Sparse Functional Dynamical Models—A Big Data Approach

Ela Sienkiewicz, Dong Song, F. Jay Breidt, and Haonan Wang

ABSTRACT
Nonlinear dynamical systems are encountered in many areas of social science, natural science, and engineering, and are of particular interest for complex biological processes like the spiking activity of neural ensembles in the brain. To describe such spiking activity, we adapt the Volterra series expansion of an analytic function to account for the point-process nature of multiple inputs and a single output (MISO) in a neural ensemble. Our model describes the transformed spiking probability for the output as the sum of kernel-weighted integrals of the inputs. The kernel functions need to be identified and estimated, and both local sparsity (kernel functions may be zero on part of their support) and global sparsity (some kernel functions may be identically zero) are of interest. The kernel functions are approximated by B-splines and a penalized likelihood-based approach is proposed for estimation. Even for moderately complex brain functionality, the identification and estimation of this sparse functional dynamical model poses major computational challenges, which we address with big data techniques that can be implemented on a single, multi-core server. The performance of the proposed method is demonstrated using neural recordings from the hippocampus of a rat during open field tasks. Supplementary materials for this article are available online.

1. Introduction
A dynamical system is a process evolving in time, with the current output of the system functionally dependent on current and past inputs. Nonlinear structure in the functional dependence can result in tremendous complexity in the system. The first documented study of a nonlinear dynamical system involved two interacting species, predators and prey, and was conducted independently by Volterra and Lotka in the 1910s and 1920s. The interaction was modeled through nonlinear differential equations (Volterra 1930; Flake 1963; Karmakar 1979), which represents the output of the system as a series of convolutions of its inputs. The Volterra series is often referred to as a Taylor series with memory.

For a system with a single input \( x(t) \) and a single output \( z(t) \), the Volterra series is

\[
z(t) = k_0 + \sum_{p=1}^{\infty} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} k_p(t_1, \ldots, t_p) \times x(t - t_1) \cdots x(t - t_p) \, dt_1 \cdots dt_p, \tag{1}
\]

where \( k_p \) is known as the \( p \)-th order Volterra kernel. In practice, the Volterra kernels are unknown and have to be estimated in the process of system identification.

In addition to early applications in the physical sciences (voltage, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981), nonlinear dynamical systems have been identified in areas as diverse as economics and psychology (human behavior, current, velocity, stress, temperature; Wiener 1958; Schetzen 1981)
In this article, we model the nonlinear dynamics of neural ensembles using spiking activities. Spikes, also termed action potentials, are brief electrical pulses generated in the soma and propagating along axons. Neurons receive spike inputs from other neurons through synapses and then transform these inputs into their spike outputs. Since spikes have stereotypical shapes, they can be simplified as point-process signals carrying information with the timings of the pulses. It is widely accepted that information in the brain is encoded in the spatio-temporal patterns of spikes. Thus, characterization of the point-process input-output properties of neurons using spiking activities is a crucial step for understanding how the neurons, as well as the neuronal networks transmit and process information; see Song et al. (2007, 2013). Within a neural network, this procedure is essentially the identification of the functional connectivity between neurons.

If the neural spikes are represented as binary signals (e.g., Song et al. 2007, 2009), then Equation (1) is not sufficient to describe the output of the system in terms of its inputs, and a generalized linear model (GLM) is often introduced. In Song et al. (2007, 2009), Volterra kernel models were combined with generalized linear models to characterize the input-output properties of neurons. To facilitate model estimation, Volterra kernels were expanded with basis functions, for example, Laguerre basis or B-spline basis, and model coefficients were estimated with a penalized likelihood method (Song et al. 2013).

However, this estimation methodology imposes major computational challenges, chiefly because the first-order Volterra series has been shown to be inadequate to capture the system dynamics (Song et al. 2007). The second-order series, with even a moderate number of inputs and basis functions, requires estimating thousands of parameters. A large number of observations is critical to account for sparsity in the inputs and to avoid over-smoothing, see Marmarelis (1993) and Korenberg and Hunter (1996).

With a large sample size and large number of parameters, system identification faces a big data challenge, which is characterized by both hardware and software issues; see Fan, Han, and Liu (2014) for a recent review. The data may not fit in the memory of a single computer, and some statistical methods, if they can be adjusted to accommodate distributed data and computing, just take too long to be computationally feasible. As a result, none of the cited articles was successful in providing a viable and reproducible method of regularized estimation of Volterra kernels of higher orders. Regularization is key to achieving functional sparsity at both local and global levels, necessitated by the sparsity in neural connectivity (Tu et al. 2012). Song et al. (2013) provided a comprehensive study of sparsity by comparing the effects of Laguerre basis and B-spline basis. Both types are adequate in representing the global sparsity, but only the B-spline basis successfully modeled the local sparsity. To achieve sparsity, Song et al. (2009) used the forward step-wise model selection and Song et al. (2013) implemented a regularization-based method, that is, LASSO (Tibshirani 1996).

Aside from nonparametric modeling using Volterra kernels, other approaches include cascade or block structure modeling (Korenberg and Hunter 1986), basis representation of orthogonalized Wiener kernels (Marmarelis 1993), and artificial neural networks, which have a correspondence with Volterra kernel models (Wray and Green 1994). These alternative methods are not considered further in this article, though some of the computational techniques we describe can be applied in these other settings.

The contributions of this article are threefold. First, we tackle the problem of function estimation using a classical likelihood method in the analysis of neural spikes. Second, for better system identification and estimation, we present a novel regularization approach for function estimation. In particular, we use a sparse basis expansion of second-order or higher-order Volterra kernels and regularize to preserve both local and global sparsity. Third, we combine various existing statistical and optimization techniques in a novel way and apply big-data techniques for fast and reliable computations. Specifically, we show that the standard big-data approach of distributing data does not lead to a feasible computational strategy in our problem, so we show instead how to perform all the necessary computations on a single multi-core server.

The rest of this article is organized as follows. Section 2 describes the data and the research goals. Section 3.1 presents our proposed dynamical multiple-input single-output (MISO) model based on the generalized second-order Volterra series. In the remaining subsections of Section 3, we discuss the likelihood-based and penalized estimation of the parameters of the model. In Section 4, we present computational details in the context of big data. We describe the challenges encountered in fitting the proposed model when both numbers of parameters and observations are large, and we suggest methods to overcome those challenges. In Section 4.2, we describe the design matrix and show how to take advantage of redundancy in its structure. In Sections 4.3 and 4.4, we detail the methods of likelihood-based and penalized estimation, using only matrix-vector operations. We show in Section 4.7 how matrix-vector operations can be executed in the map-reduce environment. In Section 5, we conclude with model development and validation for neural recordings from the hippocampus of a rat during open field tasks.

2. Data Description and Problem Formulation

The neural spike data that we consider are available from the public repository crns.org; for details see Mizuseki et al. (2009a,b). The neural recordings come from layer CA1 of the dorsal hippocampus, an area of the brain associated with learning and forming memories, in particular spatial memories. The recordings are performed with eight electrodes on a Long-Evans rat during open field tasks, within a 90-min time frame. Each electrode registers neural spikes from a cluster of neurons. Though we cannot reliably differentiate cells within a cluster, we will for simplicity refer to the cluster as a neuron. A portion of the dataset, recorded over 3 sec, is shown in Figure 1.

In Figure 1, each horizontal line contains a sequence of spikes from one neuron. Here, each short vertical line indicates the presence of a spike. The data are recorded at a sampling rate of 20 MHz. Although timings of neural spikes are exact, it is practical to encode each spike train as a binary process, a 1 is recorded when the spike is observed at some predefined time interval, and 0, if it is not. For an interval of 1 msec, this transformation leads to 5.4 million observations for each neuron. More details are provided in Section 3.1.
Our primary goal is to model the functional relationship between the output neuron and input neurons. In particular, our interest centers on the probability of the output spike at a specific time given the history of input and output neuron firings. This relationship is known to be nonlinear and time-dependent (Song et al. 2009; Fuchs 2013), so we adapt the Volterra series to the generalized linear model setting, with discrete inputs. Our second goal is the identification of functional connectivity between neurons. The connectivity is known to be sparse (He 2005; Song et al. 2007). Here, we formulate this problem as a model selection problem. In particular, we take a regularization approach to establish a set of relevant input neurons as well as intervals of relevance.

3. Multiple-Input and Single-Output Models

3.1. Generalized Second-Order Volterra Model

The output of a neuron or neural ensemble consists of a series of spikes, and it is convenient to model it in terms of a probability or a rate of action potentials. Let \( y(t) \) be a binary output signal, and let \( x_1(t), \ldots, x_d(t) \) \((0 \leq t \leq T)\) be \( d \) binary input signals indicating the times at which the spikes occurred in a \( T \) seconds interval. Both input and output signals are recorded at a fixed sampling frequency, that is, for each signal, observations are obtained at \( t = \delta, 2\delta, \ldots, m\delta \) and \( T = m\delta \), where \( \delta \) is the length of a sampling time interval. \( A \) is recorded if a spike occurs in time interval \((t - 1)\delta, t\delta]\), and 0 otherwise.

Let \( x(t) = (x_1(t), \ldots, x_d(t))^T \) be a vector of input signals. The conditional probability of the action potential of the output signal, given the history of the input and output signals, can be written as \( \theta(t) = P(y(t) = 1 \mid \mathcal{H}(t)) \), where \( \mathcal{H}(t) = (x_1(t), x_2(t - \delta), \ldots, x_d(t - m\delta), y(t - \delta), \ldots, y(t - m\delta))^T \). Here \( M = m\delta \) is the length of the system memory.

The generalized functional additive model with an appropriate link function is a natural starting point for estimation of \( \theta(t) \). To capture system dynamics in such a model, we adopt (1) to our context and propose a generalized second-order Volterra model

\[
\begin{align*}
\Phi^{-1}(\theta(t)) &= \kappa_0 + \sum_{i=1}^{d} \int_{0}^{M} \varphi_i^{(1)}(\tau) dN_{x_i}(t - \tau) \\
&\quad + \sum_{i_1=1}^{d} \sum_{i_2=1}^{d} \int_{0}^{M} \int_{0}^{M} \varphi_{i_1,i_2}^{(2)}(\tau_1, \tau_2) dN_{x_{i_1}}(t - \tau_1) dN_{x_{i_2}}(t - \tau_2)
\end{align*}
\]

where \( \Phi^{-1} \) is the probit link function, an inverse of a standard normal distribution function, and \( \kappa_0, \varphi_i^{(1)}(\tau), \varphi_{i_1,i_2}^{(2)}(\tau_1, \tau_2) \) are Volterra kernels of zero-, first-, and second-order, respectively. In addition, \( N_{x_i}, N_y \) are counting measures with respect to input and output signals. Our choice of the probit link function, over other options including the commonly used logit, is due to the broader use of the probit in neural modeling (Song et al. 2013).

Like the system output \( z(t) \) in the Volterra series (1), \( \Phi^{-1}(\theta(t)) \) in (2) is an analytic function of \( t \). Our proposed model is a causal model with a finite memory, so that \( \int_{-\infty}^{t} \) in (1) is replaced by \( \int_{0}^{M} \) in (2). This choice is determined by the nature of the brain dynamics: a neural firing is influenced by only the recent history of input firings. Our model (2) also contains an autoregressive component as its final term, which is essential to model the refractory period between neural firings. Finally, the integrals in (2) are with respect to counting measures, to account for the binary point-process inputs.

Model (2) is of second order. It can be further extended by including higher-order terms. The zeroth-order \( \kappa_0 \) captures the baseline probability of an output spike, when no inputs are present. Each first-order kernel \( \varphi_i^{(1)}(t) \) measures the contribution of a corresponding input signal. The second-order kernels \( \varphi_i^{(2)}(t_1, t_2) \) measure the interaction effects between the same input at a different time lag, and between different inputs. The kernels are unknown and need to be estimated in the process of system identification.

A discretized version of this model has been proposed in Song et al. (2007), where each kernel function is discretized and represented as a vector of coefficients. The estimation procedure for these coefficients is likelihood-based. The drawback is that the numbers of parameters and observations are very large, which makes computations intractable and unstable. For instance, in our motivating example, the total number of parameters in the discretized model is on the order of \( 10^6 \). Our proposed model, based on continuous Volterra kernels, provides a more feasible computational strategy; see Section 4 for details.

3.2. Likelihood and Parameter Estimation

In this section, we consider a likelihood-based approach for estimation of functionals \( \varphi \) in (2). For the binary output signal
Finally, we derive the approximate log-likelihood \( l_\omega \) as follows:

\[
l_\omega (\alpha_0, \alpha) = \sum_{s=m+1}^{n} y(s\delta) \log \{ \Phi (\alpha_0 + X(s\delta)\alpha) \} \\
+ (1 - y(s\delta)) \log \{ 1 - \Phi (\alpha_0 + X(s\delta)\alpha) \}. \tag{8}
\]

The maximizers of (8) are denoted \( (\hat{\alpha}_0^{\text{MLE}}, \hat{\alpha}^{\text{MLE}}) \), where

\[
\alpha = \left( \alpha_1^{(1)}, \alpha_2^{(1)}, \ldots, \alpha_j^{(d+1)}, \alpha_{j+1}^{(1)}, \alpha_{j+2}^{(1)}, \ldots, \alpha_{J}^{(d,d)} \right)^T.
\]

Maximizing (8) with respect to the unknown parameters is a well-studied convex optimization problem with an iterative solution based on the Taylor series approximation of (8) or equivalently, the Newton–Raphson method or Fisher scoring (McCullagh and Nelder 1983; Green 1984). These two methods are not equivalent for the probit link, unlike the case for logistic regression. Newton–Raphson provides faster convergence but requires the calculation of the Hessian matrix at each step (compared to computationally simpler alternative of Fisher information). More details are given in Section 4.3.

### 3.3. Functional Sparsity and Neural Connectivity

In a neural network, neural connectivity is believed to be sparse in nature (He 2005; Song et al. 2007). Two different types of sparsity were defined by Tu et al. (2012). In particular, given multiple input neurons, only a small subset of these may have any relevance to the signal recorded from the output neuron. This property is referred to as global sparsity, and it should lead to the estimation of functional \( \psi_i \) corresponding to irrelevant inputs as zero. Among the inputs that show relevance to the output signal, the impact will be limited in time, and the duration of the effect is often of interest. Outside of the interval in question, the functional will be zero, which is referred to as local sparsity.

Maximizing (8) does not guarantee either sparsity, so a penalized approach enforcing sparsity is preferable. For parametric regression models, penalized approaches have been widely studied and have gained in popularity due to simultaneous parameter estimation and variable selection. Commonly used penalization methods include LASSO (Tibshirani 1996), SCAD (Fan and Li 2001), group LASSO (Yuan and Lin 2006), and group bridge (Huang et al. 2009). In Wang and Kai (2015), these penalization methods were evaluated for the problem of detecting functional sparsity for univariate, nonparametric regression models.

In addition, these authors incorporated the group bridge penalty with an intuitive grouping method for estimating functions with global or local sparsities.

For demonstration purposes, consider the function shown as a thick, solid line in the bottom panel of Figure 2. Note that this function is zero on \([0, 0.2]\). In the top panel, 13 cubic B-spline basis functions with nine equally spaced interior knots are depicted. To approximate the bottom-panel function with these B-splines and achieve sparsity on each subinterval of \([0, 0.2]\), zero coefficients for the first four B-splines (shown with dashed lines in the top panel) are required. Hence, it is natural to treat these four coefficients as a group. This grouping generalizes beyond cubic: for B-spline basis functions with degree \( b \), each group consists of \( b + 1 \) coefficients.

**Figure 2.** Top: A set of 13 cubic B-spline basis functions with nine equally spaced interior knots on the interval \([0, 1]\). Bottom: An illustration of a function with local sparsity on the interval \([0, 0.2]\). There are 10 subintervals corresponding to 10 groups. On each subinterval, there are four basis functions coefficients taking positive values. For instance, to identify the sparsity of the function on \([0, 0.1]\), all four basis functions (dashed lines) need to have zero coefficients.
In this article, we adopt the grouping idea suggested by Wang and Kai (2015). For the ith first-order kernel $\psi^{(i)}_1(\tau)$ in (5), its coefficients for the B-spline approximation in (5) can be assigned to $f - b$ overlapping groups, $\{a_1^{(i)}, a_2^{(i)}, \ldots, a_{b+1}^{(i)}\}, \{a_2^{(i)}, a_3^{(i)}, \ldots, a_{b+2}^{(i)}\}, \ldots, \{a_{f-b}^{(i)}, a_{f-b+1}^{(i)}, \ldots, a_f^{(i)}\}$. For the second-order kernel $\psi^{(i)}_2(\tau_1, \tau_2)$, the grouping becomes rather complicated. Over each small rectangle spanned by two adjacent knots on $\tau_1$ and $\tau_2$, the sparsity can be determined by $(b + 1)^2$ coefficients. To be more specific, the coefficients of the B-spline approximation in (6) can be assigned to $(f - b)^2$ overlapping groups,

$$
\begin{bmatrix}
a_{1,1}^{(i,iz)} & \cdots & a_{1,b+1}^{(i,iz)} \\
\vdots & \ddots & \vdots \\
a_{b+1,1}^{(i,iz)} & \cdots & a_{b+1,b+1}^{(i,iz)}
\end{bmatrix},
\begin{bmatrix}
a_{1,2}^{(i,iz)} & \cdots & a_{1,b+2}^{(i,iz)} \\
\vdots & \ddots & \vdots \\
a_{b+1,2}^{(i,iz)} & \cdots & a_{b+1,b+2}^{(i,iz)}
\end{bmatrix},
\ldots
\begin{bmatrix}
a_{f-k,b}^{(i,iz)} & \cdots & a_{f-k,1}^{(i,iz)} \\
\vdots & \ddots & \vdots \\
a_{f,1}^{(i,iz)} & \cdots & a_{f,b+1}^{(i,iz)}
\end{bmatrix}
$$

For our convenience, we write the group penalty as $P_k^{(y)}(\alpha) = \lambda \sum_{g=1}^{G} |\alpha_{Ag}|^y$, where $G$ is the total number of groups and $y \in (0, 1)$ is a constant. Here $\alpha_{Ag_i}$ represent the groups of coefficients as defined above.

Finally, the penalized log-likelihood criterion can be expressed as

$$l_\alpha(\alpha_0, \alpha) - P_k^{(y)}(\alpha).$$

The penalized maximum likelihood estimation (MLE) is defined as the maximizer of this criterion. The computational aspects are covered in Section 4.5.

4. Computational Aspects With Big Data

4.1. Overview of the Algorithm

In this section, we focus on computational aspects of fitting model (2), beginning with maximization of the approximate log-likelihood (8). We proceed by selecting statistical and optimization methods that can be adapted to big data computations because they operate on input data only through matrix-vector operations that can be tailored to a parallel environment; see Section 4.2 for initial results. The methods include iteratively reweighted least squares, which maximizes a quadratic approximation to the log-likelihood function. Each iteration of IRLS relies on an $L_2$-minimization step, which can be executed using the conjugate gradient method. The computational details are available in Section 4.3.

Next, we consider maximization of (9), a penalized version of the approximate log-likelihood. Some penalties lead to non-convex optimization problems. Yet, for penalties like SCAD or Group Bridge, a solution can be found iteratively by solving a sequence of $L_1$ problems. Maximization of the log-likelihood function with $L_1$ (or LASSO) penalty can be solved using the Coordinate Descent algorithm, which requires the maximum likelihood estimate as an input. Details of this method, and the derivation of correspondence between Group Bridge and a sequence of LASSO steps, are described in Sections 4.4 and 4.5.

Pseudo-code and the code location are available in the supplementary materials.

4.2. Design Matrix

Under the B-spline approximations (5) and (6), the conditional mean of $y$ is

$$E\left[y \mid y(\delta), \ldots, y(m\delta)\right] = \Phi(\alpha_0^0 + X_0\alpha),$$

where the standard normal cdf $\Phi(\cdot)$ is applied element-wise. Write $X = [X_1, X_2]$, where $X_1$ and $X_2$ correspond, respectively, to the first-order kernels in $X_1(t)$ and the second-order kernels in $X_2(t)$, as defined in (7).

The elements of the matrix $X_1$, $(\xi(t), \ldots, \xi(t), \ldots, \xi^{(d+1)}(t))$, can be calculated as

$$\xi^{(i)}(t) = \int_0^X B_j(\tau)d\nu_{\alpha_0}(t - \tau) = \sum_k B_j(t - t_k),$$

where $\{t_k\}$ denote the set of time stamps in $(t - M, t]$ such that $X(\tau_k) = 1$, for $t = (m + 1)\delta, \ldots, n\delta, j = 1, \ldots, f$ and $i = 1, \ldots, d + 1$. The element $\xi^{(i)}(t)$ is in row $r$, such that $t = (m + r)\delta$, and column $(i - 1)f + j$. The elements of the matrix $X_2$, each row $r$ containing $(\xi^{(1)}(t), \xi^{(2)}(t), \ldots, \xi^{(d+1)}(t))$, are derived using the formula

$$\xi^{(i)}(t) = \int_0^X B_j(t_1, \tau_2)d\nu_{\alpha_0}(t - t_1)d\nu_{\alpha_0}(t - t_2),$$

The two-dimensional B-spline basis functions are tensor products of the one-dimensional B-spline basis functions. Consequently, the rows of $X_2$ can be efficiently calculated as the Kronecker products of the rows of $X_1$. This observation is rather important for big data computation; in fact, the matrix $X_2$ does not have to be generated in its entirety: its rows can be dynamically recreated during matrix-vector operations. This largely alleviates storage and computational challenges in further calculations. More details are provided in Section 4.7.

4.3. Maximum Likelihood Estimation With Conjugate Gradient

Iteratively reweighted least squares (IRLS) is a well-established algorithm for finding a maximum likelihood estimate of a generalized linear model (10); see McCullagh and Nelder (1983) for more details. This iterative approach is based on the Taylor expansion of the log-likelihood function using either observed or expected information. To simplify the implementation, we employ the expected information (i.e., Fisher scoring). At the $(k + 1)$st step, one finds the estimate of $\alpha$, say $\alpha^{(k+1)}$, based on the previously calculated $\alpha^{(k)}$ by minimizing $\|W_\alpha^{(k)}(\alpha^{(k)}) - \alpha_0 - X_0\alpha\|^2_2$. Here, $W_\alpha^{(k)}$ is a diagonal matrix with elements that depends on the previously calculated $\alpha^{(k)}$ via

$$w_j^{(k)} = \phi \left[\left|\theta_j^{(k)} - \theta_j^{(k)}\right|\right].$$
The vector $z^{(k)}$ has elements $z^{(k)}_i$ that depend on current responses via

$$z^{(k)}_i = a_0^{(k)} + x_i^T \alpha^{(k)} + \left( y_i - \hat{y}_i^{(k)} \right) / \phi(a_0^{(k)} + x_i^T \alpha^{(k)}),$$

where $x_i^T$ is the $j$th row of $X$, $\hat{y}_i^{(k)} = \Phi(a_0 + x_i^T \alpha^{(k)})$, and $\Phi$ is the density and distribution functions of standard normal, respectively. Moreover, the minimization problem above has a closed-form solution, (9), which is based on a linear approximation of the SCAD penalty function. Thus, the solution of the quadratic optimization problem can then be expressed as a linear combination of these vectors. The coefficients of the conjugate vectors can be obtained iteratively in $p$ steps. The conjugate gradient method has been successfully applied to classification in the big data context due to its scalable computational complexity; see Komarek (2004). In fact, the number of operations for a convex quadratic problem is estimated to be $O(p)$. The implementation requires approximately $p$ matrix-vector multiplications; see, for example, Antoniou and Lu (2007) for details. To carry out the IRLS algorithm, the minimization problem at the $(k+1)$st step can be expressed in terms of $H = 2Z^T W_k Z$ and $b = 2Z^T W_k z^{(k)}$, where $W_k$ and $z^{(k)}$ are defined as in (13) and (14). In big data computation, it may not be desirable to compute the matrix $H$ in every iteration of IRLS, since matrix-matrix operations are expensive. But $H$ is only used in matrix-vector multiplications, so one can compute $2Z^T W_k z^{(k)}$ instead, for some vector $x$ of size $p \times 1$. This operation is computable in a map-reduce framework; see Section 4.4.7. The vector $b$ is also of size $p \times 1$ and it can be computed with map-reduce and stored in memory.

### 4.4. Penalized Maximum Likelihood Estimation

The MLE obtained using IRLS in Section 4.3 does not provide any sparsity. One way to enforce sparsity of the identified model is to consider maximizing the penalized log-likelihood as expressed in (9). The optimization problem poses computational challenges since it is nonconvex. A popular approach is to approximate the log-likelihood function by a quadratic function. Let $l_o(\alpha) = l_o(\alpha^{(MLE)}, \alpha)$ be the log-likelihood function after plugging in $\alpha^{(MLE)}$. The approximation of $l_o(\alpha)$ can be obtained via a Taylor series expansion at $\alpha^{(MLE)}$, that is, $l_o(\alpha) \approx l_o(\alpha^{(MLE)}) + \frac{1}{2} (\alpha - \alpha^{(MLE)})^T \nabla^2 l_o(\alpha^{(MLE)}) (\alpha - \alpha^{(MLE)})$, where $\nabla^2 l_o(\alpha^{(MLE)})$ is the Hessian of the log-likelihood function evaluated at $\alpha^{(MLE)}$, $Z = [1, X]$ and $W$ is a diagonal weight matrix. This quadratic approximation can be expressed as $l_o(\alpha) \approx l_o(\alpha^{(MLE)}) - \frac{1}{2} \| W^T (z - X \alpha) \|^2_2$ where

$$z = X \alpha^{(MLE)}.$$  

The penalized criterion (9) can be expressed as

$$\min_{\alpha} \frac{1}{2} \| W^T (z - X \alpha) \|^2_2 + p(\gamma)(\alpha).$$

In the special case of $\gamma = 1$, the penalty is essentially the LASSO penalty (Tibshirani 1996). For $\gamma \in (0, 1)$, the optimization problem is nonconvex, and Huang et al. (2009) provided an iterative solution:

1. For a given $\lambda_m$, calculate $\tau_g = \lambda_m^{1/\gamma} \gamma^{1-\gamma} (1 - \gamma)$
2. For $s = 1, 2, \ldots$, until convergence, compute

$$\alpha^{(s)} = \min_{\alpha} \frac{1}{2} \| W^T (z - X \alpha) \|^2_2 + \sum_{g=1}^G (\tau_g)^{1-\gamma} \sum_{i \in A_g} | \alpha_i |,$$

where $\tau_g = (\gamma g^{1/\gamma} - 1)^{\gamma}$ for $g = 1, \ldots, G$. Each stage of the iterative procedure can be reduced to the minimization of a quadratic form with $L_1$ penalty applied to the coefficients $\alpha$. For a small dataset, it can be efficiently solved by the LARS algorithm (Efron et al. 2004). However, in the big data context, this algorithm may not be feasible. Solving this problem with a large dataset is the subject of the next section.

The group bridge penalty described above can encounter stability issues in calculations (Percival 2012). An alternative approach is to use the SCAD penalty (Fan and Li 2001) $p_\lambda(-)$, which satisfies $p_\lambda(0) = 0$ and

$$p_\lambda'(\theta) = \lambda \left\{ I(\theta \leq \lambda) + \frac{(\lambda - \theta)}{(\lambda - \theta)} I(\theta > \lambda) \right\}$$

for a constant $\lambda$ usually equal to $3.7$. An analog of (9), equipped with SCAD penalty, can be expressed as

$$\frac{1}{2} \| W^T (z - X \alpha) \|^2_2 + \sum_{g=1}^G \eta_g \| \alpha_{A_g} \|_1.$$

Zou and Li (2008) proposed a one-step estimation procedure, which is based on a linear approximation of the SCAD penalty function. Thus, (16) can be written as a standard LASSO problem, that is,

$$\frac{1}{2} \| W^T (z - X \alpha) \|^2_2 + \sum_{g=1}^G \eta_g \| \alpha_{A_g} \|_1,$$

where $\eta_g = p'(\| \alpha_{A_g}^{(MLE)} \|_1)$. As will be seen in Section 5, in our application, the estimates using the group bridge penalty and the SCAD penalty are comparable.

### 4.5. L₁ Regularization via Coordinate Descent Algorithm

As outlined in Section 4.4, parameter estimation with the group bridge penalty and SCAD penalty can be reduced to the LASSO problem. LASSO has an efficient implementation using a variation of the LARS algorithm (Efron et al. 2004), which is not well-suited for operating on a design matrix of high dimension. Another implementation was suggested by Friedman et al. (2007) and Friedman, Hastie, and Tibshirani (2010). The elastic-net penalty, of which LASSO is a special case, can be computed by a coordinate descent and soft thresholding. The algorithm is applicable to linear models, or generalized linear models, in which case it can be integrated with the IRLS steps.
The idea of soft thresholding was proposed by Donoho and Johnstone (1995), and it was designed to recover a signal in the noisy environment by providing an adaptable threshold. This approach is motivated by a simple optimization problem: \[ \min \{ \frac{1}{2} (t - t_0)^2 + \lambda |t| \} \]
where \( \lambda > 0 \) and \( t_0 \) is a fixed constant. The solution can be written as \( t_{\text{min}} = S(t_0, \lambda) \equiv \text{sign}(t_0)(|t_0| - \lambda)_+ \). This observation motivated the gradient descent algorithm for the LASSO solution; see Friedman, Hastie, and Tibshirani (2010) for more details.

In particular, starting from any given parameter values, one parameter is updated using the soft thresholding. The detailed algorithm is described in Friedman, Hastie, and Tibshirani (2010). When implementing this algorithm, only matrix-vector products of the form \( X^T y \) are necessary and can be carried out efficiently under the map-reduce framework.

### 4.6. Tuning Parameters

The performance of parameter estimation relies on the choice of \( \lambda \) and \( \gamma \). Here, the parameter \( \gamma \) is fixed at 0.5 following recommendations in Huang et al. (2009). The common procedure for selecting \( \lambda \) is to proceed with model identification for multiple values of the coefficient and choose one according to a selected criterion, for example, Akaike’s Information Criterion (AIC; Akaike 1973), Schwarz’s Bayesian criterion (BIC; Schwarz 1978), or, especially for larger models, extended BIC (EBIC; Chen and Chen 2008). Here, we use the smallest EBIC value as a selection criterion for \( \lambda \). EBIC adds an extra component to BIC, namely, the logarithm of the number of models of the given size. Such criterion-based methods are suitable for the selection in the first-order model, but more problematic for the second-order model, or for any large dataset in general. The selection of \( \lambda \) for the second-order model requires multiple identification efforts, but time cost (a major concern for big data analytics) does not allow an exhaustive search. For the second-order model, we suggest to search for the tuning parameter among the predetermined sequence of candidates starting from the value selected for the first-order model.

### 4.7. Big Matrix Processing

Computations discussed in Sections 4.3 and 4.4 rely on just three types of matrix-vector operations, specifically \( X^T W y \), \( X^T W \beta \), and \( X^T W X \beta \). Here, \( X \) is the \( N \times p \) design matrix, \( W \) is the diagonal \( N \times N \) matrix of weights, \( y \) is the \( N \times 1 \) vector, and \( \beta \) is the \( p \times 1 \) vector. In our settings, \( N = n - m \), and both \( N \) and \( p \) are large.

We perform all matrix-vector operations in the map-reduce framework; see Dean and Ghemawat (2008) for details. Map-reduce is the algorithmic concept based on the division of tasks (between mappers) and assembly of results (by reducers). Any matrix-vector operation can be divided into \( K \) smaller tasks executed by mappers, with the results of these tasks collected by reducers. For instance, denote by \( X_{[k]} \) the \( k \)th part of matrix \( X \), consisting of \( n_k \) consecutive rows. Also, denote by \( W_{[k]} \) the diagonal matrix of size \( n_k \times n_k \), with weights corresponding to rows of \( X_{[k]} \), and by \( y_{[k]} \) a vector of size \( n_k \times 1 \) with elements of \( y \) corresponding to rows of \( X_{[k]} \). One can easily verify the following:

\[
X^T W X \beta = \sum_{k=1}^{K} X_{[k]} W_{[k]} X_{[k]} \beta
\]

\[
X^T W y = \sum_{k=1}^{K} X_{[k]} W_{[k]} y_{[k]}
\]

\[
X W \beta = (X_{[1]} W_{[1]} \beta, \ldots, X_{[K]} W_{[K]} \beta)^T.
\] (17)

These tasks can be divided between \( C \) servers, or \( C \) cores, where \( C \ll N \). The distinct processes or threads will execute all \( K \) tasks, with at most \( C \) executed in parallel. The design matrix \( X \) described in Section 4.2 is too large to fit in the memory of a single computer. An established approach is to distribute the matrix row-wise between servers in the Hadoop cluster or a Spark cluster. Hadoop and Spark are Apache Software Foundation implementations of map-reduce in the distributed environment. Both impose a steep penalty in terms of computing time, primarily due to the distributed communications. Spark is a better choice for statistical computations, which often require hundreds of operations on the same matrix. The performance results, presented in Table 1, are not reassuring, especially if the cluster is small.

Here, we propose a much more efficient computational strategy than the established approach based on distributed computations. Our method supports map-reduce computations on a single, multicore server. As shown in Section 4.2, \( X \) consists of two column-wise parts, specifically \( X_1 \), which corresponds to first-order kernels, and \( X_2 \), which corresponds to second-order kernels. In addition, as demonstrated in (12), each row of \( X_2 \) is a modified Kronecker product of a part of a row of \( X_1 \) (excluding duplicate entries and columns corresponding to \( \phi_{d+1}^{(1)} \)). When a matrix–vector operation is executed on a multicore, we set \( K = N \), so that \( X_{[k]} = X_{1}^{(k)} \) and each simple task involves a single row of matrix \( X \). It is easy to notice, that, for instance,

\[
x_{k}^T w k \beta = (x_{k}^{(1)})^T w_{k} \beta^{(1)} + (x_{k}^{(1)} \otimes x_{k}^{(1)})^T w_{k} \beta^{(2)},
\] (18)

where \( x_{k}^{(1)} \) is a part of \( x_{k} \) corresponding to first-order components, \( \beta^{(1)} \) and \( \beta^{(2)} \) are parts of \( \beta \) corresponding to first- and second-order components, respectively. Also, \( \otimes \) is a modified Kronecker product, which contains only unique components of the product, and does not contain elements of \( x_{[k]} \) corresponding to \( \phi_{d+1}^{(1)} \). Each of the \( K \) tasks can be executed without having \( x_{k}^{(2)} \) available. The entire operation can be executed without \( X_2 \) ever present in memory. In our application, \( X_1 \), with over 5 million rows and 120 columns, requires around 9 gigabytes of storage. The matrix \( X_1 \) fits in the memory of a single computer, but some degree of parallelization of tasks is needed to offset the additional complexity of (18). If \( X_1 \) is too large for a single computer, it can be placed in a distributed filesystem. The single-machine strategy will not apply, but the methods described here will still be applicable.
5. Data Analysis

5.1. Analysis of Neural Spikes

In this section, we analyze the spike train data described in Section 2. When discussing the model in Section 3, we introduced several system parameters, for example, the system memory. Here, the system memory length is chosen to be $M = 400$ msec. Based on earlier studies (e.g., Song et al. 2013), the memory length is generally less than 1 sec. In our data analysis, results from $M = 400$ and $M = 1000$ are very similar. For illustration purposes, we present only the results with $M = 400$ msec. Another system parameter is $J$, the number of B-spline basis functions in the approximations (5) and (6). We choose cubic splines with 10 equally spaced knots on the interval $(0, M)$, with two additional knots near zero where, according to neuroscientists, the signal is strong. These knot choices result in $J = 15$. The number of equally spaced knots and the number of additional near-zero knots are selected from a small grid using EBIC for the first-order model. All observations are used to fit the first- and second-order models. The first-order model contains only the intercept and first-order Volterra kernels, no interaction components are included. The second-order model contains all components as listed in (2). Both models estimate first-order Volterra kernels, which can be compared in Figures 3–6. Only the second-order model estimates second-order Volterra kernels, which are presented in Figure A of the supplementary materials.

5.2. First-Order Model

In this section, we model the probability of neural spikes as a GLM defined in (10) with a matrix $X = X_1$ and its elements defined in (11). Even though the number of observations $n$ is over five million, the matrix $X_1$ can be computed and stored in memory for all calculations. Here we assume that $\varphi(m \delta) = 0$. We also consider a constraint $\varphi(0) = 0$ because the impact of the input spike is expected to have a positive delay. However, the delay might not be detectable in the data if the sampling rate is not high enough, in which case the value of $\varphi(0)$ may correspond to the maximum value of the functional. In fact, this turns out to be the case for all seven input neurons. Figures 3 and 4 show the results of the first-order model fit, with the unpenalized estimate depicted with a solid line, and three different penalized fits including group bridge (dashed), group SCAD (dotted), and LASSO (dash-dotted). The zoom-in view in each image shows the differences in sparsity estimation for all four models. The group bridge fit exhibits local sparsity in four out of seven neurons. The group SCAD and LASSO fits do not achieve the local sparsity in the examined range. The group bridge estimate was calculated with $\lambda_2 = 40$ selected using the EBIC criterion. The group bridge fit has a lower EBIC value than those of the group SCAD fit and LASSO fit, so we will employ it as a regularization method in the second-order model.

Figure 5 depicts the unpenalized estimation and a group bridge estimation of the kernel corresponding to the feedback kernel. The negative value close to 0 confirms the existence of a clustering effect in neural spikes, which can also be observed in Figure 1. Local sparsity is not detected by any estimator.

5.3. Second-Order Model

In this section, we model the probability of neural spikes as a GLM defined in (10) with a matrix $X = [X_1, X_2]$ and its elements defined in (11) and (12). Only the matrix $X_1$ is computed, and the matrix $X_2$ is created dynamically to perform necessary matrix operations, as explained in Section 4.7. Here, we compute two estimates, the maximum likelihood estimate and the penalized maximum likelihood estimate using the group bridge penalty. Recall that this penalty function outperforms others in the first-order analysis based on the EBIC criterion.

In Figure 6, the functional estimates of the first-order kernels $\psi^{(1)}_i$ are very close to the first-order counterparts as depicted in Figures 3–5. The local sparsity in the second-order model is more pronounced. The estimated interaction components, that is, the second-order kernels, are presented in Figure A of the supplementary materials. Only four second-order kernels are shown, $\psi_{11}$, $\psi_{37}$, $\psi_{57}$, and $\psi_{67}$. The remaining kernels exhibit global sparsity, which suggests no interaction exists between corresponding input neurons or within the same input neurons at different time lags.

To examine the computational performance in big data context, we implement our proposed method in different hardware environments. Computing times for second-order model fit are presented in Table 1. Columns 2–4 show the approximate number of matrix–vector operations required to achieve the acceptable numerical approximation, for each method used in this article. These numbers are data dependent but are reported to show the overall computational complexity. Columns 5–10 provide...
Figure 3. Estimated kernels with sparsity in the first-order model. Unpenalized estimation (solid) and penalized maximum likelihood estimation using a group bridge penalty (dashed), SCAD (dotted), and LASSO (dash-dotted) of $\psi$ functionals in the first-order model corresponding to the input neurons: $x_1, x_2, x_4, x_6$. The estimates are comparable, and only a zoom-in view highlights differences in sparsity estimation.

Figure 4. Estimated kernels without sparsity in the first-order model. Maximum likelihood estimation (solid) and penalized maximum likelihood estimation using a group bridge penalty (dashed), SCAD (dotted), and LASSO (dash-dotted) of $\psi$ functionals in the first-order model corresponding to the input neurons: $x_3, x_5, x_7$. No sparsity is detected.

Figure 5. Estimated kernel for auto-regressive component in the first-order model. Unpenalized estimation (solid) and a group bridge estimation (dashed) of a $\psi$ functional in the first-order model corresponding to the output neuron. SCAD (dotted) and LASSO (dash-dotted) estimates also included but not clearly visible. Local sparsity is not detected.

5.4. Model Validation

Model validation for neuron spike train data in functional dynamical settings is a challenging task and is still an area of

exact timings for the matrix–vector operations in different hardware configurations. The results are rather unfavorable for the Hadoop cluster. This may partially be because the performance test was conducted in R, which is not a very efficient computing platform. The Spark test uses Python, which is known to be more efficient at matrix operations, but even those results cannot match the in-memory multicore operations, which are executed using R and C. As the results make clear, the penalty for distributing data across multiple servers is very severe. The computational cost of recreating the data (e.g., recreating the matrix columns that constitute a Kronecker product of the existing columns) is much smaller than the cost of distributed communication. However, the clusters we used are very small. Larger clusters will drive the performance numbers down. On the other hand, multicore servers with 250 and more cores are becoming available, and that should make the in-memory identification of second- and higher-order models faster and feasible.
active research. Brown et al. (2002) proposed to use a time-rescaling theorem to evaluate goodness of fit of a neural spike model. The method relies on the estimate of the spike rate (a.k.a. intensity) function for continuous time. The intensity function, integrated between two adjacent observed output spikes, is assumed to follow an exponential distribution with a unit rate. Thus, after an appropriate transformation to a uniform distribution, the Kolmogorov–Smirnov (KS) test can be used to compare against a reference uniform distribution and thus test the goodness of fit.

It is more practical to consider time as a discrete rather than a continuous variable. Haslinger, Pipa, and Brown (2010) focused on a discretized model for neuron spikes, in which time is divided into short intervals or bins, and a spike train encoded as a binary sequence of 0’s and 1’s, corresponding to whether a spike was observed in a given bin. Those authors pointed out that the KS test in Brown et al. (2002) may exhibit biases for a discretized approach. The biases are caused by physical and numerical constraints. For instance, neural spikes are not instantaneous events, and in fact are believed to last about 1 msec. In addition, the bin length creates a lower bound on the interspike interval. For the discretization approach, the distribution of interspike intervals is geometric rather than exponential, and the exponential approximation may not be valid for large firing probabilities. Haslinger, Pipa, and Brown (2010) further suggested an improved rescaling of estimated firing probabilities for bias correction.

In our analysis, we adopt the discrete time rescaling theorem (Haslinger, Pipa, and Brown 2010). We transform the estimated probabilities $\theta(t)$ to the uniform distribution as suggested by these authors. The resulting KS plot is shown as Figure B in the supplementary materials. It can be seen that the overall model fit is satisfactory. The transformed data, depicted as a dashed black line, fall within the 95% confidence bounds of a standard uniform distribution in all but a short interval near zero. Small bias is visible for low probabilities, which is expected due to the discretization of the time interval and the dependence on the spike history. This phenomenon was previously reported in Haslinger, Pipa, and Brown (2010). In fact, for the KS plot generated from the true model, such biases are also visible.

**Supplementary Materials**

**Document**: This document includes pseudo-code for computations and additional images for the second-order model and model validation. (pdf file)

**Acknowledgments**

The authors are very grateful to the editor, associate editor, and two referees for their constructive comments and suggestions. The research of Dong Song was supported by NSF INSPIRE CBET-1343193 and DARPA N66001-14-C-4016. The research of Haonan Wang was partially supported by NSF grants DMS-1106975 and DMS-1521746.

**References**

Akaike, H. (1973), “Information Theory and an Extension of the Maximum Likelihood Principle,” in *Proceedings of the 2nd International Symposium on Information Theory*, pp. 267–281. [325]

Antoniou, A., and Lu, W. (2007), *Practical Optimization*, New York: Springer. [324]

Brown, E., Barbieri, R., Ventura, V., Kass, R., and Frank, L. (2002), “The Time-Rescaling Theorem and Its Application to Neural Spike Train Data Analysis,” *Neural Computation*, 14, 325–346. [328]

Cannady, J. (1998), “Artificial Neural Networks for Misuse Detection,” in *National Information Systems Security Conference*, pp. 368–381. [319]

Chen, J., and Chen, Z. (2008), “Extended Bayesian Information Criteria for Model Selection With Large Model Spaces,” *Biometrika*, 95, 759–771. [325]

Dean, J., and Ghemawat, S. (2008), “Mapreduce: Simplified Data Processing on Large Clusters,” *Communications of the ACM*, 51, 107–113. [325]

Debar, H., Becker, M., and Siboni, D. (1992), “A Neural Network Component for an Intrusion Detection System,” in *Proceedings of 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, pp. 240–250. [319]

Donoho, D. L., and Johnstone, I. M. (1995), “Adapting to Unknown Smoothness via Wavelet Shrinkage,” *Journal of the American Statistical Association*, 90, 1200–1224. [325]

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004), “Least Angle Regression,” *The Annals of Statistics*, 32, 407–499. [324]

Fan, J., Han, F., and Liu, H. (2014), “Challenges of Big Data Analysis,” *National Science Review*, 1, 293–314. [320]

Fan, J., and Li, R. (2001), “Variable Selection via Nonconcave Penalized Likelihood and Its Oracle Properties,” *Journal of the American Statistical Association*, 96, 1348–1360. [322,324]
Flake, R. H. (1963), "Volterra Series Representation of Nonlinear Systems," Transactions of the American Institute of Electrical Engineers, Part II: Applications and Industry, 81, 330–335. [319]
Friedman, J., Hastie, T., H"olling, H., and Tibshirani, R. (2007), "Pathwise Coordinate Optimization," The Annals of Applied Statistics, 1, 302–332. [324]
Friedman, J., Hastie, T., and Tibshirani, R. (2010), "Regularization Paths for Generalized Linear Models via Coordinate Descent," Journal of Statistical Software, 33, 1–22. [324]
Fuchs, A. (2013), Nonlinear Dynamics in Complex Systems, New York: Springer. [319,321]
Green, P. J. (1984), "Iteratively Reweighted Least Squares for Maximum Likelihood Estimation, and Some Robust and Resistant Alternatives," Journal of the Royal Statistical Society, Series B, 46, 149–192. [322]
Haslinger, R., Pipa, G., and Brown, E. (2010), "Discrete Time Rescaling Theorem: Determining Goodness of Fit for Discrete Time Statistical Models of Neural Spiking," Neural Computation, 22, 2477–2506. [328]
He, B. (2005), Neural Engineering, New York: Springer. [321,322]
Huang, J., Ma, S., Xie, H., and Zhang, C. (2009), "A Group Bridge Approach for Variable Selection," Biometrika, 2, 339–355. [322,324,325]
Karmakar, S. B. (1979), "Solution of Nonlinear Differential Equations by Using Volterra Series," Indian Journal of Pure and Applied Mathematics, 10, 421–425. [319]
Komarek, P. (2004), "Logistic Regression for Data Mining and High-Dimensional Classification," Robotics Institute, 1–222. [324]
Korenberg, M. J., and Hunter, I. W. (1986), "The Identification of Nonlinear Biological Systems: LNL Cascade Models," Biological Cybernetics, 55, 125–134. [320]
——— (1996), "The Identification of Nonlinear Biological Systems: Volterra Kernel Approaches," Annals of Biomedical Engineering, 24, 250–268. [320]
Marmarelis, V. Z. (1993), "Identification of Nonlinear Biological Systems Using Laguerre Expansions of Kernels," Annals of Biomedical Engineering, 21, 573–589. [319,320]
McCullagh, P., and Nelder, J. A. (1983), Generalized Linear Models (Vol. 37), London: Chapman and Hall. [322,323]
Mizuseki, K., Sirota, A., Pastalkova, E., and Buzsáki, G. (2009a), "Theta Oscillations Provide Temporal Windows for Local Circuit Computation in the Entorhinal-Hippocampal Loop," Neuron, 64, 267–280. [320]
——— (2009b), "Multi-Unit Recordings From the Rat Hippocampus Made During Open Field Foraging," available at https://www.openaire.eu/search/dataset?datasetId=datacite___:1218e321b4dd3826af8cfd29e52aab33. [320]
Percival, D. (2012), "Theoretical Properties of the Overlapping Groups Lasso," Electronic Journal of Statistics, 6, 269–288. [324]
Schenk-Hoppé, K. R. (2001), "Random Dynamical Systems in Economics," Stochastics and Dynamics, 1, 63–83. [319]
Schatzen, M. (1981), "Nonlinear System Modeling Based on the Wiener Theory," Proceedings of the IEEE, 69, 1557–1573. [319]
Schwarz, G. (1978), "Estimating the Dimensions of a Model," The Annals of Statistics, 6, 461–464. [325]
Song, D., Chan, R. H., Marmarelis, V. Z., Hampson, R. E., Deadwyler, S. A., and Berger, T. W. (2007), "Nonlinear Dynamic Modeling of Spike Train Transformations for Hippocampal-Cortical Prostheses," IEEE Transactions on Biomedical Engineering, 54, 1053–1066. [320,321,322]
——— (2009), "Nonlinear Modeling of Neural Population Dynamics for Hippocampal Prostheses," Neural Networks, 22, 1340–1351. [320]
Song, D., Wang, H., Tu, C. Y., Marmarelis, V. Z., Hampson, R. E., Deadwyler, S. A., and Berger, T. W. (2013), "Identification of Sparse Neural Functional Connectivity Using Penalized Likelihood Estimation and Basis Functions," Journal of Computational Neuroscience, 35, 335–357. [320,321,326]
Tibshirani, R. (1996), "Regression Shrinkage and Selection via the Lasso," Journal of the Royal Statistical Society, Series B, 58, 267–288. [320,322,324]
Tu, C. Y., Song, D., Breidt, F. J., Berger, T. W., and Wang, H. (2012), "Functional Model Selection for Sparse Binary Time Series With Multiple Inputs," in W.R. Bell, S.H. Holan, and T.S. McElroy (eds.), Economic Time Series: Modeling and Seasonality, Boca Raton, FL: CRC Press, pp. 477–495. [320,322]
Vallacher, R. R., and Nowak, A. E. (1994), Dynamical Systems in Social Psychology, San Diego, CA: Academic Press. [319]
Volterra, V. (1930), "Theory of Functionals and of Integral and Integro," Differential Equations, 147–149. [319]
Wang, H., and Kai, B. (2015), "Functional Sparsity: Global Versus Local," Statistica Sinica, 25, 1337–1354. [322]
Wiener, N. (1958), Nonlinear Problems in Random Theory (Vol. 1), Cambridge, MA: The MIT Press, pp. 142. [319]
Wray, J., and Green, G. G. (1994), "Calculation of the Volterra Kernels of Non-Linear Dynamic Systems Using an Artificial Neural Network," Biological Cybernetics, 71, 187–195. [320]
Yuan, M., and Lin, Y. (2006), "Model Selection and Estimation in Regression With Grouped Variables," Journal of the Royal Statistical Society, Series B, 68, 49–67. [322]
Zou, H., and Li, R. (2008), "One-Step Sparse Estimates in Nonconcave Penalized Likelihood Models," Annals of Statistics, 36, 1509–1533. [324]