ARTICLE **Communicated by Jonathan Pillow**

Extracting Low-Dimensional Latent Structure from Time Series in the Presence of Delays

Karthik C. Lakshmanan
karthikl@cs.cmu.edu
Robotics Institute and Center for the Neural Basis of Cognition, Carnegie Mellon University, Pittsburgh, PA 15213, U.S.A.

Patrick T. Sadtler
patrick.t.sadtler@gmail.com
Department of Bioengineering, Center for the Neural Basis of Cognition, and Systems Neuroscience Institute, University of Pittsburgh, Pittsburgh, PA 15261, U.S.A.

Elizabeth C. Tyler-Kabara
Elizabeth.Tyler-Kabara@chp.edu
Department of Neurological Surgery, Department of Bioengineering, and Department of Physical Medicine and Rehabilitation, University of Pittsburgh, Pittsburgh, PA 15261, U.S.A.

Aaron P. Batista
apb10@pitt.edu
Department of Bioengineering, Center for the Neural Basis of Cognition, and Systems Neuroscience Institute, University of Pittsburgh, Pittsburgh, PA 15261, U.S.A.

Byron M. Yu
byronyu@cmu.edu
Department of Electrical Engineering and Computer Engineering, Department of Biomedical Engineering, and Center for the Neural Basis of Cognition, Carnegie Mellon University, Pittsburgh, PA 15213, U.S.A.

Noisy, high-dimensional time series observations can often be described by a set of low-dimensional latent variables. Commonly used methods to extract these latent variables typically assume instantaneous relationships between the latent and observed variables. In many physical systems, changes in the latent variables manifest as changes in the observed variables after time delays. Techniques that do not account for these delays can recover a larger number of latent variables than are present in the system, thereby making the latent representation more difficult to interpret. In this work, we introduce a novel probabilistic technique, time-delay gaussian-process factor analysis (TD-GPFA), that performs dimensionality reduction in the presence of a different time delay

Neural Computation **27, 1825–1856** (2015)
doi:10.1162/NECO_a_00759

between each pair of latent and observed variables. We demonstrate how using a gaussian process to model the evolution of each latent variable allows us to tractably learn these delays over a continuous domain. Additionally, we show how TD-GPFA combines temporal smoothing and dimensionality reduction into a common probabilistic framework. We present an expectation/conditional maximization either (ECME) algorithm to learn the model parameters. Our simulations demonstrate that when time delays are present, TD-GPFA is able to correctly identify these delays and recover the latent space. We then applied TD-GPFA to the activity of tens of neurons recorded simultaneously in the macaque motor cortex during a reaching task. TD-GPFA is able to better describe the neural activity using a more parsimonious latent space than GPFA, a method that has been used to interpret motor cortex data but does not account for time delays. More broadly, TD-GPFA can help to unravel the mechanisms underlying high-dimensional time series data by taking into account physical delays in the system.

1 Introduction

High-dimensional data are commonly seen in domains ranging from medicine to finance. In many cases, these high-dimensional data can be explained by a smaller number of explanatory variables. Dimensionality reduction techniques are a family of techniques that seek to recover these explanatory variables. Such techniques have been successfully applied to a wide variety of problems such as analyzing neural activity (Cunningham and Yu, 2014), modeling video textures (Siddiqi, Boots, & Gordon, 2007), economic forecasting (Aoki & Havenner, 1997), analyzing human gait (Omlor & Giese, 2011), seismic series analysis (Posadas et al., 1993), and face recognition (Turk & Pentland, 1991). For time series data, dimensionality-reduction techniques can capture the temporal evolution of the observed variables by a low-dimensional unobserved (or latent) driving process that provides a succinct explanation for the observed data.

The simplest class of dimensionality reduction techniques applied to time series data considers linear, instantaneous relationships between the latent variables and the high-dimensional observations. According to these techniques the ith observed variable $y^i(t)$, $(i = 1 \dots q)$ is a noisy linear combination of the p low-dimensional latent variables $x^j(t)$, $(j = 1 \dots p, p < q)$,

$$y^i(t) = \sum_{j=1}^{p} c_{i,j} x^j(t) + d^i + \varepsilon^i(t), \tag{1.1}$$

where $c_{i,j} \in \mathbb{R}$, $d^i \in \mathbb{R}$ and $\varepsilon^i(t) \in \mathbb{R}$ is additive noise. Many techniques based on latent dynamical systems (Roweis & Ghahramani, 1999), as well as

gaussian-process factor analysis (GPFA) (Yu et al., 2009), fall into this category. This implicit assumption of instantaneous relationships between the latent and observed variables is also made while applying static dimensionality reduction techniques to time series data. Examples of such techniques include principal component analysis (PCA) (Pearson, 1901), probabilistic principal component analysis (PPCA) (Roweis & Ghahramani, 1999; Tipping & Bishop, 1999), and factor analysis (FA) (Everitt, 1984).

Many real-world problems involve physical time delays whereby a change in a latent variable may not manifest as an instantaneous change in an observed variable. These time delays may be different for each latent-observed variable pair. For example, such time delays appear when a few sound sources are picked up by microphones placed at various locations in a room. Each sound source has a different delay to each microphone that depends on the distance between them (Morup, Madsen, & Hansen, 2007; Omlor & Giese, 2011). Other examples where such time delays appear include fMRI imaging, where these delays could arise from hemodynamic delays (Morup, Hansen, Arnfred, Lim, & Madsen, 2008) and human motion analysis, where the same control signal may drive muscles after different delays (Barliya, Omlor, Giese, & Flash, 2009). Dimensionality reduction algorithms that consider only instantaneous latent-observed variable relationships can explain these delays only by the introduction of additional latent variables. To see this, consider a conceptual illustration (see Figure 1) that shows three observed variables, each of which exhibits a single hill of activity but at different times. A model that ignores delays presents a more complicated picture of the underlying latent space, needing two latent variables to explain the observations (see Figure 1a). In contrast, a model that accounts for these time delays needs only a single latent variable to describe the observations (see Figure 1b). Because the goal of dimensionality reduction is to obtain the most parsimonious explanation of the observations, a latent space that is lower dimensional is more desirable.

We can extend the observation model in equation 1.1 to explicitly account for these delays,

$$y^i(t) = \sum_{j=1}^{p} c_{i,j} x^j(t - D_{i,j}) + d^i + \varepsilon^i(t), \tag{1.2}$$

where $c_{i,j}, d^i, \varepsilon^i(t) \in \mathbb{R}$. In equation 1.2, each observed variable is a linear combination of latent variables, with $c_{i,j}$ as the weights, such that there is a constant time delay $D_{i,j}$ between observed variable $y^i(t)$ and latent variable $x^j(t)$. $\varepsilon^i(t)$ is the observation noise. Equation 1.2 is known in the signal processing literature as the anechoic mixing model (Omlor & Giese, 2011). In this work, we present time-delay gaussian-process factor analysis (TD-GPFA), a novel technique that uses the observation model defined in

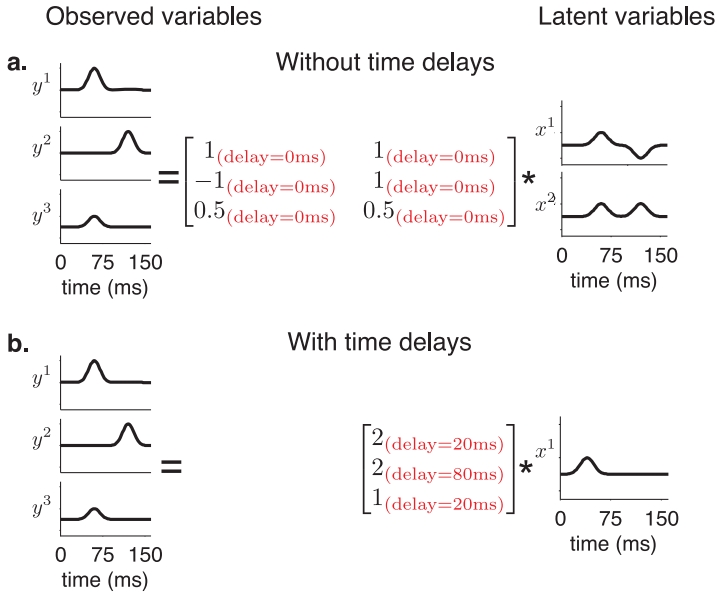

Figure 1: Conceptual illustration showing how dimensionality reduction algorithms that consider time delays can present a more compact view of the latent space. (a) Three observed variables y^1, y^2, and y^3 each shows a single hill of activity but at different times. If only instantaneous latent-observation relationships are considered, two latent variables are needed to explain these observed variables. (b) If nonzero time delays between latent and observed variables are considered, only one latent variable with a single hill of activity is needed to explain the same observed variables. The matrices show the delays as well as the weights that linearly combine the latent variables to reconstruct the observed variables.

equation 1.2 to capture time delays between latent and observed variables while also accounting for temporal dynamics in the latent space.

While the TD-GPFA framework is applicable in any domain with noisy, high-dimensional time series data where time delays may play a role, in this work we focus on neuroscience as the area of application. The field of neuroscience is witnessing a rapid development of recording technologies that promise to increase the number of simultaneously recorded neurons by orders of magnitude (Ahrens, Orger, Robson, Li, & Keller, 2013). Dimensionality-reduction methods provide an attractive way to study the neural mechanisms underlying various brain functions (Cunningham & Yu, 2014). In this context, the latent process can be thought of as a common input (Kulkarni and Paninski, 2007; Vidne et al., 2012) that drives the system, while the neurons are noisy sensors that reflect the time evolution of

this underlying process. However, this common input from a driving area in the brain may follow different pathways to reach each neuron. Physical conduction delays as well as different information processing times along each pathway may introduce different time delays from the driving area to each neuron. Previous work in neuroscience has focused on dimensionality-reduction techniques that do not consider time delays between latent variables and neurons. In this work, we investigate the effects of incorporating time delays into dimensionality reduction for neural data.

The rest of this article is structured as follows. In section 2 we briefly discuss related work and provide motivation for the TD-GPFA model. In section 3 we provide a formal mathematical description and describe how the model parameters can be learned using the expectation/conditional maximization either (ECME) algorithm (Liu & Rubin, 1994). In section 4 we demonstrate the benefits of TD-GPFA on simulated data and show that TD-GPFA is an attractive tool for the analysis of neural spike train recordings. Finally in section 5, we discuss our findings and mention possible avenues for future work.

2 Background and Related Work

Factor analysis (FA) (Everitt, 1984) is a commonly used dimensionality-reduction technique that has been applied across many domains ranging from psychology (Spearman, 1904) to neuroscience (Sadtler et al., 2014; Santhanam et al., 2009). FA is a linear dimensionality-reduction method that incorporates the concept of observation noise, whose variance can be different for each observed variable. An extension of FA, shifted factor analysis (SFA) (Harshman, Hong, & Lundy, 2003a), allows integer delays between latent and observed variables but requires an exhaustive integer search to identify the values of these delays (Harshman, Hong, & Lundy, 2003b). One way to avoid this combinatorial search over delays is to work in the frequency domain, since a shift by τ in time domain can be approximated by multiplication by $e^{-\imath\omega\tau}$ in the frequency domain, where $\imath = \sqrt{-1}$ and ω is the frequency. This property has been exploited to tackle the so-called anechoic blind source separation problem where the aim is to separate mixtures of sounds received at microphones that have a different delay to each source, assuming no echoes (anechoic). There has been a sizable body of work in the underdetermined case, where the number of sources is greater than or equal to the number of observed mixtures (Torkkola, 1996; Yeredor, 2003; Be'ery & Yeredor, 2008). The overdetermined case, where the number of sources is strictly less than the number of observed mixtures, is similar in flavor to dimensionality reduction and has received less attention. One approach uses an iterative procedure to perform joint diagonalization of the observations' cross spectra (Yeredor, 2001). Approaches based on alternating least squares that minimize errors in time and frequency

domain have been suggested (Morup et al., 2007, 2008). A framework derived from stochastic time-frequency analysis uses properties of the Wigner-Ville spectrum to tackle the blind source separation problem (Omlor & Giese, 2011).

These techniques do not explicitly consider temporal dynamics in the latent space. However, in many situations with time series data, we might have some a priori information about the latent process that is driving the observations. For example, while analyzing neural spike train recordings, we typically seek to extract latent variables that evolve smoothly in time (Smith & Brown, 2003; Kulkarni & Paninski, 2007; Yu et al., 2009). We could model the latent variables using a linear dynamical system, but the naive approach runs into the same problem as SFA, where the need for a combinatorial search over integer delays makes the problem intractable in the time domain.

Our novel technique, time-delay gaussian-process factor analysis (TD-GPFA), uses an independent gaussian process to model the temporal dynamics of each latent variable along with a linear-gaussian observation model. As shown in subsequent sections, this allows us to tractably learn delays over a continuous space while also combining smoothing and dimensionality reduction into a unified probabilistic framework. Our approach can be viewed as an extension of gaussian-process factor analysis (GPFA) (Yu et al., 2009) that introduces time delays between latent and observed variables.

3 Methods

3.1 TD-GPFA Model. Given q observed variables $\mathbf{y}(t) \in \mathbb{R}^{q\times 1}$ recorded at times $t = 1, 2, \ldots T$, we seek to identify p latent variables $\mathbf{x}(t) \in \mathbb{R}^{p\times 1}$ that provide a succinct explanation for these observations, where $p < q$.

The TD-GPFA model can be summarized as follows. Each observed variable y^i ($i = 1, \ldots, q$) is a linear combination of p latent variables x^j ($j = 1, \ldots, p$). The key innovation is the introduction of a time delay $D_{i,j}$ between the ith observed variable and the jth latent variable such that y^i reflects the value of x^j on a grid of T time points. Because each observed variable y^i can have a different time delay $D_{i,j}$ with respect to a latent variable x^j, these grids are shifted in time with respect to each other. Therefore, for q observed variables, the value of each latent variable must be estimated along q such grids (see Figure 2). We use an independent gaussian process for each latent variable to weave these qT points along a smooth function that represents the temporal evolution of the latent variable.

We specify a fixed time delay $D_{i,j}$ between the ith observed variable y^i and the jth latent variable x^j, such that $y^i(t)$ reflects the values of $x^j(t - D_{i,j})$. We then relate the latent and observed variables by the observation model

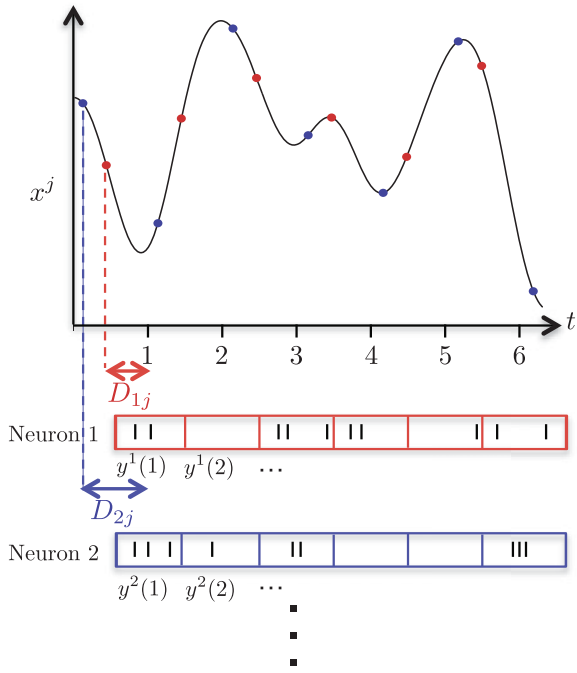

Figure 2: Illustration of the TD-GPFA observation model. y^1 and y^2 are two time series observed at times $t = 1, \ldots, 6$. The first observed variable y^1 reflects the values of the jth latent variable x^j at points on a grid (red dots) shifted by D_{1j}. The second observed variable y^2 reflects the values of x^j at points on a grid (blue dots) shifted by D_{2j}, and so on. For example, y^1 and y^2 could be binned spike counts of neurons, with the bins centered at times $t = 1, \ldots, 6$.

defined in equation 1.2 and specify the observation noise to be gaussian,

$$\varepsilon^i(t) \sim \mathcal{N}(0, R_{i,i}), \tag{3.1}$$

where $\varepsilon^i(t) \in \mathbb{R}$.

For convenience of notation, let y_t^i denote $y^i(t)$, and let $x_t^{(i,j)}$ denote $x^j(t - D_{i,j})$. In words, $x_t^{(i,j)}$ is the value of the jth latent variable that drives the ith observed variable at time t. We form the vector $\mathbf{x}_t^{(i,1:p)} \in \mathbb{R}^p$ by stacking the values of the p latent variables that drive the ith observed variable at time t. Then, collecting $c_{i,j}$ from equation 1.2 into a vector $\mathbf{c}_i \in \mathbb{R}^p$, we have

$$y_t^i = \mathbf{c}_i^T \mathbf{x}_t^{(i,1:p)} + d^i + \varepsilon_t^i. \tag{3.2}$$

We embed the vectors $\mathbf{c}_i$ in a matrix of zeros and collect the noise terms ε_t^i into a vector $\boldsymbol{\varepsilon}_t$. This allows us to write the observation model as a matrix product:

$$\begin{bmatrix} y_t^1 \\ y_t^2 \\ \vdots \\ y_t^q \end{bmatrix} = \begin{bmatrix} -\mathbf{c_1}^T- & & & \\ & -\mathbf{c_2}^T- & & \\ & & \ddots & \\ & & & -\mathbf{c_q}^T- \end{bmatrix} \begin{bmatrix} \mathbf{x}_t^{(1,1:p)} \\ \mathbf{x}_t^{(2,1:p)} \\ \vdots \\ \mathbf{x}_t^{(q,1:p)} \end{bmatrix} + \begin{bmatrix} d^1 \\ d^2 \\ \vdots \\ d^q \end{bmatrix} + \boldsymbol{\varepsilon}_t \tag{3.3}$$

$$\boldsymbol{\varepsilon}_t \sim \mathcal{N}(0, R). \tag{3.4}$$

We can then stack these vectors $\mathbf{x}_t^{(i,1:p)} \in \mathbb{R}^p$ for each observed variable $i = 1, \ldots, q$ to form a vector $\mathbf{x}_t^{(1:q,1:p)} \in \mathbb{R}^{pq}$. Equations 3.3 and 3.4 can then be compactly written as

$$\mathbf{y}_t | \mathbf{x}_t \sim \mathcal{N}(C\mathbf{x}_t^{(1:q,1:p)} + \mathbf{d}, R). \tag{3.5}$$

Here $C \in \mathbb{R}^{q \times pq}$ is a structured, sparse matrix that maps the latent variables onto the observed variables. The vector $\mathbf{d} \in \mathbb{R}^q$ is an offset term that captures the nonzero mean of the observed variables, where the ith element is d^i. As in FA we assume the noise covariance matrix $R \in \mathbb{R}^{q \times q}$ is diagonal, where the ith diagonal element is $R_{i,i}$ as defined in equation 1.2. We collect all the delays $D_{i,j}$ into a matrix $D \in \mathbb{R}^{q \times p}$. This completes our specification of the TD-GPFA observation model parameters C, $\mathbf{d}$, R, and D.

We now desire a way to describe latent variables that evolve continuously in time so that we can specify the elements of $\mathbf{x}_t^{(1:q,1:p)}$ for any real-valued setting of the delays in D. In addition, we seek a latent variable model that can capture the underlying temporal dynamics of the time series. To this end, we model the temporal dynamics of the jth latent variable, ($j = 1, 2, \ldots p$) by a separate gaussian process (GP) (Rasmussen & Williams, 2006):

$$\mathbf{x}_{1:T}^{(1:q,j)} \sim \mathcal{N}(0, K_j) \tag{3.6}$$

In equation 3.6, $\mathbf{x}_{1:T}^{(1:q,j)} \in \mathbb{R}^{qT}$ is a vector formed by stacking the vectors $\mathbf{x}_t^{(1:q,j)} \in \mathbb{R}^q$ for all times $t = 1, \ldots, T$. $K_j \in \mathbb{R}^{qT \times qT}$ is the covariance matrix for this GP. Choosing different forms for the covariance matrix provides different smoothing properties of the latent time courses. In this work, we construct the elements of K_j using the commonly used squared exponential (SE) covariance function (Rasmussen & Williams, 2006). Given two points

along the jth latent variable, $x_{t_1}^{(i_1,j)}$ and $x_{t_2}^{(i_2,j)}$:

$$k(\Delta t) = \sigma_{f,j}^2 \exp\left(-\frac{(\Delta t)^2}{2\tau_j^2}\right) + \sigma_{n,j}^2 \delta_{\Delta t}, \tag{3.7}$$

$$\text{where } \Delta t = (t_2 - D_{i_2,j}) - (t_1 - D_{i_1,j}). \tag{3.8}$$

The characteristic timescale $\tau_j \in \mathbb{R}_+$ specifies a degree of smoothness for the jth latent variable, while the delays $D_{i_1,j}$ and $D_{i_2,j}$, along with t_1 and t_2, specify the temporal locations of the pair of points. $\delta_{\Delta t}$ is the Kronecker delta, which is 1 if $\Delta t = 0$ and 0 otherwise. $\sigma_{f,j}^2 \in \mathbb{R}_+$ is the GP signal variance, while $\sigma_{n,j}^2 \in \mathbb{R}_+$ is the noise variance. The SE is an example of a stationary covariance; other stationary and nonstationary GP covariances (Rasmussen & Williams, 2006) can be seamlessly incorporated into this framework. This completes our specification of the TD-GPFA latent variable model.

3.2 Fitting the TD-GPFA Model. Given observations Y and a prescribed number of latent dimensions p, we seek to fit model parameters $\theta = \{C, \mathbf{d}, R, D, \tau_1, \ldots, \tau_p\}$. We initially sought to learn the model parameters using the expectation maximization (EM) algorithm (Dempster, Laird, & Rubin, 1977). The EM algorithm iteratively seeks model parameters θ that maximize the data likelihood $P(Y|\theta)$, by performing alternating expectation (E) steps and maximization (M) steps. The E-step updates the latents X keeping the model parameters fixed, by computing $P(X|Y)$ using the most recent parameter values. The M-step updates the model parameters by maximizing $f(\theta) = E[\log P(X, Y|\theta)]$ with respect to the parameters θ, where the expectation in $f(\theta)$ is taken with respect to the distribution $P(X|Y)$ found in the E-step. However, the delays in D moved very little in each iteration. This motivated us to consider a variant of the EM algorithm, known as the expectation/conditional maximization either (ECME) algorithm (Liu & Rubin, 1994). The ECME algorithm uses the same E-step as EM but replaces the M-step in each EM iteration with S conditional maximization (CM) steps. Each CM step maximizes either the expected joint probability $f(\theta)$ or directly the data likelihood $P(Y|\theta)$, with the constraint that some function of θ, say, $g_s(\theta)$, $s = 1, 2, \ldots, S$ remains fixed. Different choices for $g_s(\theta)$, and different choices of whether to optimize $f(\theta)$ or $P(Y|\theta)$ in each CM step result in different ECME algorithms. If the $g_s(\theta)$ span the θ space as described in Liu and Rubin (1994), then ECME guarantees monotone convergence of the data likelihood.

We now describe an ECME algorithm with two CM steps, where the first CM step maximizes the expected joint probability $f(\theta)$ over all parameters keeping the delays fixed, and the second CM step directly maximizes the data likelihood $P(Y|\theta)$ over the delays keeping all the other parameters

Algorithm 1: ECME Algorithm to Learn TD-GPFA Model Parameters.

Initialize θ^0;

$n \leftarrow 1$;

repeat

E-step: *Compute* $P(X|Y)$ *according to* θ^n;

CM-step 1: *Define* $g_1(\theta^n) = \{\boldsymbol{vec}(D^n)\}$ *and maximize*

$f(\theta^n) = E[\log P(X, Y|\theta^n)]$ *wrt* θ^n *keeping* $g_1(\theta^n)$ *fixed*;

CM-step 2: *Define* $g_2(\theta^n) = \{\boldsymbol{vec}(C^n), \mathbf{d}^n, \boldsymbol{diag}(R^n), \tau_1^n \ldots \tau_p^n\}$ *and*

maximize $L(\theta^n) = P(Y|\theta^n)$ *wrt* θ^n *keeping* $g_2(\theta^n)$ *fixed*;

$n \leftarrow n + 1$

until θ^n *has converged or* $n =$ *max number of iterations*;

fixed. This is outlined in algorithm 1. Here, the **vec** operator converts a matrix into a column vector, while the ***diag*** operator extracts the diagonal elements of a matrix into a column vector. Although it is often possible to maximize $P(Y|\theta)$ directly with respect to a subset of parameters while keeping all other parameters fixed (as is done here), this does not imply that it is possible to readily maximize $P(Y|\theta)$ with respect to all parameters jointly.

Because X and Y are jointly gaussian, the conditional in the E-step can be calculated in closed form. In CM step 1, we can find analytical solutions for C, $\mathbf{d}$, and R, while the timescales for each GP, $\tau_1, \ldots, \tau_p$, are estimated using an iterative gradient-based approach, for which the gradient can be computed analytically. In CM step 2, the delays in D are updated using an iterative gradient-based approach, where the gradient can be computed analytically as well. The mathematical details for the parameter updates are provided in appendix A. Partitioning the parameters θ in this way satisfies the requirements for valid $g_s(\theta)$ specified in equation 3.6 of Meng and Rubin (1993), and we are guaranteed monotone convergence to a local optimum of the data likelihood.

Under this formulation, the delays D are unconstrained. However, we may wish to constrain the time delays in D to lie within some desired

range. This can be motivated by two reasons: (1) in some applications, only delays that lie within a known range may be physically realizable, and we may desire to use this domain knowledge to constrain the delays learned by the model, and (2) if the delays are allowed to be arbitrarily large (e.g., much larger than the observation duration T), the model can use completely nonoverlapping segments from each latent variable to drive the observations. Since we want to identify latent variables that are shared by the observed variables, we need to enforce some overlap. We can constrain the delays to lie within a desired range by reparameterizing $D_{i,j}$ using a logistic function. If we want $-D_{\max} \leq D_{i,j} \leq D_{\max}$, we can define

$$D_{i,j} = D_{\max} \frac{(1 - e^{-D^*_{i,j}})}{(1 + e^{-D^*_{i,j}})}. \tag{3.9}$$

This reparameterization converts the constrained optimization for $D_{i,j}$ into an unconstrained optimization over $D^*_{i,j}$ using a change of variable. For the results presented in this article, we set $D_{\max} = \frac{T}{2}$.

The TD-GPFA model as defined above exhibits a degeneracy in the delays; shifting a latent variable x^j in time can be compensated for by appropriately subtracting out the shift from the delays between x^j and all observed variables (i.e., from all elements in the jth column of D). We temporally align all the latents to the first observed variable by fixing all elements in the first row of the delay matrix D at 0 during model fitting. The delays in D are therefore specified with respect to the first observed variable.

The scale of the latents as defined by the K_j in equation 3.6 is arbitrary, since any scaling of $x_t^{(i,j)}$ can be compensated by appropriately scaling $c_{i,j}$ such that the scale of y_t^i remains unchanged. Following Yu et al. (2009) and, by analogy to factor analysis, we fix the scale of the latents by fixing $k(0) = 1$, where $k(\Delta t)$ is defined in equation 3.7. This can be achieved by setting $\sigma^2_{f,j} = 1 - \sigma^2_{n,j}$, where $0 < \sigma^2_{n,j} \leq 1$ to ensure that $\sigma^2_{f,j}$ is nonnegative. We fixed $\sigma^2_{n,j}$ to a small quantity (10^{-6}), which allowed us to extract smooth latent time courses.

Since the ECME algorithm is iterative and converges to a local optimum, initialization of the model parameters is important. We initialized our model parameters using M-SISA, an extension of shift invariant subspace analysis (SISA) (Morup et al., 2007) that we developed to handle multiple time series where each time series may correspond, for example, to a different experimental trial. We chose to extend SISA as it identifies delays between latent and observed variables and has low computational demands for fitting model parameters. However, SISA does not take into account temporal dynamics in the latent space, which motivates the use of TD-GPFA. A detailed description of M-SISA is provided in appendix B. Another reasonable initialization would be to use the parameters fit by GPFA

and initialize all delays in D to 0. Although in this work we present results obtained by initializing with M-SISA, we found that in some cases, initializing with GPFA can yield higher cross-validated log likelihood. We describe the computational requirements of the ECME algorithm in appendix C.

3.3 Estimating Latent Dimensionality. We have thus far presented an algorithm to extract a given number of latent variables from high-dimensional observations. In practice, however, we do not know the true dimensionality of the latent space. In order to estimate the unknown latent dimensionality, we compute the cross-validated log likelihood (CV-LL) of the data for a range of candidate latent dimensionalities. Although in principle, we want to select the latent dimensionality at which the CV-LL curve peaks, the CV-LL curve in practice often begins to flatten at a lower latent dimensionality. Although the corresponding CV-LL is lower, it is often close to the peak, and this latent dimensionality can be selected to provide a more parsimonious representation of the data. Hence, we also report the latent dimensionality at which each CV-LL curve begins to flatten, which we define as the latent dimensionality at which the curve reaches 90% of its total height, where the height of each curve is the difference between its maximum and minimum values over the range of dimensionalities we tested. We call this point the elbow. In all subsequent sections, the CV-LL is computed using four-fold cross-validation.

3.4 Reach Task and Neural Recordings. All animal handling procedures were approved by the University of Pittsburgh Institutional Animal Care and Use Committee. We trained an adult male monkey on a center-out reaching task. At the beginning of each trial, a start target (circle, radius = 14 mm) appeared in the center of the work space. The monkey needed to acquire the start target with the center of the cursor (circle, radius = 15 mm) and hold the cursor within the start target for a duration that is randomly selected uniformly between 250 and 500 ms. Then one of eight possible peripheral targets appeared (circle, radius = 14 mm, 125 mm from the center of the work space, separated by 45 degrees). The monkey needed to move the cursor to the peripheral target and hold it there for a time randomly selected uniformly between 200 and 300 ms. The monkey received a liquid reward when he successfully held the cursor on the peripheral target. We monitored hand movements using an LED marker (PhaseSpace Inc.) on the hand contralateral to the recording array.

While the monkey was performing this reaching task, we recorded from the shoulder region of the primary motor cortex using a 96-channel electrode array (Blackrock Microsystems) as the monkey sat, head fixed, in a primate chair. The neural activity was recorded 10 days after array implantation. At the beginning of each session, we estimated the root-mean-square voltage of the signal on each electrode while the monkey sat calmly in a darkened room. We then set the spike threshold at 3.0 times the

root-mean-square value for each channel and extracted 1.24 ms waveform snippets (31 samples at 25 kHz) that began 0.28 ms (7 samples) before the threshold crossing. We manually sorted the waveforms recorded on each electrode using the waveforms themselves or the low-dimensional components that resulted from applying PCA to the waveforms. In this work, we used waveforms that were from well-isolated neurons.

We analyzed the activity of 45 simultaneously recorded neurons. For each experimental trial, we analyzed neural activity for a duration of 520 ms that started 100 ms before movement onset. We considered the first 700 successful trials of the experiment for which the movement times (i.e., from movement onset to movement end) were longer than 420 ms. For these trials, the mean movement time was 510 ms (standard deviation = 82 ms).

4 Results

We compared the performance of TD-GPFA to that of GPFA on two sets of simulated data, as well as on neural spike train recordings. GPFA was chosen as our benchmark for comparison for two reasons. First, the TD-GPFA and GPFA models are nominally identical and differ only in that the former accounts for time delays. This comparison therefore highlights the effects of accounting for time delays while analyzing neural data. Second, for the analysis of single-trial neural spike train data recorded from the motor cortex of a monkey during a reaching task, GPFA has been shown to outperform two-stage approaches that involve kernel smoothing spike trains and then applying PCA, PPCA, or FA (Yu et al., 2009). For both simulated and real data, we fit the TD-GPFA model by running our ECME algorithm for 200 iterations. GPFA was fit by running the EM algorithm described in Yu et al. (2009) for 200 iterations.

4.1 Simulated Data

4.1.1 Simulation 1. Simulation 1 follows directly from our conceptual illustration in Figure 1. We constructed a one-dimensional latent variable that exhibits a single hill of activity (see Figure 3a). The height of this hill of activity was different for each of 120 simulated trials, but its location in time was the same. We then generated 10 observed variables sampled at 20 ms intervals for a total duration of 600 ms for each trial. These observed variables were generated according to equation 3.5 such that each observed variable is a noisy, scaled, and temporally shifted version of the latent variable. The 10 delays in D were obtained by randomly drawing integers from $[-5, 5]$ and multiplying by 20 ms, the scale factors in C were all identically 5, and the offsets in $\mathbf{d}$ were all identically 2. All the diagonal elements of the observation noise matrix R were set to 0.5. We then evaluated the performance of TD-GPFA and GPFA on this data set.

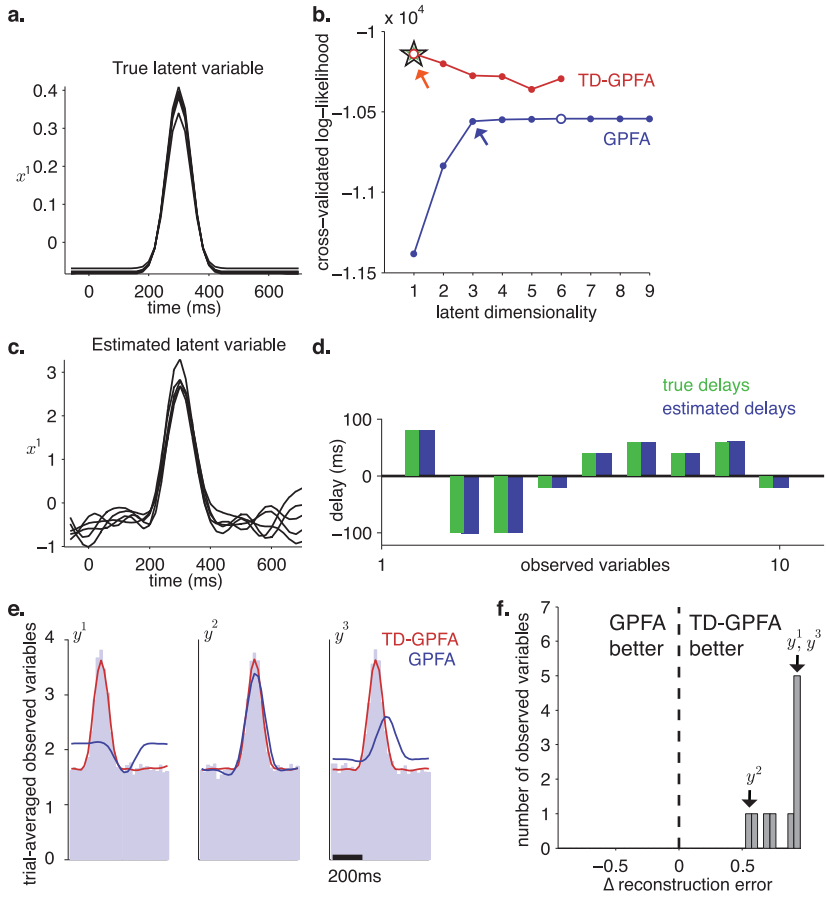

Figure 3: Simulation 1. (a) Simulated latent variable (five trials shown). (b) Cross-validated log-likelihood (CV-LL) plots versus latent dimensionality for TD-GPFA (red) and GPFA (blue). For each curve, the open circle indicates the peak, while the arrow indicates the elbow. The green star indicates the CV-LL obtained by initializing ECME for TD-GPFA with ground-truth parameters. (c) Latent variable estimated by TD-GPFA for latent dimensionality 1 (same five trials as shown in panel a). (d) Ground truth (green) and estimated (blue) delays from the latent variable to all observed variables. (e) Comparison of trial-averaged observed time courses for three representative observed variables (light blue) with those predicted by TD-GPFA (red) and GPFA (blue), with a single latent variable. (f) Histogram of the normalized difference between GPFA and TD-GPFA reconstruction errors, computed separately for each observed variable. Observed variables to the right of 0 are better reconstructed by TD-GPFA, while those to the left of 0 are better reconstructed by GPFA.

We computed four-fold cross-validated log likelihood (CV-LL) for a range of candidate latent dimensionalities for both TD-GPFA and GPFA. For each method, we evaluated performance by comparing the peak values of the CV-LL curves (see Figure 3b). We found that the CV-LL curve for TD-GPFA has a higher peak (red open circle) than the corresponding peak for GPFA (blue open circle). Next, we compared the compactness of the latent space required by each method to describe the data. In this simulation, the CV-LL curve for GPFA has an elbow at three dimensions (blue arrow) and reaches its maximum value at six dimensions (blue unfilled circle). However, when we account for delays, we are able to find a more compact model that describes the data better. This claim is supported by the following two findings: (1) the CV-LL curve for TD-GPFA (red) attains its maximum value at one latent dimension (red open circle), which agrees with the true latent dimensionality, and (2) the maximum CV-LL for TD-GPFA is higher than that of GPFA. Next, because ECME guarantees convergence only to a local optimum, we sought to evaluate how good the local optimum found by initializing ECME with M-SISA is. Since our simulated latent variables were not drawn from the TD-GPFA latent variable model (see equation 3.6), we did not have ground-truth values for the timescales and only knew C up to scale. Therefore, we could not directly evaluate CV-LL with the parameters used to simulate the data. Instead, we performed this comparison by running our ECME algorithm for 100 iterations initializing C, $\mathbf{d}$, D, and R at their simulated values and initializing the timescales τ at 300 ms. We then computed the corresponding CV-LL (green star). We found that this CV-LL is close to the CV-LL attained by our algorithm (red open circle on top of the green star).

The latent variable recovered by TD-GPFA (see Figure 3c) captures the single hill of activity seen in the simulated latent variable (see Figure 3a), even though it is determined only up to scale. There are 10 delays for the latent variable, 1 to each observed variable (see Figure 3d). The estimated delays are in close agreement with the true delays, with a mean absolute error of 0.67 ms (standard deviation = 0.53 ms).

While the CV-LL curves provide a quantitative comparison of the two algorithms, we also sought to illustrate the benefit of incorporating delays into the dimensionality reduction framework. To this end, we compared the abilities of TD-GPFA and GPFA to capture the time course of the observations using the same latent dimensionality. For each method, we used the model parameters estimated from each training fold and the recovered latent variable, $E[X|Y]$ from the corresponding validation folds. We then used the corresponding observation models (see equation 3.5 for TD-GPFA, and equation 1.1 for GPFA) to reconstruct the high-dimensional observed variables. We show these reconstructions averaged across all trials for three representative observed variables, y^1, y^2, and y^3, plotted on top of their actual trial averaged values (see Figure 3e). Note that GPFA (blue) misses the hill of activity for y^1 and gets the location of the hill wrong for y^3,

while TD-GPFA (red) is in close agreement with the actual trial average. Next, we asked if the reconstruction provided by TD-GPFA agrees better with the observed time courses across all the observed variables. For each of the four cross-validation folds, we calculated the reconstruction error for each observed variable by computing the sum of squared errors between the observed and reconstructed values at each time point and then averaging across trials. We then averaged this reconstruction error across the four folds. To interpret the difference in reconstruction errors between TD-GPFA and GPFA, we computed a normalized difference in reconstruction error for each observed variable by subtracting the reconstruction error of TD-GPFA from the reconstruction error of GPFA and then dividing by the reconstruction error of GPFA (see Figure 3e). A normalized difference in reconstruction error equal to 0 implies that TD-GPFA provides no improvement over GPFA, while a value of 1 implies that TD-GPFA provides perfect reconstruction. A negative value indicates that GPFA is better than TD-GPFA at reconstructing the observed variable. For this simulation, the normalized difference in reconstruction error is positive for all 10 observed variables (0.82 ± 0.16, mean $\pm$ standard deviation), which means that TD-GPFA is better able to capture the time course for every observed variable.

4.1.2 Simulation 2. Our second simulation takes us a step closer to the neuroscience application, where we posit that neural population activity can be explained by latent variables that drive the neurons after different delays. To simulate a data set with known delays, we first applied GPFA to extract latents from real neural data, then artificially introduced delays and constructed observed variables from these latents according to the TD-GPFA observation model (see equation 3.5). We considered 140 experimental trials of neural activity of 45 neurons, where the monkey made movements toward one reach target. We applied GPFA to this data and extracted two latent dimensions (see Figure 4a). Next, we generated TD-GPFA observation model parameters to simulate 45 observed variables. The delays D were integers randomly chosen from $[-5, 5]$ and multiplied by 20 ms. The scale factors in C were integers randomly chosen from $[1, 5]$. All elements of $\mathbf{d}$ were set to 2, and the observation noise matrix R was the identity matrix. Using these parameters and the TD-GPFA observation model (see equation 3.5), we then constructed 140 trials from the latent variables extracted from GPFA. Each trial was 520 ms, discretized into 26 time steps at intervals of 20 ms.

We found that by accounting for delays, TD-GPFA describes the data using a smaller number of latent variables than GPFA (see Figure 4b). The CV-LL curve for GPFA (blue) exhibits an elbow at 5 latent dimensions (blue arrow) and reaches its maximum at 12 latent dimensions (blue open circle). In contrast, the CV-LL curve for TD-GPFA (red) has an elbow at 2 latent dimensions (red arrow) where 2 is the true latent dimensionality,

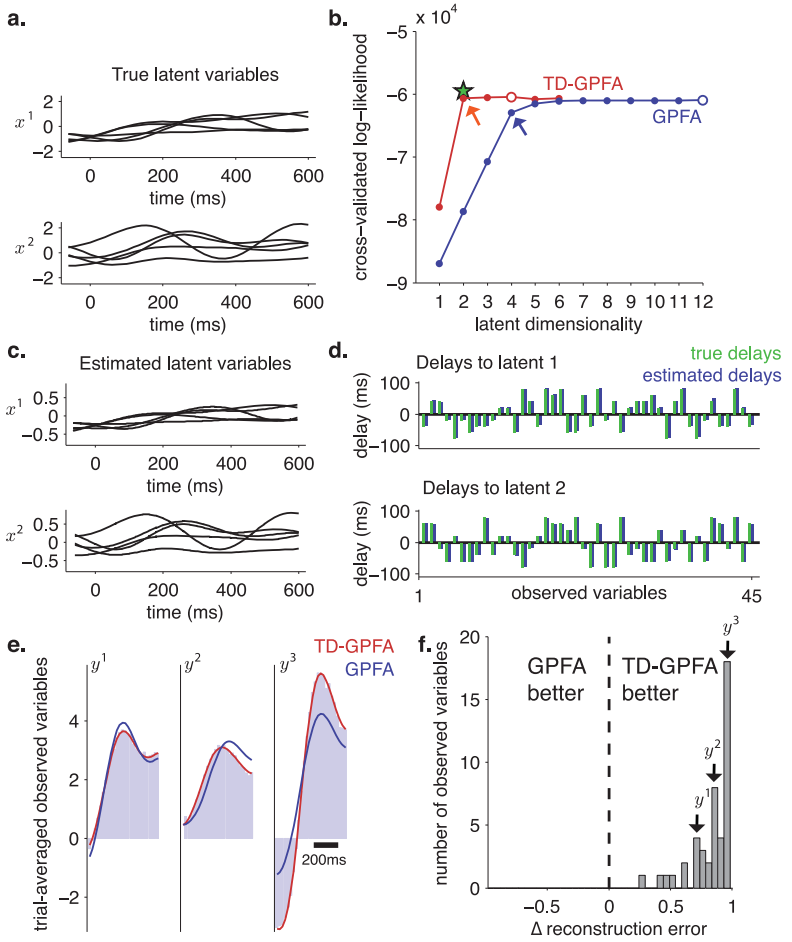

Figure 4: Simulation 2. (a) Simulated latent variables (5 trials shown). (b) Cross-validated log-likelihood (CV-LL) plots versus latent dimensionality for TD-GPFA (red) and GPFA (blue). For each curve, the open circle indicates the peak, and the arrow indicates the elbow. The green star indicates CV-LL obtained by initializing ECME for TD-GPFA with ground-truth parameters. (c) Latent variables estimated by TD-GPFA for latent dimensionality 2 (same five trials as shown in panel a). (d) Ground-truth (green) and estimated (blue) delays from each latent variable to all observed variables. (e) Comparison of trial-averaged observed time courses for three representative observed variables (light blue) with those predicted by TD-GPFA (red) and GPFA (blue), with a latent dimensionality of 2. (f) Histogram of the normalized difference between GPFA and TD-GPFA reconstruction errors, computed separately for each observed variable. Observed variables to the right of 0 are better reconstructed by TD-GPFA, while those to the left of 0 are better reconstructed by GPFA.

although the peak is attained at 4 latent dimensions (red open circle). As in simulation 1, the maximum CV-LL attained by TD-GPFA is higher than that of GPFA. Next, we ran ECME for 100 iterations initializing C, $\mathbf{d}$, D, and R at their ground-truth values and the time scales τ at the values estimated by GPFA for the simulated latents. We found that the CV-LL attained by TD-GPFA for 2 latent variables is close to the CV-LL obtained by this initialization (green star). The latent variables recovered by TD-GPFA for latent dimensionality 2 show time courses that closely resemble those of the simulated latents (see Figure 4c), even though the scale of the latent variables is different. Each latent variable has 45 delays, one to each observed variable (see Figure 4d). The estimated delays are again very close to the true delays, with a mean absolute error of 1.84 ms (standard deviation = 1.79 ms).

As in simulation 1, we show trial-averaged reconstructions achieved by each method using two latent variables for three representative observed variables y^1, y^2, and y^3 (see Figure 4e). Once again we see that TD-GPFA is able to capture the empirical trial average well. GPFA, however, misses out on key features of the activity time course. We then asked if this is true across all observed variables by computing the normalized difference in reconstruction error between TD-GPFA and GPFA (see Figure 4f). We found that this difference is positive across all observed variables (0.84 ± 0.17, mean $\pm$ standard deviation), indicating that TD-GPFA reconstructed all 45 observed time courses better than GPFA.

4.2 Neural Spike Train Data. We applied TD-GPFA and GPFA to neural spike train recordings obtained as described in section 3.4. Our observations were spike counts taken in nonoverlapping 20 ms bins and then square-root-transformed to stabilize the variance of the spike counts (Yu et al., 2009).

We compared the CV-LL versus latent dimensionality curve of TD-GPFA with that of GPFA (see Figure 5a). The CV-LL curve of TD-GPFA is consistently higher than that of GPFA. Furthermore, with as few as four latent dimensions (red dotted line), TD-GPFA finds a better description of the data than GPFA with any latent dimensionality up to 10. This is the main result of this work: that TD-GPFA finds a model that is more compact yet can describe the neural recordings better than GPFA.

We present a closer look at the model parameters returned by TD-GPFA with five latent dimensions, which is the latent dimensionality at the elbow of the CV-LL curve. We found that most of the recovered delays are nonzero (see Figure 5b). This indicates that TD-GPFA makes use of time delays to obtain a more compact yet better description of the neural activity. As in our simulations, we sought to illustrate the benefit of incorporating delays in capturing the neural activity. We show trial-averaged reconstructions using five latent variables for three representative neurons (y^1, y^2, and y^3) plotted on top of the empirical trial average (see Figure 5c). TD-GPFA (red) is better able to capture the firing rate profiles of neurons y^1 and

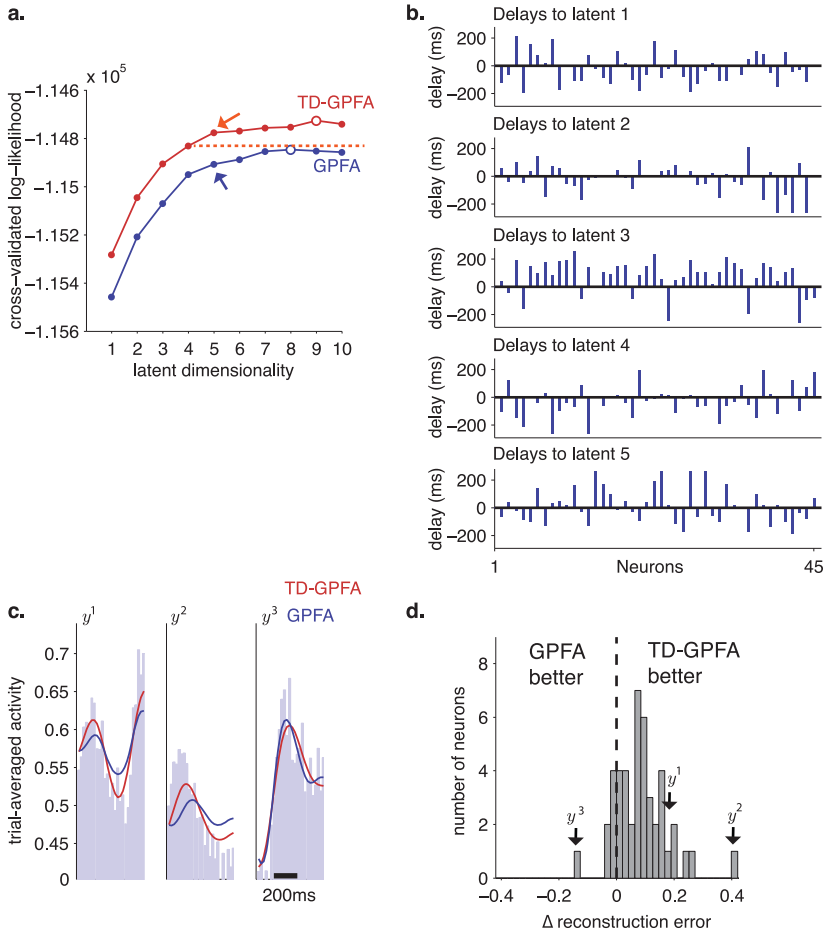

Figure 5: TD-GPFA applied to neural activity recorded in the motor cortex. (a) Cross-validated log-likelihood (CV-LL) plots versus latent dimensionality for TD-GPFA (red) and GPFA (blue). For each curve, the open circle indicates the peak, and the arrow indicates the elbow. The red dotted line is the CV-LL of the TD-GPFA model with four latent dimensions. (b) Estimated delays from each latent variable to all neurons. (c) Comparison of empirical firing rate histograms (light blue) to firing rate histograms predicted by TD-GPFA (red) and GPFA (blue) for three representative neurons. The vertical axis has units of square-rooted spike counts taken in a 20 ms bin. Five latent variables were used for both TD-GPFA and GPFA. (d) Histogram of the normalized difference between GPFA and TD-GPFA reconstruction errors, computed separately for each neuron. Neurons to the right of 0 are better reconstructed by TD-GPFA, while those to the left of 0 are better reconstructed by GPFA.

y^2 as compared to GPFA (blue), which tends to underestimate the hills of activity. We also show a neuron (y^3) for which GPFA is better than TD-GPFA at capturing the trial-averaged activity. Finally, to compare TD-GPFA and GPFA for all neurons, we computed the normalized difference in reconstruction error (see Figure 5d). We found that this difference is positive for 38 out of 45 neurons (0.09 ± 0.09, mean $\pm$ standard deviation). Thus, TD-GPFA provided overall better reconstructions than GPFA (paired t-test, $P < 0.001$). In addition, we verified that TD-GPFA with five latent dimensions is able to provide better reconstructions than GPFA with 6 to 10 latent dimensions, consistent with our findings from the CV-LL curves (see Figure 5a).

The time courses of the estimated latent variables (see Figure 6) can be interpreted as time courses of latent drivers that influence neurons at different time delays. Although we leave a detailed analysis of the estimated latent variables for future scientific studies, we can verify here that the extracted time courses of the latent variables vary systematically with reach target, as would be expected from neurons with smooth tuning curves. For example, the first latent variable (x^1) shows a pronounced peak for rightward reaches, and this peak becomes less pronounced for directions that are farther from the right. Similarly, the fifth latent variable (x^5) shows a pronounced peak for upward reaches. Note that while the shapes of the latent time courses are meaningful, the absolute position in time for each latent variable is arbitrary. Intuitively, if we have a number of observed variables that reflect a latent driver after different delays, we can determine the delays of the observed variables only with respect to each other, but not the absolute delay between the latent driver and the observed variables. Thus, a latent variable (i.e., all the time courses for the latent variable) can be shifted arbitrarily forward or backward in time by correspondingly adding or subtracting the shift to all the delay values of that latent variable in D.

5 Discussion

In this work we have presented a novel method (TD-GPFA) that can extract low-dimensional latent structure from noisy, high-dimensional time series data in the presence of a fixed time delay between each latent and observation variable. TD-GPFA unifies temporal smoothing and dimensionality reduction in a common probabilistic framework, using gaussian processes to capture temporal dynamics in the latent space. We then presented an ECME algorithm that learns TD-GPFA model parameters from data. Next, we verified using simulated data that TD-GPFA is able to recover the time delays and the correct dimensionality of the latent space, while GPFA, which is nominally identical but does not consider delays, needs a latent space of higher dimensionality to explain the same observations. Finally, we applied TD-GPFA to neural activity recorded in the motor cortex during a

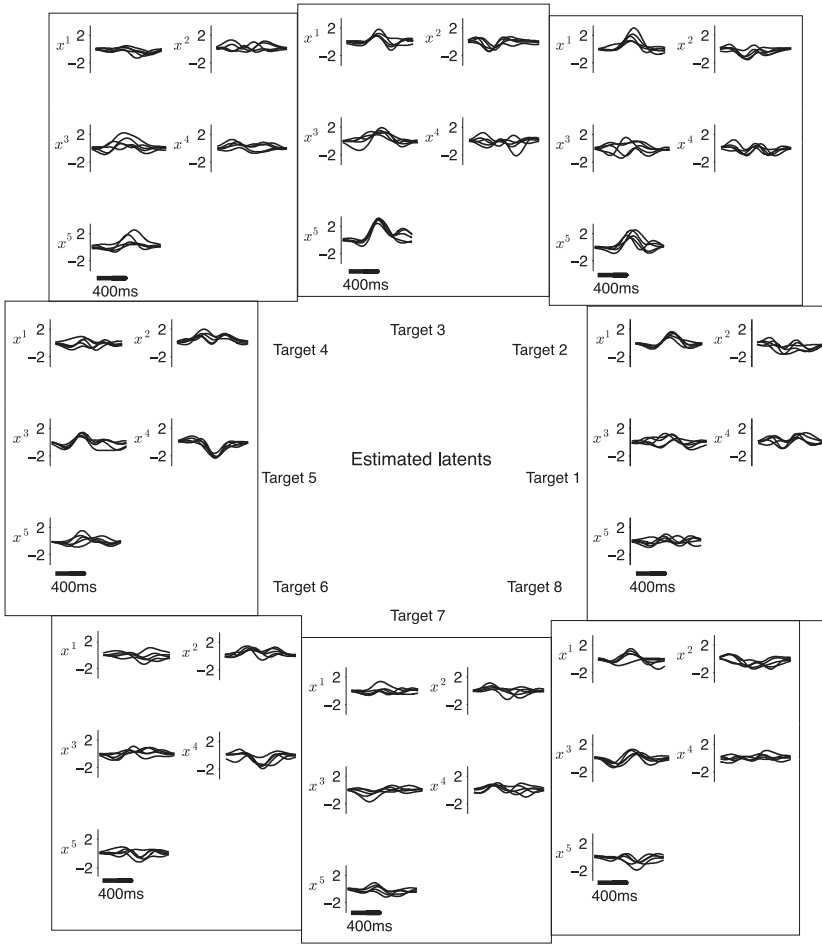

Figure 6: Latent variable time courses extracted by TD-GPFA from neural activity recorded in the motor cortex. Results shown for latent dimensionality five and plotted separately for each latent dimension (x^1, x^2, x^3, x^4 and x^5) and each reach target (five trials shown per target).

center-out reaching task. We found that (1) the 45-dimensional neural activity could be succinctly captured by TD-GPFA using five latent dimensions, and (2) TD-GPFA finds a more compact description of the latent space that better describes the neural activity, as compared to GPFA.

Modeling temporal dynamics of the latent variables using gaussian processes (GPs) offers us the following advantages. First, using GPs allows us to tractably learn time delays over a continuous domain. In our

formulation, the delays appear only in the GP covariance kernel. During parameter learning using the ECME algorithm, these delays can be optimized using any gradient descent method. Second, GPs offer an analytically tractable yet expressive way to model time series in cases where the true latent dynamics are unknown, as is often the case in exploratory data analysis. This is because we need only to specify the covariance kernel for each GP. The extracted latent variables can guide further parametric modeling of latent dynamics, such as using a dynamical system (Smith & Brown, 2003; Yu et al., 2006; Kulkarni & Paninski, 2007; Macke et al., 2011; Petreska et al., 2011; Pfau, Pnevmatikakis, & Paninski, 2013). Third, the GPs smooth the latent dynamics according to the specified covariance kernel and can be thought of as performing regularization on the latent space, which is helpful in analyzing time series data (Rahimi, Recht, & Darrell, 2005). Fourth, our approach allows us to combine smoothing of the noisy time series with dimensionality reduction in a unified probabilistic framework. This has been shown to be advantageous in analyzing neural spike train data, as opposed to previous methods that smooth the data as a preprocessing step using a kernel of fixed width (Yu et al., 2009). The key insight is that performing smoothing as a preprocessing step assumes a single timescale of evolution of neural activity, while our approach allows each latent variable to evolve with its own timescale. This benefit may translate well to other domains as well, where underlying latent variables evolve with different timescales.

In neuroscience, dimensionality reduction methods have proven useful in studying various neural systems, including the olfactory system (Mazor & Laurent, 2005; Broome, Jayaraman, & Laurent, 2006; Saha et al., 2013), the motor system (Briggman, Abarbanel, & Kristan, 2005; Yu et al., 2009; Churchland et al., 2012; Sadtler et al., 2014), decision making (Harvey, Coen, & Tank, 2012; Mante, Sussillo, Shenoy, & Newsome, 2013) and working memory (Machens, Romo, & Brody, 2010; Rigotti et al., 2013). However, these and other previous studies do not consider delays between latent variables and neurons. Because information in the brain can flow across multiple pathways to reach the recorded neurons, physical conduction delays as well as differences in the information processing times along each pathway can give rise to delays. For example, a latent process originating in the visual cortex may influence some motor cortical neurons (M1) through parietal cortex but may also influence some M1 neurons through a longer path involving parietal cortex and dorsal premotor cortex. When we applied TD-GPFA to M1 neurons recorded during a reaching task, TD-GPFA recovered different nonzero delays, which is suggestive of latent drivers influencing neurons at different delays. Future experiments are needed to establish the connection between latent variables and upstream brain areas. Another property of the TD-GPFA framework that makes it suitable to application in neuroscience is its ability to perform single-trial analysis. If neural activity varies substantially on nominally identical trials, common approaches like averaging responses across trials may mask the true

underlying processes that generate these responses. TD-GPFA allows us to leverage the statistical power of recording from an entire neural population to extract a succinct summary of the activity on a trial-by-trial basis.

While our metrics show that the description of the neural activity extracted by TD-GPFA is more accurate than one extracted without considering delays, it remains to be seen whether these delays correspond to actual neural processes and, if so, to understand their physical underpinnings. In addition, TD-GPFA may provide an attractive tool for analyzing data recorded simultaneously from multiple brain areas, where a driving process may manifest in neurons belonging to each area after different delays (Semedo, Zandvakili, Kohn, Machens, & Yu, 2014).

Several extensions to the TD-GPFA methodology can be envisaged. It may be possible to allow each latent variable to drive an observation at multiple delays, which might be a more realistic assumption for some systems; allow nonlinear relationships between latent and observed variables; or use nonstationary GP covariances. Software for TD-GPFA is available at http://bit.ly/1HAjyBl.

Appendix A: ECME Update Equations

Given observations $\mathbf{y}_{1:T}$ and a prescribed number of latent dimensions $p < q$, we seek to fit model parameters $\theta = \{C, \mathbf{d}, R, D, \tau_1, \ldots, \tau_p\}$. We first define $\tilde{\mathbf{y}} \in \mathbb{R}^{qT}$ and $\tilde{\mathbf{x}} \in \mathbb{R}^{pqT}$ by stacking observed variables $\mathbf{y}_t$ and latent variables $\mathbf{x}_t^{(1:q,1:p)}$ (as defined in equations 3.3 to 3.5) across all time points:

$$\tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_T \end{bmatrix}, \tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{x}_1^{(1:q,1:p)} \\ \vdots \\ \mathbf{x}_T^{(1:q,1:p)} \end{bmatrix}. \tag{A.1}$$

This allows us to write the latent and observation variable models across all time as follows:

$$\tilde{\mathbf{x}} \sim \mathcal{N}(\mathbf{0}, \tilde{K}), \tag{A.2}$$

$$\tilde{\mathbf{y}}|\tilde{\mathbf{x}} \sim \mathcal{N}(\tilde{C}\tilde{\mathbf{x}} + \tilde{\mathbf{d}}, \tilde{R}), \tag{A.3}$$

where $\tilde{C} \in \mathbb{R}^{qT \times pqT}$ and $\tilde{R} \in \mathbb{R}^{qT \times qT}$ are block diagonal matrices comprising T blocks of C and R, respectively. $\tilde{\mathbf{d}} \in \mathbb{R}^{qT}$ is constructed by stacking T copies of $\mathbf{d}$. The elements of $\tilde{K} \in \mathbb{R}^{pqT \times pqT}$ can be computed using the covariance kernel defined in equation 3.7. From Equations A.2 and A.3, we can write the joint distribution of $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$ as

$$\begin{bmatrix} \tilde{\mathbf{x}} \\ \tilde{\mathbf{y}} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ \tilde{\mathbf{d}} \end{bmatrix}, \begin{bmatrix} \tilde{K} & \tilde{K}\tilde{C}^T \\ \tilde{C}\tilde{K} & \tilde{C}\tilde{K}\tilde{C}^T + \tilde{R} \end{bmatrix} \right). \tag{A.4}$$

A.1 E-Step. Using the basic result of conditioning for jointly gaussian random variables,

$$\tilde{\mathbf{x}}|\tilde{\mathbf{y}} \sim \mathcal{N}(\tilde{K}\tilde{C}^T(\tilde{C}\tilde{K}\tilde{C}^T + \tilde{R})^{-1}(\tilde{\mathbf{y}} - \tilde{\mathbf{d}}), \tilde{K} - \tilde{K}\tilde{C}^T(\tilde{C}\tilde{K}\tilde{C}^T + \tilde{R})^{-1}\tilde{C}\tilde{K}). \tag{A.5}$$

Thus, the extracted latent variables are

$$E[\tilde{\mathbf{x}}|\tilde{\mathbf{y}}] = \tilde{K}\tilde{C}^T(\tilde{C}\tilde{K}\tilde{C}^T + \tilde{R})^{-1}(\tilde{\mathbf{y}} - \tilde{\mathbf{d}}). \tag{A.6}$$

A.2 CM Step 1. In CM step 1 we maximize $f(\theta) = E[\log P(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}|\theta)]$ with respect to $\{C, D, R, \gamma_1 \ldots \gamma_p\}$ keeping $g_1(\theta) = \{\mathbf{vec}(D)\}$ fixed. The expectation in $f(\theta)$ is taken with respect to the distribution $P(\tilde{\mathbf{x}}|\tilde{\mathbf{y}})$ found in equation A.5. Although this is a joint optimization with respect to all the parameters, their optimal values are dependent on only a few or none of the other parameters.

We first define the following notation. Given a vector $\mathbf{v}$,

$$\langle \mathbf{v} \rangle = E[\mathbf{v}|\tilde{\mathbf{y}}], \tag{A.7}$$

$$\langle \mathbf{v}\mathbf{v}^T \rangle = E[\mathbf{v}\mathbf{v}^T|\tilde{\mathbf{y}}], \tag{A.8}$$

where the expectations can be found from equation A.5.

C, **d update:** Maximizing $f(\theta)$ with respect to C and $\mathbf{d}$ yields joint updates for each row of C and the corresponding element of $\mathbf{d}$, such that for $i \in 1, 2, \ldots, q$,

$$[\mathbf{c}_i^T \quad d^i] = \left(\sum_{t=1}^{T} y_t^i \left[\langle \mathbf{x}_t^{(i,1:p)} \rangle^T \quad 1 \right] \right) \left(\sum_{t=1}^{T} \begin{bmatrix} \langle \mathbf{x}_t^{(i,1:p)} \mathbf{x}_t^{(i,1:p)^T} \rangle & \langle \mathbf{x}_t^{(i,1:p)} \rangle \\ \langle \mathbf{x}_t^{(i,1:p)} \rangle^T & 1 \end{bmatrix} \right)^{-1}. \tag{A.9}$$

***R* update:** Maximizing $f(\theta)$ with respect to R yields

$$R = \frac{1}{T}\mathbf{diag}\left\{ \sum_{t=1}^{T} ((\mathbf{y}_t - \mathbf{d})(\mathbf{y}_t - \mathbf{d})^T - (\mathbf{y}_t - \mathbf{d})\langle \mathbf{x}_t \rangle^T C^T - C\langle \mathbf{x}_t \rangle (\mathbf{y}_t - \mathbf{d})^T + C\langle \mathbf{x}_t \mathbf{x}_t^T \rangle C^T) \right\}. \tag{A.10}$$

The updated C, $\mathbf{d}$ found from equation A.9 should be used in equation A.10.

$\tau_1 \ldots \tau_p$ **update:** Although there is no analytic form of the update equations for the timescales, we can learn them using any gradient optimization technique,

$$\frac{\partial f(\theta)}{\partial \tau_j} = \mathbf{tr}\left(\left(\frac{df(\theta)}{\partial K_j}\right)^T \frac{\partial K_j}{d\tau_j}\right), \tag{A.11}$$

where

$$\frac{\partial f(\theta)}{\partial K_j} = -\frac{1}{2}K_j^{-1} + \frac{1}{2}(K_j^{-1}\langle \mathbf{x}_{1:T}^{(1:q,j)} \mathbf{x}_{1:T}^{(1:q,j)^T}\rangle K_j^{-1}), \tag{A.12}$$

$$\frac{\partial k_j(\Delta t)}{\partial \tau_j} = \sigma_{f,j}^2 \frac{(\Delta t)^2}{\tau_j^3} \exp\left(-\frac{(\Delta t)^2}{2\tau_j^2}\right). \tag{A.13}$$

This is a constrained optimization problem because the GP timescales need to be positive. We can turn this into an unconstrained optimization problem by optimizing with respect to $\log \tau_j$ using a change of variable.

A.3 CM Step 2. In CM step 2 we directly optimize the log likelihood $L(\theta) = P(\tilde{\mathbf{y}}|\theta)$ with respect to each $D_{i,j}$. $P(\tilde{\mathbf{y}}|\theta)$ can be computed easily since

$$\tilde{\mathbf{y}} \sim N(\tilde{\mathbf{d}}, \tilde{C}\tilde{K}\tilde{C}^T + \tilde{R}). \tag{A.14}$$

The delays, like the GP timescales, can also be learned using any gradient optimization technique. If $\Sigma = \tilde{C}\tilde{K}\tilde{C}^T + \tilde{R}$,

$$\frac{\partial L(\theta)}{\partial D_{i,j}} = \mathbf{tr}\left(\left(\frac{\partial L(\theta)}{\partial \Sigma}\right)^T \frac{\partial \Sigma}{\partial D_{i,j}}\right), \tag{A.15}$$

where

$$\frac{\partial L(\theta)}{\partial \Sigma} = -\frac{1}{2}\Sigma^{-1} + \frac{1}{2}(\Sigma^{-1}(\tilde{\mathbf{y}} - \tilde{\mathbf{d}})(\tilde{\mathbf{y}} - \tilde{\mathbf{d}})^T\Sigma^{-1}), \tag{A.16}$$

$$\frac{\partial \Sigma}{\partial D_{i,j}} = \tilde{C}\frac{\partial \tilde{K}}{\partial D_{i,j}}\tilde{C}^T, \tag{A.17}$$

$$\frac{\partial k_j(\Delta t)}{\partial D_{i,j}} = \sigma_{f,j}^2 \frac{\Delta t}{\tau_j^2} \exp\left(-\frac{(\Delta t)^2}{2\tau_j^2}\right) \frac{\partial(\Delta t)}{\partial D_{i,j}}, \tag{A.18}$$

$$\frac{\partial(\Delta t)}{\partial D_{i,j}} = \begin{cases} 1 & \text{if } i = i_2 \\ -1 & \text{if } i = i_1 \end{cases}, \tag{A.19}$$

where $\Delta t, i_1, i_2$ are defined in equation 3.8.

The derivations in this section are presented for a single time series (corresponding to a single experimental trial) of duration T. In many cases, we wish to learn the model given multiple time series, each of which may have a different T. It is straightforward to extend equations A.5 to A.15 to account for N time series by considering $\partial[\sum_{n=1}^{N} f_n(\theta)]/\partial\theta$ instead of $\partial f(\theta)/\partial\theta$. This assumes that each time series is independent given the model parameters, meaning that we do not explicitly constrain the latent variables to follow similar time courses on different trials. However, the latent variables are assumed to all lie within the same low-dimensional state-space with the same timescales and delays.

Appendix B: Parameter Initialization for ECME

Shift invariant subspace analysis (SISA) (Morup et al., 2007) is a technique developed to address the anechoic blind source separation problem. It uses the fact that a delay $D_{i,j}$ in the time domain can be approximated by multiplication by the complex coefficients $e^{-\imath\omega D_{i,j}}$ in the frequency domain, where $\imath = \sqrt{-1}$ and $\omega = 2\pi\frac{k-1}{T}$, where $k = 1, \ldots, T$ and T is the number of time steps in the time series. While SISA was originally designed to handle a single time series, in some applications, observations are collected as multiple time series as opposed to a single long time series, as in the neural application considered in this work. We developed an extension M-SISA that can handle multiple time series.

If Y^i is the Fourier transform of $\mathbf{y}^i$, we can rewrite the anechoic mixing model as a matrix product in the frequency domain. We first center our observed variables at mean $\mathbf{0}$ by subtracting out the mean computed across all time points in all time series. For convenience of notation, in the rest of this section, $\mathbf{y}$ will denote the mean $\mathbf{0}$ observed variables.

Then for each time series, we have

$$y_t^i = \sum_{j=1}^{p} c_{i,j} x_{t-D_{i,j}}^j + \varepsilon_t^i, \tag{B.1}$$

where $i \in 1 \ldots q,\ j \in 1 \ldots p$. Taking the Fourier transform of each side yields

$$Y_k^i = \sum_{j=1}^{p} c_{i,j} X_k^j e^{-\imath 2\pi \frac{k-1}{T} D_{i,j}} + E_k^i. \tag{B.2}$$

We can collect the terms of $c_{i,j}$ into a matrix $\check{C} \in \mathbb{R}^{q \times p}$, where the elements of $\check{C}$ are the same as the nonzero elements of the sparse matrix $C \in \mathbb{R}^{q \times pq}$ defined in equation 3.5. If $\check{C}^{(k)} = \check{C} \circ e^{-\imath 2\pi \frac{k-1}{T} D}$, where $\circ$ denotes Hadamard's (element-wise) product and the exponential is computed element-wise for the matrix $D \in \mathbb{R}^{q \times p}$, we can write this for each time series indexed by n in matrix notation as

$$Y_k\{n\} = \check{C}^{(k)} X_k\{n\} + E_k\{n\}, \tag{B.3}$$

where $Y_k\{n\}, E_k\{n\} \in \mathbb{C}^{q \times 1}$ and $X_k\{n\} \in \mathbb{C}^{p \times 1}$. We can now alternatingly solve for $\check{C}$, X, and D to minimize the least-square error in both time and frequency domain (Morup et al., 2007). Briefly, at each iteration of the M-SISA algorithm, $\check{C}$ and X are computed using a pseudo-inverse, while D is computed using Newton-Raphson iterations. The detailed update equations are presented below.

***X* update:** For each time series indexed by n,

$$X_k\{n\} = \check{C}^{(k)^\dagger} Y_k\{n\}. \tag{B.4}$$

For X to be real valued, we update only the first $\lfloor T/2 \rfloor + 1$ elements according to equation B.4 and set the remaining elements such that $X_{T-k+1} = X_k^*$ where * denotes the complex conjugate.

$\check{C}$ **update:** Let $X_k^{j(i)} = X_k^j e^{-\imath 2\pi \frac{k-1}{T} D_{i,j}}$ be the delayed version of the latent $X_k^{(j)}$ to the ith observed variable. Then for each time series indexed by n,

$$y_t^i\{n\} = \sum_{j=1}^{p} \check{C}_{i,j} x_t^{j(i)}\{n\} + \varepsilon_t^i\{n\} \tag{B.5}$$

$$= \check{C}_{i,:} \mathbf{x}_t^{(i)}\{n\} + \varepsilon_t^i\{n\}. \tag{B.6}$$

We form $\tilde{\mathbf{y}}^i \in \mathbb{R}^{1 \times NT}$, $\tilde{X}^i \in \mathbb{R}^{p \times NT}$, and $\tilde{\boldsymbol{\varepsilon}}^i \in \mathbb{R}^{1 \times NT}$ by horizontally stacking $y_{1:T}^i\{n\}$, $x_{1:T}^{(i)}\{n\}$, and $\varepsilon_{1:T}^i\{n\}$ as follows:

$$\tilde{y}^i = [\, y_{1:T}^i\{1\} \quad y_{1:T}^i\{2\} \quad \dots \quad y_{1:T}^i\{N\} \,], \tag{B.7}$$

$$\tilde{X}^i = [\, \mathbf{x}_{1:T}^{(i)}\{1\} \quad \mathbf{x}_{1:T}^{(i)}\{2\} \quad \dots \quad \mathbf{x}_{1:T}^{(i)}\{N\} \,], \tag{B.8}$$

$$\tilde{\varepsilon}^i = [\, \varepsilon_{1:T}^i\{1\} \quad \varepsilon_{1:T}^i\{2\} \quad \dots \quad \varepsilon_{1:T}^i\{N\} \,]. \tag{B.9}$$

Then,

$$\tilde{y}^i = \check{C}_{i,:} \tilde{X}^{(i)} + \tilde{\varepsilon}^i. \tag{B.10}$$

This allows us to update each row of $\check{C}$ as

$$\check{C}_{i,:} = \tilde{y}^i \tilde{X}^{(i)^\dagger}. \tag{B.11}$$

***D* update:** Following Morup et al. (2007), the least square error for the single time series, equation B.3, is given by

$$ls\{n\} = \frac{1}{T} \sum_k (Y_k\{n\} - \check{C}^{(k)} X_k\{n\})^H (Y_k\{n\} - \check{C}^{(k)} X_k\{n\}). \tag{B.12}$$

Analytical forms for the gradient $g\{n\}$ and the Hessian $H\{n\}$ of $ls\{n\}$ with respect to the delays D can be computed (Morup et al., 2007). Therefore, for multiple time series indexed by n,

$$ls = \sum_{n=1}^{N} ls\{n\}, \tag{B.13}$$

$$g = \sum_{n=1}^{N} g\{n\}, \tag{B.14}$$

$$H = \sum_{n=1}^{N} H\{n\}. \tag{B.15}$$

We performed Newton-Raphson using the gradient g and the Hessian H to update the delays D.

M-SISA itself is prone to local optima and is initialized using a multistart procedure. Briefly, we ran 15 iterations of M-SISA initialized at 30 randomly drawn settings of the model parameters. Of these 30 draws, we selected the draw with the lowest least square error and ran 100 more iterations.

Once $\check{C}$ and D are computed, we can compute the residuals ε such that

$$\varepsilon_t^i\{n\} = \mathbf{y}_t^i\{n\} - \sum_{j=1}^{p} \check{C}_{i,j} \mathbf{x}_{t-D_{i,j}}^j\{n\}. \tag{B.16}$$

We then form R such that the diagonal elements of R are equal to the diagonal terms of the covariance matrix of ε across all time points and time series. C is formed by rearranging the elements of $\check{C}$ as required in equation 3.5. The initial value of the offset $\mathbf{d}$ is the mean of all observations, computed across all time steps in all the observed time series. We set the initial values of all the GP timescales τ_j, $j = 1, \ldots, p$ to 100 ms. We fix the GP noise variance $\sigma_{n,j}^2$ to a small positive value ($= 10^{-6}$) as in Yu et al. (2009).

Appendix C: Computational Requirements

This section summarizes the computational time required to fit TD-GPFA model parameters and extract the latent variables. The computation time depends on the latent dimensionality p, number of observed variables q, number of trials, length of each trial T, and number of ECME iterations.

In the ECME algorithm, the most expensive equation to compute is the E-step, equation A.5, which involves the inversion of a $qT \times qT$ matrix and multiplication with matrices of size $pqT \times pqT$. Large parts of this computation are constant for a fixed p, q, and T, and therefore they can be performed once for all trials of the same length T as they do not depend on the values of the observed variables in each trial. Each gradient computation in CM-step 2, equation A.15, is fast, but because this computation is performed thousands of times, it accounts for a large fraction of the total running time.

Our results were obtained on a Linux 64-bit workstation with a 4 core Intel Xeon CPU (X5560), running at 2.80 GHz and equipped with 50 GB RAM. To fit the parameters for 45 observed variables and 140 trials of length 520 ms ($T = 26$ with 20 ms time bins), our ECME algorithm (for 200 iterations) took 3 hours 14 minutes for two latent dimensions, and 9 hours 54 min for six latent dimensions. These results were obtained using an exact implementation without harnessing any possible advantages of using approximate techniques. We can also take advantage of parallelization to reduce run time, especially since during each ECME iteration, the delays and timescales can be learned independently for each latent variable. Lower run times may also be obtained by running fewer overall ECME iterations (in practice, the parameters often converge within the first 100 ECME iterations), as well as relaxing convergence criteria for gradient optimization of the delays (see equation A.15).

Acknowledgments

This work was funded by NIH NICHD CRCNS R01-HD071686 (B.M.Y. and A.P.B.), the Craig H. Neilsen Foundation (B.M.Y. and A.P.B.), NIH NINDS R01-NS065065 (A.P.B.), Burroughs Wellcome Fund (A.P.B.), NSF DGE-0549352 (P.T.S.) and NIH P30-NS076405 (Systems Neuroscience Institute).

References

Ahrens, M. B., Orger, M. B., Robson, D. N., Li, J. M., & Keller, P. J. (2013). Whole-brain functional imaging at cellular resolution using light-sheet microscopy. *Nature Methods*, *10*, 413–420.

Aoki, M., & Havenner, A. (1997). *Applications of computer aided time series modeling*. New York: Springer.

Barliya, A., Omlor, L., Giese, M. A., & Flash, T. (2009). An analytical formulation of the law of intersegmental coordination during human locomotion. *Experimental Brain Research*, *193*(3), 371–385.

Be'ery, E., & Yeredor, A. (2008). Blind separation of superimposed shifted images using parameterized joint diagonalization. *IEEE Transactions on Image Processing*, *17*, 340–353.

Briggman, K. L., Abarbanel, H. D. I., & Kristan Jr., W. B. (2005). Optical imaging of neuronal populations during decision-making. *Science*, *307*, 896–901.

Broome, B. M., Jayaraman, V., & Laurent, G. (2006). Encoding and decoding of overlapping odor sequences. *Neuron*, *51*, 467–482.

Churchland, M. M., Cunningham, J. P., Kaufman, M. T., Foster, J. D., Nuyujukian, P., Ryu, S. I., & Shenoy, K. V. (2012). Neural population dynamics during reaching. *Nature*, *487*, 51–56.

Cunningham, J. P., & Yu, B. M. (2014). Dimensionality reduction for large-scale neural recordings. *Nature Neuroscience*, *17*, 1500–1509.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, *39*(1), 1–38.

Everitt, B. S. (1984). *An introduction to latent variable models*. Amsterdam: Springer Netherlands.

Harshman, R., Hong, S., & Lundy, M. (2003a). Shifted factor analysis—Part I: Models and properties. *Journal of Chemometrics*, *17*, 363–378.

Harshman, R., Hong, S., & Lundy, M. (2003b). Shifted factor analysis—Part II: Algorithms. *Journal of Chemometrics*, *17*, 379–388.

Harvey, C. D., Coen, P., & Tank, D. W. (2012). Choice-specific sequences in parietal cortex during a virtual-navigation decision task. *Nature*, *484*, 62–68.

Kulkarni, J., & Paninski, L. (2007). Common-input models for multiple neural spike-train data. *Network: Computation in Neural Systems*, *18*, 375–407.

Liu, C., & Rubin, D. (1994). The ECME algorithm: A simple extension of EM and ECM with faster monotone convergence. *Biometrika*, *81*(4), 633–648.

Machens, C. K., Romo, R., & Brody, C. D. (2010). Functional, but not anatomical, separation of what and when in prefrontal cortex. *J. Neurosci.*, *30*, 350–360.

Macke, J. H., Buesing, L., Cunningham, J. P., Yu, B. M., Shenoy, K. V., & Sahani, M. (2011). Empirical models of spiking in neural populations. In J. Shawe-Tayor, R. S. Zemel, P. L. Bartlett, F. Pereira, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems, 24*, (pp. 1350–1358). Red Hook, NY: Curran Associates.

Mante, V., Sussillo, D., Shenoy, K. V., & Newsome, W. T. (2013). Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, *503*, 78–84.

Mazor, O., & Laurent, G. (2005). Transient dynamics versus fixed points in odor representations by locust antennal lobe projection neurons. *Neuron*, *48*, 661–673.

Meng, X.-L., & Rubin, D. B. (1993). Maximum likelihood estimation via the ECM algorithm: A general framework. *Biometrika*, *80*(2), 267–278.

Morup, M., Hansen, L. K., Arnfred, S. M., Lim, H. M., & Madsen, K. H. (2008). Shift-invariant multilinear decomposition of neuroimaging data. *NeuroImage*, *42*, 1439–1450.

Morup, M., Madsen, K. H., & Hansen, L. K. (2007). Shifted independent component analysis. In *Proceedings of Independent Component Analysis and Signal Separation 2007* (vol. 42, pp. 89–96), New York: Springer.

Omlor, L., & Giese, M. (2011). Anechoic blind source separation using Wigner marginals. *Journal of Machine Learning Research*, *12*, 1111–1148.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, *2*(6), 559–572.

Petreska, B., Yu, B. M., Cunningham, J. P., Santhanam, G., Ryu, S. I., Shenoy, K. V., & Sahani, M. (2011). Dynamical segmentation of single trials from population neural data. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems, 24* (pp. 756–764). Red Hook, NY: Curran.

Pfau, D., Pnevmatikakis, E. A., & Paninski, L. (2013). Robust learning of low-dimensional dynamics from large neuronal ensembles. In C. J. C. Burges, L. Bottov, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*, 26. Cambridge, MA: MIT Press.

Posadas, A. M., Vidal, F., de Miguel, F., Alguacil, G., Peña, J., Ibañez, J. M., & Morales, J. (1993). Spatial-temporal analysis of a seismic series using the principal components method: The Antequera series, Spain, 1989. *Journal of Geophysical Research: Solid Earth*, *98*(B2), 1923–1932.

Rahimi, A., Recht, B., & Darrell, T. (2005). Learning appearance manifolds from video. In *Computer Vision and Pattern Recognition* (pp. 868–875). New York: Springer.

Rasmussen, C. E., & Williams, C. (2006). *Gaussian processes for machine learning*. Cambridge, MA: MIT Press.

Rigotti, M., Barak, O., Warden, M. R., Wang, X. J., Daw, N. D., Miller, E. K., & Fusi, S. (2013). The importance of mixed selectivity in complex cognitive tasks. *Nature*, *497*, 585–590.

Roweis, S., & Ghahramani, Z. (1999). A unifying review of linear gaussian models. *Neural Computation*, *11*(2), 305–345.

Sadtler, P. T., Quick, K. M., Golub, M. D., Chase, S. M., Ryu, S. I., Tyler-Kabara, E. C., . . . Batista, A. P. (2014). Neural constraints on learning. *Nature*, *512*, 423–426.

Saha, D., Leong, K., Li, C., Peterson, S., Siegel, G., & Raman, B. (2013). A spatiotemporal coding mechanism for background-invariant odor recognition. *J. Neurosci.*, *16*, 1830–1839.

Santhanam, G., Yu, B. M., Gilja, V., Ryu, S. I., Afshar, A., Sahani, M., & Shenoy, K. V. (2009). Factor-analysis methods for higher-performance neural prostheses. *Journal of Neurophysiology*, *102*(2), 1315–1330.

Semedo, J., Zandvakili, A., Kohn, A., Machens, C., & Yu, B. (2014). Extracting latent structure from multiple interacting neural populations. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*, *27*, 2942–2950.

Siddiqi, S., Boots, B., & Gordon, G. (2007). A constraint generation approach to learning stable linear dynamical systems. In J. C. Platt, D. Koller, Y. Singer, & S. T. Roweis (Eds.), *Advances in neural information processing systems*, 20. Cambridge, MA: MIT Press.

Smith, A., & Brown, E. (2003). Estimating a state-space model from point process observations. *Neural Computation*, *15*, 965–991.

Spearman, C. (1904). General intelligence objectively determined and measured. *American Journal of Psychology*, *15*, 201–293.

Tipping, M. E., & Bishop, C. M. (1999). Mixtures of probabilistic principal component analyzers. *Neural Comput.*, *11*, 443–482.

Torkkola, K. (1996). Time-delay estimation in mixtures. In *IEEE Workshop on Neural Networks for Signal Processing* (4:423–432). Piscataway, NJ: IEEE.

Turk, M., & Pentland, A. (1991). Eigenfaces for recognition. *J. Cognitive Neuroscience*, *3*, 71–86.

Vidne, M., Ahmadian, Y., Shlens, J., Pillow, J. W., Kulkarni, J., Litke, A. M., . . . & Paninski, L. (2012). Modeling the impact of common noise inputs on the network activity of retinal ganglion cells in the primate retina. *Journal of Computational and Neuroscience*, *33*, 97–121.

Yeredor, A. (2001). Blind source separation with pure delay mixtures. In *Proceedings of the 3rd International Workshop on Independent Component Analysis and Blind Source Separation (ICA)*. San Diego: Institute for Neural Computation, University of California.

Yeredor, A. (2003). Time-delay estimation in mixtures. *Acoustics, Speech, and Signal Processing*, *5*, 237–240.

Yu, B. M., Afshar, A., Santhanam, G., Ryu, S. I., Shenoy, K. V., & Sahani, M. (2006). Extracting dynamical structure embedded in neural activity. In Y. Weiss, B. Schölkopf, & L. Bottou (Eds.), *Advances in neural information processing systems*, 18. Cambridge, MA: MIT Press.

Yu, B. M., Cunningham, J. P., Santhanam, G., Ryu, S. I., Shenoy, K. V., & Sahani, M. (2009). Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. In Y. Benglo, D. Schuurmans, J. Lafferty, C.K.I. Williams, & A. Culotta (Eds.), *Advances in neural information processing systems*, 21 (pp. 1881–1888).

Received January 22, 2015; accepted April 17, 2015.