the same accuracy as the PF based on 25 neurons).

Figure 2 displays decoded velocity for 1 of the 60 data sets in the simulation study. It may be seen that PF reduces both the bias and the noise in the PV reconstruction, and it is much less noisy than OLE. Due to the nonuniformity of preferred directions, there are particular regions of the velocity trace where the PV algorithm is especially inaccurate. This produces the MMaxSE of 3.000 reported in Table 1.

To further illustrate how the PF method works, Fig. 3 shows the first 100 particles $\hat{x}_t^{(j)}$, $j = 1, 2, \ldots, 100$ (see APPENDIX A, *Algorithm,* for details), at times $t = 100$ (3 s) and $t = 300$ (9 s), along with the entire velocity trajectory, for a typical run of the decoding algorithm. The actual decoded velocity at each point in time is taken to be the average of the corresponding "cloud" of particles.

It is also worth noting that the algorithm was relatively fast to execute. For this simulation study, a single run of the PF algorithm with 2,500 particles, over all 400 time points, took approximately three minutes on a Pentium-4 machine. (The algorithm was implemented using programs written in C++.)

TABLE 1. *Mean-integrated squared error (MISE) and mean maximum squared error (MMaxSE) for population vector (PV), optimal linear estimation (OLE), and particle filtering (PF) methods in the simulation study*

	PV	OLE	PF
MISE	0.712 ± 0.023	0.327 ± 0.003	0.068 ± 0.001
MmaxSE	3.000 ± 0.081	2.328 ± 0.069	0.530 ± 0.017

MISE averages, while MMaxSE maximizes, squared error over time. Values (from the 60 data replications) means ± SE. In terms of MISE, PF is ~10 times more efficient than PV.

Motor cortex data

Results are shown in Fig. 4. Again, the PF is much less noisy than the PV algorithm, and its improved accuracy is apparent especially from the squared errors displayed in Fig. 4*C*. Summaries of the squared errors are given in Table 2.

Again, as in the simulation study, particle filtering performs better than the PV and OLE methods, increasing efficiency by factors of approximately seven and three, respectively.

DISCUSSION

We have specified an implementation of particle filtering for Bayesian decoding and shown that the algorithm can substantially outperform the PV and OLE methods. The improved accuracy of velocity prediction was due to the Bayesian algorithm's ability to adapt to both nonuniformly distributed preferred directions and nonlinear tuning functions and, by virtue of its recursive sequential formulation, to smooth the predictions across neighboring time points. Note that it may be possible to improve the performance of PV and OLE methods by altering the bin size, thereby effectively altering the degree of smoothing, but compared with the recursive Bayesian approach, in which smoothing is implicitly controlled by specification of the state model, this would be a relatively ad hoc procedure with limited flexibility and effectiveness. Our results support the general arguments of Oram et al. (1998) and supplement the work on recursive methods by Brown et al. (1998, 2001), Gao et al. (2002), and Shoham (2001), demonstrating the power of the Bayesian sequential formalism in creating effective decoding schemes.

Recursive Bayesian decoding of cortical signals uses a probability model for the firing rate as a function of relevant behavioral parameters and a probability model for the evolution of those parameters. When these probability models provide a reasonable approximation to the phenomena under study, we may expect the approach to produce superior results. It is worth emphasizing a key distinction between our two studies: in the simulation study, the assumed probability model for the relationship between velocity and the spike trains matched perfectly the actual relationship because this was under our control. As a consequence, the Bayesian decoding scheme performed extremely well as theoretical arguments say it should. On the other hand, with the ventral premotor cortex data, we implemented an imperfect probability model, hoping that it would provide a reasonable approximation to reality. Although the PF algorithm remained clearly superior to the PV and OLE algorithms, the gain was slightly reduced from the 10-fold (resp. 5-fold) increases in efficiency seen in the simulation study. It is worth noting the wide spread of estimated lags obtained for the ventral premotor cortex data as shown in Fig. 1. Due to the repetitive nature of the ellipse-tracing task, it is possible that neurons with higher magnitude lags are in fact coding aspects of behavior that are only indirectly connected with motor coordination of the task. In this case, one might expect the methods to be less effective for less repetitive tasks. However, the relative performances of the different methods we consider should remain roughly the same because

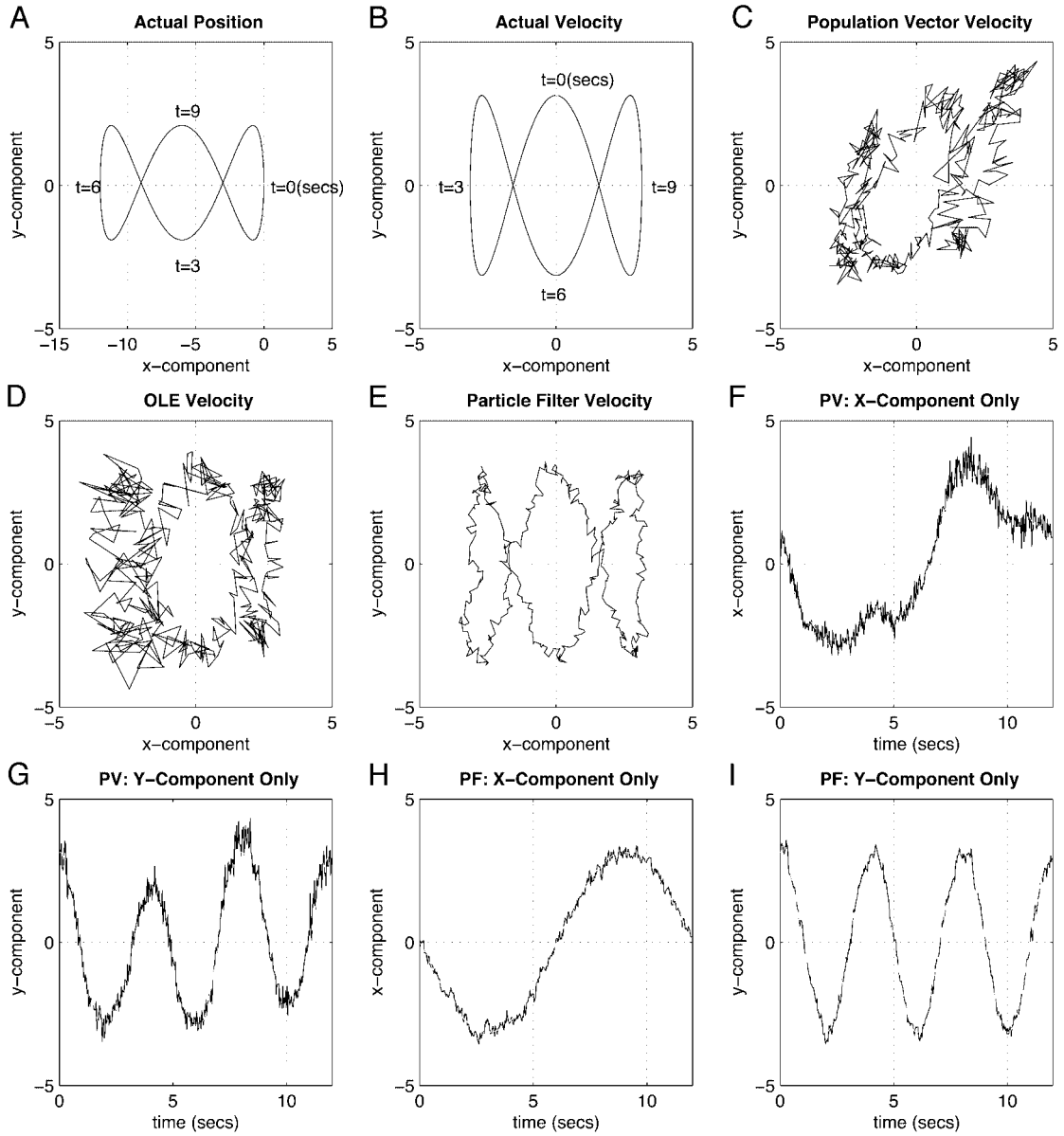

FIG. 2. Results for 1 of the 60 simulated 200-neuron data sets, with nonuniformly distributed preferred directions. *A*: the actual position trajectory from *time 0* to time 12; *B*: the corresponding velocity trajectory; *C*: population vector (PV)-decoded velocity trajectory; *D*: optimal linear estimation (OLE)-decoded velocity trajectory; *E*: particle filtering (PF)-decoded velocity trajectory; *F–I*: *x* and *y* components of decoded velocity as a function of time for both PV and PF methods with actual velocity shown (- - -). Due to nonuniformity of preferred directions, the PV-decoded trajectory misses its target noticeably (*C*) in several places; in terms of *x* and *y* components the error is greatest near times of peak *x* and *y* velocity (*F* and *G*). The PF is highly accurate throughout.

the same advantage is conferred on all three methods we consider.

In considering the distinction between the simulation and real-data studies, it should be kept in mind that the premotor cortical neurons were not recorded simultaneously, and the "actual velocity" was taken to be an average across 258 very similar experimental trials. It is possible that the motor cortical signal contains information about small time-locked trial-to-trial fluctuations in hand velocities, in which case, it would be reasonable to expect further benefit from the PF algorithm when predictions are compared with contemporaneous behavior. More fundamentally, the improvement for simulated data underscores the essential challenge of probabilistic decoding: gains in performance will likely accrue as additional features are built into the probability models. For example, the flexible framework of recursive Bayesian decoding allows it to accommodate such things as variable time lags between neuronal activity and movement, fine-scale time resolution, and correlation structure, all of which could lead to much better performance in driving prosthetic devices.

The framework we have used here is based on the full, sequential form of Bayes' Theorem, aided by its simple recursive structure (see APPENDIX A). This follows the general approach of Brown et al. (1998) and Gao et al. (2002) but should be contrasted with the static versions of Bayes' Theorem discussed by Oram et al. (1998), Sanger (1996), and Zhang et

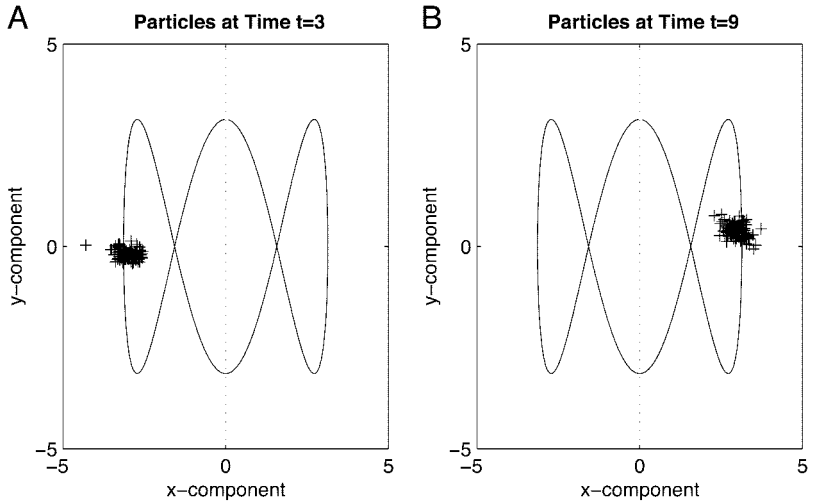

FIG. 3. Typical "particle clouds" obtained using the PF Algorithm, superimposed on the actual velocity trajectory. *A*: the 1st 100 particles at time $t = 100$ (3 s); *B*: the 1st 100 particles at time $t = 300$ (9 s). Decoded velocities at times $t = 3$ s and $t = 9$ s are obtained by averaging the values of the respective particle clouds.

al. (1998). It is also important to distinguish the general Bayesian analysis formalism from its implementation. APPENDIX A of this paper presents the simplest form of the particle filtering algorithm, which is easily implemented. The method worked well, but it is important to keep in mind that it may sometimes suffer from a problem known as "particle depletion," which arises typically when the probability model is poor, the state vector is high dimensional, or large outliers occur in the data. Further discussion of this topic is beyond the scope of this paper, but many more details of the problem and a number of proposed solutions can be found in Doucet et al. (2001). In addition to particle filtering, alternative methods have been proposed for implementation of recursive Bayesian schemes. Brown et al. (1998, 2001) and have used Gaussian approxima-

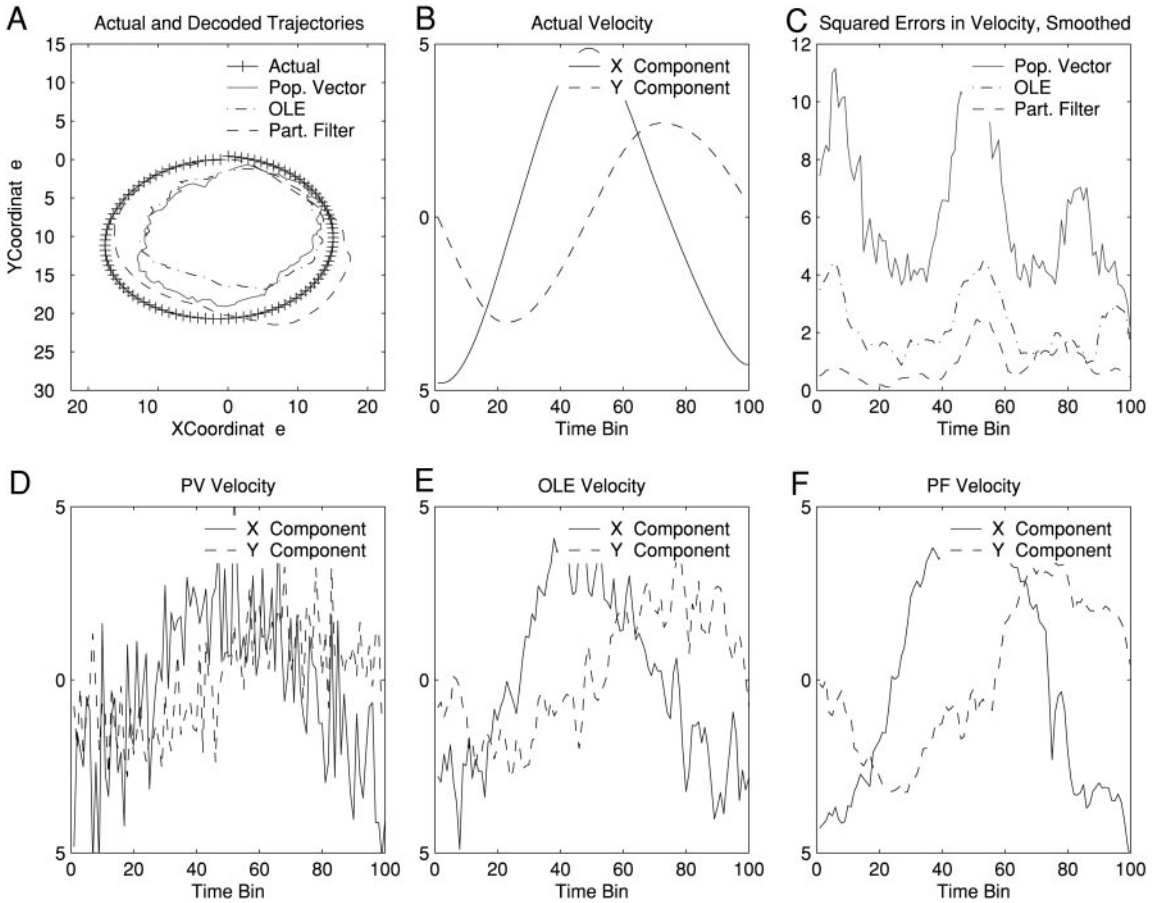

FIG. 4. Decoding of *x* and *y* components of 5th loop of the ellipse-tracing task, based on 258 neurons in the ventral premotor cortex. *A*: actual position trajectory along with decoded trajectories obtained by integrating decoded velocity; *B*: actual velocity over the 100 time bins in the 5th loop; *C*: squared errors of decoded velocities for the 3 decoding methods, smoothed with a moving average smoother to make them easier to distinguish; *D*: decoded velocity (both *x* and *y* components) for the PV method; *E*: decoded velocity for the OLE method; and *F*: decoded velocity for the PF method.

TABLE 2. *Decoding errors for ventral premotor cortex data summarized across time*

	PV	OLE	PF
ISE	6.245	2.362	0.886
MaxSE	33.978	9.349	4.904

ISE averages, while MaxSE maximizes, squared error across time. In terms of ISE, the PF offers a roughly sevenfold improvement over the PV algorithm.

tions, effectively employing Laplace's method (e.g., Kass 1997), which in many situations furnishes highly accurate approximations with a substantially smaller computational cost than simulation-based methods such as the particle filter. Our own preliminary work suggests that in the context of motor cortical signal decoding, with substantial numbers of neurons and smoothly varying movements, Gaussian approximations can be nearly as accurate as particle filtering. A number of variations can be considered, along the lines of Gao et al. (2002). These, and related alternatives to the particle filtering algorithm used here, including Gaussian approximations, are subjects for future investigation.

APPENDIX A: DETAILS OF BAYESIAN DECODING AND PARTICLE FILTERING

In Bayesian decoding, the object is to find, for each time t, the distributions of the unobserved signal v_t, given observations $\{y_1, y_2, \ldots, y_t\}$. [In this paper, v_t has been used to represent the velocity of a monkey's arm during the tth time bin with the 2- or 3-dimensional vector encoding both speed (magnitude) and direction, while the observations y_t represented vectors of spike counts during the respective time bins.] The mean of the distribution of v_t given $\{y_1, y_2, \ldots, y_t\}$ can then be used as a point estimate of v_t.

The fundamental components of any recursive Bayesian decoding scheme are two statistical models that must be specified: a "state model" and an "observation model." The general approach to recursive Bayesian decoding consists of the following steps. *1*) Specify a state model. *2*) Specify an observation model. *3*) Implement the particle filter, or a suitable alternative scheme, to estimate the unobserved sequence $\{v_t\}$.

In what follows, we use the (standard) notation $p(x)$ to denote the density function of a generic random variable x, and we denote the conditional density of x given another random variable y by $p(x|y)$.

State model

The state model describes the distribution of the unobserved signal one step in the future, v_{t+1}, given the current value of the signal v_t. In other words, the state model specifies

$$p(v_{t+1}|v_t)$$

In many cases, a random walk can be used here. For instance, choosing $p(v_{t+1}|v_t) = (2\pi)^{-\dim(v_t)/2} \det(\Sigma)^{-1/2} \exp\{-(v_{t+1} - v_t)^T \Sigma^{-1}(v_{t+1} - v_t)/2\}$ specifies that given v_t, v_{t+1} has a Gaussian distribution with mean v_t and covariance matrix Σ. This simple random walk model effectively imposes a continuity constraint on the underlying signal process $\{v_t\}$. The covariance matrix should be chosen so that the model for unobserved signal is not unrealistic. One way to do this would be to construct a number of "typical" paths for the unobserved signal, compute the sample covariance matrix of all the steps $(v_{t+1} - v_t)$ in all the paths, and use this as Σ.

The initial distribution $p(v_0)$ must also be specified. If v_0 was known exactly, this distribution would be concentrated on one point (the starting point). However, in many cases, v_0 is unknown, and this distribution is chosen to represent an initial guess of the signal. A common choice would be to take $p(v_0)$ to be multivariate Gaussian, with mean equal to a "best guess" at the starting value v_0, and variance reflecting uncertainty (higher variance corresponding to higher uncertainty). Roughly speaking, a (say) 95% confidence interval for the initial value based on this distribution should correspond to a region in which an individual would be 95% certain that the true initial value actually lies.

Observation model

The observation model specifies the relationship between the unobserved signal v_t and the observation y_t, that is, it specifies $p(y_t|v_t)$. For the problems considered in this paper, this model is based on the assumptions that *1*) in a time bin, a neuron generates a Poisson-distributed spike count, with mean specified by a tuning-function relating arm velocity, adjusted by some neuron-dependent time lag, to the average spike count. *2*) Given arm velocity, the spike counts are independent of each other.

This leads to the relationship

$$p(y_t|v_t) = \prod_{i=1}^{N} \frac{\exp(-\lambda_i(v_t))[\lambda_i(v_t)]^{y_t^{(i)}}}{y_t^{(i)}!}$$

where $\lambda_i(v_t)$ is the tuning-function for the ith neuron, evaluated with velocity equal to v_t, and $y_t^{(i)}$ represents the spike count for the ith neuron, in the tth time bin. Lags can be accounted for by time shifting each one of the components of y_t according to the respective neuron's lag.

Other models could be used. In particular, one might propose alternatives to the Poisson distribution (as in Shoham 2001), which take into account position as well as velocity (this would require the position at time t to be included in the vector v_t) or any number of other possible models. As a general rule, more appropriate models lead to better decoding.

Recursions for posterior distributions

The distribution of v_t given the observations $\{y_1, \ldots, y_t\}$ up to and including the one in the tth time bin is referred to as a posterior distribution, and the set of these posterior distributions, for $t = 1, 2, \ldots$, is determined by the recursive relationships

$$p(v_t|y_1, \ldots, y_t) \propto p(y_t|v_t)p(v_t|y_1, \ldots, y_{t-1}) \quad \text{(A1)}$$

and

$$p(v_t|y_1, \ldots, y_{t-1}) = \int p(v_t|v_{t-1})p(v_{t-1}|y_1, \ldots, y_{t-1})\mathrm{d}v_{t-1} \quad \text{(A2)}$$

We start with $p(v_0)$ already specified (as part of the state model). *Equation A1* is then used to obtain $p(v_0|y_0)$. Then *Eq. A2* gives us $p(v_1|y_0)$. At this point, we go back to *Eq. A1* with $t = 1$ to get $p(v_1|y_0, y_1)$. The process continues in this manner indefinitely.

While this looks like a straightforward procedure, it is typically impossible to find analytical solutions to these recursions. Furthermore, because each iteration of the equations adds an integration operation, we end up with a high-dimensional integral for which standard numerical integration methods are too slow. However, the particle filtering algorithm given in the following text provides a simple way of approximating the desired posterior distributions.

Algorithm

The PF algorithm computes numerical approximations to the distributions in the recursions (*Eqs. A1–A2*). Let m denote a "number of particles," often chosen to be on the order of $m = 1{,}000$. The algorithm is as follows.

1) Create a population of particles $\tilde{x}_1^{(j)}$, $j = 1, 2, \ldots, m$, by repeatedly drawing from the initial distribution specified for the state model. Set $t = 1$.

2) Compute weights $w_t^{(j)}$, $j = 1, 2, \ldots, m$ using the formula

$$w_t^{(j)} = p(y_t|v_t = \tilde{x}_t^{(j)})$$

where $y_t = (y_t^{(1)}, \ldots, y_t^{(N)})$ is the observed set of spike counts at time t. If neurons are conditionally independent, given velocity, then this expression further reduces to

$$w_t^{(j)} = \prod_{i=1}^{N} p(y_t^{(i)}|v_t = \tilde{x}_t^{(j)})$$

3) Normalize the weights $w_t^{(j)}$ so that they sum to one and then draw a sample of size m, with replacement, from the set of particles $\{\tilde{x}_t^{(1)}, \tilde{x}_t^{(2)}, \ldots, \tilde{x}_t^{(m)}\}$, where each draw picks $\tilde{x}_t^{(j)}$ with probability $w_t^{(j)}$. Define this sample to be $\hat{x}_t^{(j)}$, $j = 1, 2, \ldots, m$. The estimate of the state at time t is then

$$\hat{v}_t = \frac{1}{m}\sum_{j=1}^{m} \hat{x}_t^{(j)}$$

4) For each particle $\hat{x}_t^{(j)}$, $j = 1, 2, \ldots, m$, simulate one step forward in the state equation, and define the simulated value to be $\tilde{x}_{t+1}^{(j)}$. In other words, draw $\tilde{x}_{t+1}^{(j)}$ from the distribution

$$p(v_{t+1}|v_t = \hat{x}_t^{(j)})$$

5) Replace t by $t+1$ and go back to step 2.

Intuitively, the algorithm can be thought of as follows. It starts with a collection of particles which can be thought of as initial guesses of v_1. Weights are then assigned to each guess (step 2), according to how likely y_1 is, given the particular guess. By resampling (step 3), we obtain a set of particles that is in a certain sense consistent with y_1; this set of particles in fact can be regarded as a draw from the posterior distribution $p(v_1|y_1)$. A good set of guesses for v_2, given y_1, can then be obtained by simulating one-step forward in time (step 4). By weighting proportionally to the likelihood of y_2 given these guesses (step 2 again) and resampling (step 3), we obtain an approximate draw from the distribution $p(v_2|y_1, y_2)$. This process repeats itself until we have scanned through all time points for which observations are available.

The desired estimates $\hat{v}_t$ obtained in step 3 of the algorithm are approximations to the means of the corresponding posterior distributions $p(v_t|y_t, \ldots, y_1)$. If desired, one can also estimate the variances of these distributions by the sample variances of the particles $\hat{x}_t^{(j)}$ to construct, for instance, predictive confidence intervals for the reconstructed velocities.

In this form, the algorithm does not apparently account for lags between the velocities $\{v_t\}$ and the observations $\{y_t\}$. However, this is fairly easy to do. We can assume that the ith neuron has an associated lag denoted lag_i, and that

$$y_t^{(i)}|v_{t-\text{lag}_i} \sim \text{Poisson}[\lambda_i(v_{t-\text{lag}_i})] \tag{A3}$$

(That is, given $v_{t-\text{lag}_i}$, $y_t^{(i)}$ has a Poisson distribution with parameter $\lambda_i(v_{t-\text{lag}_i})$.) Then to implement the particle filtering algorithm, we can simply replace the vector y_t used in step 2 with $y_t^* = (y_{t+\text{lag}_i}^{(1)}, \ldots, y_{t+\text{lag}_N}^{(N)})$, so that the weights become

$$w_t^{(j)} = \prod_{i=1}^{N} p(y_{t+\text{lag}_i}^{(i)}|v_t = \tilde{x}_t^{(j)})$$

APPENDIX B: COMPUTATIONAL METHOD FOR OLE

For decoding by OLE, it is necessary to determine the vectors d_1^*, $d_2^*, \ldots$, to be used in *Eq. 3*. Salinas and Abbott (1994) showed that these vectors satisfy the system of equations

$$D = Q^{-1}L \tag{B1}$$

where D is the $N \times d$ matrix with rows d_1^*, $d_2^*, \ldots, d_N^*$, L is an $N \times d$ matrix with rows $L_1, \ldots, L_N$ given by

$$L_i = \mathbf{E}[w_{t,i}v_t] \tag{B2}$$

where $\mathbf{E}[\cdot]$ denotes the standard statistical "expectation" (or averaging) operator, $Q = [Q_{ij}]_{i,j=1,\ldots,N}$ is an $N \times N$ matrix with elements

$$Q_{ij} = \mathbf{E}[w_{t,i}w_{t,j}] \tag{B3}$$

and d is the dimension of the velocity vector. Typically $d = 3$, but sometimes experiments restrict attention to movement in a plane, in which case we would use $d = 2$. (Note that $w_{t,i}$ depends indirectly on v_t because the distribution of $y_{t,i}$ depends on v_t.) In computing L and Q, the expectation is taken over some distribution of possible velocities v_t.

In general, for an arbitrary distribution of v_t, the expected values in the matrices L and Q do not have a closed-form solution. However, they can be estimated by Monte-Carlo simulation. To be specific, one can draw a large set of "possible velocities" $v_t^{(k)}$, $k = 1, 2, \ldots, m$ (in this paper we used $m = 100{,}000$), and then approximate L_i by

$$\frac{1}{m}\sum_{k=1}^{m} w_{t,i}^* v_t^{(k)}$$

where for each component of the sum, $w_{t,i}^*$ is a simulated value of the weight (*Eq. 2*), assuming that $v_t = v_t^{(k)}$. Similarly, Q_{ij} can be approximated by

$$\frac{1}{m}\sum_{k=1}^{m} w_{t,i}^* w_{t,j}^*$$

Once L and Q have been estimated, *Eq. B1* can be solved for D using standard routines available in mathematical software packages.

ACKNOWLEDGMENTS

The authors are grateful to T. Reina, A. Schwartz, and M. Velliste for valuable comments and supplying the data studied in this paper, to E. Brown for many helpful conversations, and to two anonymous referees for constructive comments and suggestions.

GRANTS

This work was supported in part by the National Institute of Mental Health Grant R01-MH-64537-01.

REFERENCES

Black MJ, Bienenstock E, Donoghue JP, Serruya M, Wu W, and Gao Y. Connecting brains with machines: the neural control of 2d cursor movement. In: *Proc. 1st International IEEE/EMBS Conference on Neural Engineering*. Capri, Italy: IEEE/EMBS, 2003, p. 580–583.

Brown EN, Frank LM, Tang D, Quirk MC, and Wilson MA. A statistical paradigm for neural spike train decoding applied to position prediction from ensemble firing patterns of rat hippocampal place cells. *Neuroscience* 18: 7411–7425, 1998.

Brown EN, Nguyen DP, Frank LM, Wilson MA, and Solo V. An analysis of neural receptive field plasticity by point process adaptive filtering. *Proc Nat Acad Sci USA* 98: 12261–12266, 2001.

Chapin JK, Moxon KA, Markowitz RS, and Nicolelis MAL. Real-time control of a robot arm using simultaneously recorded neurons in the motor cortex. *Nat Neurosci* 2: 664–670, 1999.

DeGroot MH. *Optimal Statistical Decisions.* New York: McGraw-Hill, 1970.

Doucet A, de Freitas N, and Gordon N (editors). *Sequential Monte Carlo Methods in Practice*. New York: Springer, 2001.

Gao Y, Black MJ, Bienenstock E, Shoham S, and Donoghue JP. Probabilistic inference of hand motion from neural activity in motor cortex. *Adv Neural Inform Process Syst* 14: 213–220, 2002.

Gentilucci M, Fogassi L, Luppino G, Matelli M, Camarda T, and Rizzolatti G. Functional organization of inferior area 6 in the macaque monkey. I. Somatotopy and the control of proximal movements. *Exp Brain Res* 71: 475–490, 1988.

Georgopoulos AP, Kettner RE, and Schwartz AB. Primate motor cortex and free arm movements to visual targets in three-dimensional space. II. Coding of the direction of movement by a neuronal population. *Neuroscience* 8: 2928–2937, 1988.

Georgopoulos AP, Schwartz AB, and Kettner RE. Neuronal population coding of movement direction. *Science* 243: 234–236, 1989.

Kakei S, Hoffman DS, and Strick PK. Direction of action is represented in the ventral premotor cortex. *Nat Neurosci* 4: 1020–1025, 2001.

Kass RE. Laplace's method. In: *The Encyclopedia of Statistical Sciences*. New York: Wiley, 1997, vol. I, p. 347–354.

McCullagh P and Nelder JA. *Generalized Linear Models* (2nd ed.). London, UK: Chapman & Hall, 1989.

Oram MW, Földiák P, Perrett DI, and Sengpiel F. The "ideal homunculus": decoding neural population signals. *Trends Neurosci* 21: 259–265, 1998.

Reina GA and Schwartz AB. Eye-hand coupling during closed-loop drawing: evidence of shared motor planning? *Hum Move Sci* 22: 137–152, 2003.

Salinas E and Abbott LF. Vector reconstruction from firing rates. *J Comput Neurosci* 1: 89–107, 1994.

Sanger TD. Probability density estimation for the Â interpretation of neural population codes. *J Neurophysiol* 76: 2790–2793, 1996.

Schwartz AB. Motor cortical activity during drawing movements: Single-unit activity during sinusoid tracing. *J Neurophysiol* 68: 528–541, 1992.

Shoham S. *Advances Towards an Implantable Motor Cortical Interface* (PhD thesis). Salt Lake City, UT: University of Utah, 2001.

Taylor DM, Helms Tillery SI, and Schwartz AB. Direct cortical control of 3d neuroprosthetic devices. *Science* 296: 1829–1832, 2002.

Zhang K, Ginzburg I, McNaughton BL, and Sejnowski TJ. Interpreting neuronal population activity by reconstruction: unified framework with application to hippocampal place cells. *J Neurophysiol* 79: 1017–1044, 1998.