Published in final edited form as:
Neural Comput. 2009 July ; 21(7): 1797–1862. doi:10.1162/neco.2009.06-08-799.

Discrete- and Continuous-Time Probabilistic Models and Algorithms for Inferring Neuronal UP and DOWN States

Zhe Chen,
Neuroscience Statistics Research Laboratory, Department of Anesthesia and Critical Care, Massachusetts General Hospital, Harvard Medical School, Boston, MA 02114, U.S.A., and Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A

Sujith Vijayan,
Program in Neuroscience, Harvard University, Cambridge, MA 02139, U.S.A., and Picower Institute for Learning and Memory, Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A

Riccardo Barbieri,
Neuroscience Statistics Research Laboratory, Department of Anesthesia and Critical Care, Massachusetts General Hospital, Harvard Medical School, Boston, MA 02114, U.S.A

Matthew A. Wilson, and
Picower Institute for Learning and Memory, RIKEN-MIT Neuroscience Research Center, Department of Brain and Cognitive Sciences and Department of Biology, Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A

Emery N. Brown
Neuroscience Statistics Research Laboratory, Department of Anesthesia and Critical Care, Massachusetts General Hospital, Harvard Medical School, Boston, MA 02114, U.S.A., Harvard-MIT Division of Health Sciences and Technology, Cambridge, MA 02139, U.S.A., and Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A

Zhe Chen: zhechen@mit.edu; Sujith Vijayan: vijayan@post.harvard.edu; Riccardo Barbieri: barbieri@neurostat.mit.edu; Matthew A. Wilson: mwilson@mit.edu; Emery N. Brown: enb@neurostat.mit.edu

Abstract

UP and DOWN states, the periodic fluctuations between increased and decreased spiking activity of a neuronal population, are a fundamental feature of cortical circuits. Understanding UP-DOWN state dynamics is important for understanding how these circuits represent and transmit information in the brain. To date, limited work has been done on characterizing the stochastic properties of UP-DOWN state dynamics. We present a set of Markov and semi-Markov discrete- and continuous-time probability models for estimating UP and DOWN states from multiunit neural spiking activity. We model multiunit neural spiking activity as a stochastic point process, modulated by the hidden (UP and DOWN) states and the ensemble spiking history. We estimate jointly the hidden states and the model parameters by maximum likelihood using an expectation-maximization (EM) algorithm and a Monte Carlo EM algorithm that uses reversible-jump Markov chain Monte Carlo sampling in the E-step. We apply our models and algorithms in the analysis of both simulated multiunit spiking activity and actual multiunit spiking activity recorded from primary somatosensory cortex in a behaving rat during slow-wave sleep. Our approach provides a statistical characterization of UP-DOWN state dynamics that can serve as a basis for verifying and refining mechanistic descriptions of this process.

1 Introduction

1.1 Neuronal State and Recurrent Networks

The state of the neural system reflects the phase of an active recurrent network, which organizes the internal states of individual neurons into synchronization through recurrent network synaptic activity with balanced excitation and inhibition.[1] The neuronal state dynamics can be externally or internally driven. The externally driven dynamics results from either sensory-driven adaptation or encoding of sensory percept; the internally driven dynamics results from changes in internal factors, such as attention shift. Different levels of neuronal state also bring in the dynamics of state transition. Generally state transitions are network controlled and can be triggered by the activation of single cells, which are reflected by changes in their intracellular membrane conductance.

From a computational modeling point of view, two types of questions arise. First, how do neurons generate, maintain, and transit between different states? Second, given the neuronal (intracellular or extracellular) recordings, how can the neuronal states be estimated? The computational solutions to the first question emphasize the underlying neuronal physiology or neural mechanism, which we call mechanistic models, whereas the solutions to the second question emphasize the representation or interpretation of the data, which we call statistical models. In this article, we are taking the second approach to model a specific phenomenon regarding the neuronal state.

1.2 Neuronal UP and DOWN States

The notion of neuronal UP and DOWN states refers to the observation that neurons have two distinct subthreshold membrane potentials that are relevant for action potential (i.e., spike) generation. A neuron is said to be depolarized (or excited) if its intracelluar membrane potential is above the resting membrane potential threshold (around −70 to −80 mV) and is said to be hyperpolarized (or inhibited) if its membrane potential is below the threshold. When a sufficient level of excitation is reached, a spike is likely to occur. Essentially, membrane potential fluctuations define two states of the neocortex. The DOWN state defines a quiescent period during which little or no activity occurs, whereas the UP state corresponds to an active cortical state with depolarized membrane potentials and action potential firing driven by synaptic input. It was generally believed that the spontaneous UP and DOWN states are generated by a balance of excitatory and inhibitory neurons in recurrent networks (Haider, Duque, Hasentaub, & McCormick, 2006). In recent years, many neurophysiological studies have been reported regarding the neuronal UP and DOWN states, ranging from intracellular or extracellular recordings (e.g., Sanchez-Vives & McCormick, 2000; Haider, Duque, Hasentaub, Yu, & McCormick, 2007). The UP and DOWN states that are characterized by the cortical slow oscillation in intracellular membrane potentials are also reflected in extracellular recordings, such as local field potential (LFP) or electroencephalograph (EEG), single-unit activity or multiunit activity (MUA). In the literature, the UP and DOWN states have been characterized by examining extracellular LFP recordings (Sirota, Csicsvari, Buhl, & Buzsáki, 2003; Battaglia, Sutherland, & McNaughton, 2004; Wolansky, Clement, Peters, Palczak, & Dickson, 2006) in either the somatosensory cortex of anesthetized or awake animals (e.g., Haslinger, Ulbert, Moore, Brown, & Devor, 2006; Luczak, Barthó, Marguet, Buzsáki, & Harris, 2007) or the visual cortex of nonanesthetized animals during sleep (e.g., Ji & Wilson, 2007). Recently, attention has also turned to multiunit spike trains in an attempt to relate spike firing activities with EEG recordings (Ji & Wilson, 2007).

[1]The neuronal state sometimes can refer to a single cell level, during which neurons exhibit different lengths or durations of depolarizing shift (e.g., Fujisawa, Matsuki, & Ikegaya, 2005).

In order to examine the relationship between sleep and memory in rats or animals, simultaneous recordings are often conducted in the neocortex and hippocampus with the goal of studying the cortico-hippocampal circuit and the functional connectivity of these two regions while the animals perform different tasks. It has been reported (e.g., Volgushev, Chauvette, Mukovski, & Timofeev, 2006) that during the slow wave sleep (SWS), which is characterized by 0.5 to 2.0 Hz slow oscillations (Buzsáki, 2006), neorcortical neurons undergo near-synchronous transitions, every second or so, between UP and DOWN states. The process of the alternating switch between the two states appears to be a network phenomenon that originates in the neocortex (Ji & Wilson, 2007; Vijayan, 2007).

The work reported here was driven by the experimental data accumulated in our lab (Ji & Wilson, 2007; Vijayan, 2007). The growing interest in UP and DOWN states in the neuroscience literature motivated us to develop probabilistic models for the UP and DOWN modulated MUA. Specifically, the UP-DOWN states are modeled as a latent two-state Markovian (or semi-Markovian) process (Battaglia et al., 2004), and the modeling goal is to establish the probability for state transition or the probability density of UP or DOWN state duration and the likelihood model that takes into account both a global hidden state variable and individual history dependence of firing. In comparison with the standard and deterministic threshold-based method, our proposed stochastic models provide a means for representing the uncertainty of state estimation given limited experimental recordings.

1.3 Markov and Semi-Markov Processes and Hidden Markov Models

A stochastic process is said to be Markovian if it satisfies the Markov property; the knowledge of the previous history of states is irrelevant for the current and future states. A Markov chain is a discrete-time Markov process with the Markov property. The Markov process and Markov chain are both “memoryless.” A Markov process or Markov chain contains either continuous-valued or finite discrete-valued states. A discrete-state Markov process contains a finite alphabet set (or finite state space), with each element representing a distinct discrete state. The change of the state is called the transition, and the probability of changing from one state to the other is called the transition probability. For the Markov chain, the current state has only finite-order dependence on the previous states. Typically the first-order Markov property is assumed; in this case, the probability of S_{k+1} being in a particular state at time $k + 1$, given knowledge of states up to time k, depends on the state S_k at time k, namely, $\Pr(S_{k+1} \mid S_0, S_1, \ldots, S_k) = \Pr(S_{k+1} \mid S_k)$. A semi-Markov process (the Markov renewal process) extends the continuous-time Markov process to the condition that the interoccurrence times are not exponential.

When the state space is not directly observable, a Markov process is called hidden or latent. The so-called hidden Markov process is essentially a probabilistic function of the stochastic process (for a review, see Ephraim & Merhav, 2002). In the discrete-time context, the hidden Markov model (HMM) is a probabilistic model that characterizes the hidden Markov chain. The HMM is a generative model in that its full model $\{\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{B}\}$ (where π denotes the initial state probability, $\boldsymbol{P}$ denotes the transition probability, and $\boldsymbol{B}$ denotes the emission probability) completely characterizes the underlying probabilistic structure of the Markov chain. Generally several conditions are assumed in the standard HMM: (1) the transition and emission probabilities are stationary or quasi-stationary; (2) the observations, either continuous or discrete valued, are assumed to be identically and independently distributed (i.i.d.); and (3) the model generally assumes a first- or finite-order Markov property. In the literature, there are several methods to tackle the inference problem in the HMM. One (and maybe the most popular) approach is rooted in maximum likelihood estimation. A particular solution is given by the expectation-maximization (EM) algorithm (Dempster, Laird, & Rubin, 1977), which attempts to solve the missing data problem in the statistics literature. This turns out to be also equivalent to the Baum-Welch algorithm proposed by Baum and Welch and colleagues (Baum, Petrie, Soules, & Weiss, 1970; Baum, 1972). The Baum-Welch algorithm contains a forward-backward procedure (E-step) and reestimation (M-step), and it iteratively increases the likelihood of the incomplete data until the local maximum or the stationary point of the likelihood function is reached. Another inference method is rooted in Bayesian statistics. The Bayesian inference for HMM defines the prior probability for the unknown parameters (including the number of state) and attempts to estimate their posterior distributions. Since the posterior distribution is usually analytically intractable, a numerical approximation method is also used. The Markov chain Monte Carlo (MCMC) algorithms try to simulate a Markov chain to approach the equilibrium of the posterior distribution. The Metropolis-Hastings algorithm is a general MCMC procedure to simulate a Markov or semi-Markov chain. When the state space is transdimensional (this problem often arises from model selection in statistical data analysis), the reversible-jump MCMC (RJMCMC) methods (Green, 1995; Robert, Rydén, & Titterington, 2000) have also been developed. Due to the development of efficient inference algorithms, HMM and its variants have been widely used in speech recognition, communications, bioinformatics, and many other applications (e.g., Rabiner, 1989; Durbin, Eddy, Krough, & Mitchison, 1998).

1.4 Point Process and Cox Process

A point process is a continuous-time stochastic process with observations being either 0 or 1. Spike trains recorded from either single or multiple neurons are point processes. We will give a brief mathematical background for point process in a later section and refer the reader to Brown (2005) and Brown, Barbieri, Eden, and Frank (2003) for the complete and rigorous mathematical details of point processes in the context of computational neuroscience treatment. An important feature of spike trains is that the point process observations are not independently distributed; in other words, the current observation (either 0 or 1) is influenced by the previous spiking activities. This type of history dependence requires special attention for probabilistic modeling of the point process.

A Cox process is a doubly stochastic process, which defines a generalization of Poisson process (Cox & Isham, 1980; Daley & Vere-Jones, 2002). Specifically, the time-dependent conditional intensity function (CIF), often denoted as λ_t, is a stochastic process by its own.[2] A representative example of the Cox process is the Markov-modulated Poisson process, which has a state-dependent Poisson rate parameter.

1.5 Overview of Relevant Literature

Hidden Markov processes have a rich history of applications in biology. Tremendous effort has been devoted to modeling ion channels as discrete- or continuous-time Markov chains; several inference algorithms were developed for these models (Chung, Krishnamurthy, & Moore, 1991; Fredkin & Rice, 1992; Ball, Cai, Kadane, & O'Hagan, 1999). However, the observations used in ion-channel modeling are continuous, and the likelihood is often modeled by a gaussian or gaussian mixture distribution.

For discrete observations, Albert (1991) proposed a two-state Markov mixture model of a counting Poisson process and provided a maximum likelihood estimate (MLE) for the parameters. A more efficient forward-backward algorithm was later proposed by Le, Leroux, and Puterman (1992) with the same problem setup. In these two models, the two-state Markov transition probability is assumed to be stationary; although Albert also pointed out the possibility of modeling nonstationarity, no exact algorithm was given. In addition, efficient

[2]The CIF is also known as the *hazard rate function* in survival analysis. The value $\lambda_t \Delta$ measures the probability of a failure or death of an event in $[t, t + \Delta)$ given the process has survived up to time t.

EM algorithms have been developed for discrete- or continuous-time Markov-modulated point processes (Deng & Mark, 1993; Rydén, 1996; Roberts, Ephraim, & Dieguez, 2006), but applying them to neural spike trains is not straightforward.

In the context of modeling neural spike trains, many authors (e.g., Radons, Becker, Dülfer, & Krüger, 1994; Abeles et al., 1995; Gat, Tishby, & Abeles, 1997; Jones, Fontanini, Sadacca, & Katz, 2007; Achtman et al., 2007; Kemere et al., 2008) used HMM for the purpose of analyzing and classifying the patterns of neural spike trains, but their models are restricted to discrete time and the Markov chain is homogeneous (i.e., the transition probability is stationary). In these studies, the hidden states are discrete, and the spike counts were used as the discrete observations for the likelihood models. Smith and Brown (2003) extended the standard linear state-space model (SSM) with continuous state and observations to an SSM with a continuous state Markov-modulated point process, and an EM algorithm was developed for the hidden state estimation problem. Later the theory was extended to the SSM with mixed continuous, binary, and point process observations (Coleman & Brown, 2006; Prerau et al., 2008; Eden & Brown, 2008), but the latent process was still limited to the continuous-valued state. In a similar context, Danóczy and Hahnloser (2006) also proposed a two-state HMM for detecting the “singing-like” and “awake-like” states of sleeping songbirds with neural spike trains; their model assumes a continuous-time Markov chain (with the assumption of knowing the exact timing of state transitions), and the sojourn time follows an exponential distribution; in addition, the CIF of the point process was assumed to be discrete in their work. All of these restricted assumptions have limited the computational model for analyzing real-world spike trains. Recently, more modeling efforts have been dedicated to estimating the hidden state and parameters using an HMM (or its variants) for estimating the stimulus-response neuronal model (Jones et al., 2007; Escola & Paninski, 2008). Xi and Kass (2008) recently also used a RJMCMC method to characterize the bursty and nonbursty states from goldfish retinal neurons.

In modeling the hidden semi-Markov processes or semi-Markov chains, in which the sojourn time is no longer exponentially distributed, Guon (2003) developed an EM algorithm for a hidden semi-Markov chain with finite discrete-state sojourn time, but the computational complexity of the EM algorithm is much greater than the conventional HMM.[3]

1.6 Contribution and Outline

In this article, with the goal of estimating the population neuron's UP or DOWN state, we propose discrete-state Markov or semi-Markov probabilistic models for neural spikes trains, which are modeled as doubly stochastic point processes. Specifically, we propose discrete-time and continuous-time SSMs and develop the associated inference algorithms for tackling the joint (state and parameter) estimation problem.

Our contributions have three significant distinctions from the published literature: (1) the point-process observations are not i.i.d. Specifically, the rate parameters or the CIFs of the spike trains are modulated by a latent discrete-state variable and past spiking history. (2) In the continuous-time probabilistic models, the state transition is not necessarily Markovian; in other words, the hidden state is semi-Markovian in the sense that the sojourn time is no longer exponentially distributed. (3) The maximum likelihood inference algorithms are derived for discrete-time and continuous-time probabilistic models for estimating the neuronal UP or DOWN states, and the proposed Monte Carlo EM (MCEM) algorithm is rooted in a RJMCMC sampling method and is well suited for various probabilistic models of the sojourn time.

[3]In the worst case, the complexity is $\mathcal{O}(NT(N+T))$ in time, in contrast to $\mathcal{O}(N^2T)$ for the HMM, where N denotes the number of discrete state and T denotes the total length of sequences.

The rest of the article is organized as follows. In section 2, we present the discrete-time HMM and the EM-based inference algorithm. In section 3, we develop the continuous-time probabilistic Markov and semi-Markov chains and their associated inference algorithms. In section 4, we demonstrate and validate our proposed models and inference algorithms with both simulated and real-world spike train data. We present some discussions in section 5, followed by the conclusion in section 6.

1.7 Notation

In neural spike analysis, we examine spike trains from either single or multiunit activity. Due to digitalized recordings, we assume that the time interval $[0, T]$ of continuous-time neural spike train observations is properly discretized with a small time resolution Δ, so the time indexes are discrete integers within $k \in [1, K]$, such that $k\Delta \in ((k-1)\Delta, k\Delta]$ and $K\Delta = T$. Let $N^c_{t_k} \equiv N^c_k$ denote the counting process for spike train c at time t_k, and let $dN^c_{t_k} \equiv dN^c_k$ denote the indicator variable for 0/1 observation: $dN^c_k = 1$ if there is a spike and 0 otherwise. Other notations are rather straightforward, and we will define them in the proper places. Most notations used in this article are summarized in Table 1.

2 Discrete-Time Markov Modulated Probabilistic State-Space Model

To infer the neuronal UP and DOWN states, in this section we develop a simple, discrete-time Markov modulated state-space model that can be viewed as a variant of the standard HMM applied to spike train analysis. The underlying probabilistic structure is Markovian and homogeneous, and the inference algorithm is efficient in identifying the statistics of the hidden state process. Based on that, in the next section we develop a continuous-time probabilistic model in order to overcome some of limitations imposed by this discrete-time probabilistic model.

2.1 Hidden Markov Model

Let us consider a discrete-time homogeneous Markov chain. By discrete time, we assume that the time is evenly discretized into fixed-length intervals, which have time indices $k = 1, \dots, K$. The neuronal UP or DOWN state, which is characterized by a latent discrete-time first-order Markov chain, is unobserved (and therefore hidden), and the observed spike trains or the spike counts recorded from the MUA are functionally determined by the hidden state. The standard HMM is characterized by three elements: transition probability, emission probability,[4] and initial state probability (Rabiner, 1989). At the first approximation, we assume that the underlying latent process follows a two-state HMM with stationary transition and emission probabilities.

- The initial probability of state is denoted by a vector $\boldsymbol{\pi} = \{\pi_i\}$, where $\pi_i = \Pr(S_0 = i)$ $(i = 0, 1)$. Without loss of generality, we assume that the amplitude of the hidden state is predefined, and the discrete variable $S_k \in \{0, 1\}$ indicates either a DOWN (0) or UP (1) state.
- The transition probability matrix is written as

$$\mathbf{P} = \begin{pmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{pmatrix}, \tag{2.1}$$

[4]The term *emission probability* arose from the HMM literature in the context of speech recognition; it refers to the probability of observing a (finite) symbol given a hidden state (finite alphabet).

with $P_{01} = 1 - P_{00}$ and $P_{10} = 1 - P_{11}$ corresponding to the transition probabilities from state 0 to state 1 and from state 1 to state 0, respectively.

- Given the discrete hidden state S_k, the observed numbers of total spikes across all tetrodes (i.e., MUA), $y_1, y_2, \ldots, y_K$ ($y_k \in \mathbb{N}$), follow probability distributions that depend on the time-varying rate λ_k,

$$\Pr(Y_k = y_k | S_k = i) = \frac{e^{-\lambda_k}\lambda_k^{y_k}}{y_k!}, \tag{2.2}$$

where the parameter λ_k is determined by

$$\lambda_k = \exp(\mu + \alpha S_k + \beta(N_{k-1} - N_{k-J})), \tag{2.3}$$

where $\exp(\mu)$ denotes the baseline firing rate and S_k denotes the hidden discrete-state variable at time k. The term $N_{k-1} - N_{k-J}$ represents the total number of spikes observed during the history period $(k - J, k - 1]$, which accounts for the history dependence of neuronal firing. The choice of the length of history dependence is often determined empirically based on the preliminary data analysis, such as the histogram of the interspike interval (ISI). Equations 2.2 and 2.3 can be understood in terms of a generalized linear model (GLM) (e.g., McCullagh & Nelder, 1989; Truccolo, Eden, Fellow, Donoghue, & Brown, 2005), where the link function is a log function and the distribution is Poisson. Note that when $\beta = 0$ (i.e., history independence is assumed), we obtain an inhomogeneous Poisson process, and λ_k reduces to a Poisson rate parameter.

Taking the logarithm to both sides of equation 2.2, equation 2.3 can be rewritten as

$$\log\lambda_k = \mu + \alpha S_k + \beta\tilde{n}_k, \tag{2.4}$$

where $\tilde{n}_k = N_{k-1} - N_{k-J}$. More generally, we can split the time period $(k - J, k - 1]$ into several windows (say, with equal duration δ), and equation 2.4 can be rewritten as

$$\begin{aligned} \log\lambda_k &= \mu + \alpha S_k + \sum_j \beta_j \tilde{n}_{k,j} \\ &= \mu + \alpha S_k + \boldsymbol{\beta}^T \tilde{\mathbf{n}}_k, \end{aligned} \tag{2.5}$$

where $\boldsymbol{\beta} = \{\beta_j\}$ and $\tilde{\mathbf{n}}_k = \{\tilde{n}_{k,j}\}$ are two vectors with proper dimensionality, and $\tilde{n}_{k,j} = N_{k-j\delta} - N_{k-(j+1)\delta}$ denotes the observed number of multiunit spike counts within the time interval $(k - (j + 1)\delta, k - j\delta]$. If we further assume that the observations y_k at different time indices k are mutually independent, the observed data likelihood is given by

$$p(y_{1:K} | S_{1:K}, \boldsymbol{\theta}) = \prod_{k=1}^{K} \frac{\exp(-\lambda_k)\lambda_k^{y_k}}{y_k!}. \tag{2.6}$$

Note that $\lambda_k \equiv \lambda(S_k)$ is functionally dependent on the latent process S_k, although we have omitted it from the notation for brevity. In statistics, the hidden variables $\{S_k\}$ are treated as the missing data, $\{y_k\}$ as the observed (incomplete) data, and their combination $\{S_k, y_k\}$ as the complete data. Let $\boldsymbol{\theta}$ denote all of the unknown parameters; then the complete data likelihood is given by

$$p(S_{1:K}, y_{1:K}|\boldsymbol{\theta}) = p(y_{1:K}|S_{1:K}, \boldsymbol{\theta})p(S_{1:K}|\boldsymbol{\theta}). \tag{2.7}$$

And the complete data log likelihood, denoted as $\mathscr{L}$, is derived as (by ignoring the constant)

$$\mathscr{L} = \log p(S_{0:K}, y_{1:K}|\boldsymbol{\theta}) = \sum_{k=1}^{K}(y_k \log \lambda_k - \lambda_k) + \sum_{i=0}^{1} i \log \pi_i + \sum_{k=2}^{K}\sum_{i=0}^{1}\sum_{j=0}^{1}\xi_k(i, j)\log P_{ij}, \tag{2.8}$$

where $\xi_k(i, j) = \Pr(S_{k-1} = i, S_k = j)$.

2.2 Forward-Backward and Viterbi Algorithms

The inference and learning procedure for the standard HMM is given by an efficient estimation procedure known as the EM algorithm, which is also known as the Baum-Welch algorithm (Baum et al., 1970; Baum, 1972). Rooted in maximum likelihood estimation, the EM algorithm iteratively and monotonically maximizes (or increases) the log-likelihood function given the incomplete data (Dempster et al., 1977). In the E-step, a forward-backward procedure is used to recursively estimate the hidden state posterior probability. In the M-step, based on the missing state statistics (estimated from the E-step), the reestimation procedure and Newton-Ralphson algorithm are used to estimate the unknown parameters $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{P}, \mu, \alpha, \beta)$. In each full iteration, the EM algorithm iteratively maximizes the so-called Q-function,

$$\begin{aligned} Q(\boldsymbol{\theta}^{new}|\boldsymbol{\theta}^{old}) &= \mathbb{E}[\log p(\widehat{S}_{1:K}, y_{1:K}|\boldsymbol{\theta})|\boldsymbol{\theta}^{old}] \\ &= \mathbb{E}\left[\sum_{k=1}^{K}(y_k \log \widehat{\lambda}_k - \widehat{\lambda}_k) + \sum_{i=0}^{1} i \log \widehat{\pi}_i \right. \\ &\quad \left. + \sum_{k=2}^{K}\sum_{i=0}^{1}\sum_{j=0}^{1}\xi_k(i, j)\log \widehat{P}_{ij}|\boldsymbol{\theta}^{old}\right]; \end{aligned} \tag{2.9}$$

the new $\boldsymbol{\theta}^{new}$ is obtained by maximizing the incomplete data likelihood conditional on the old parameters $\boldsymbol{\theta}^{old}$; and the iterative optimization procedure continues until the algorithm ultimately converges to a local maximum or a stationary point. For the self-containing purpose, we present a brief derivation of the EM algorithm (Rabiner, 1989) for the two-state HMM estimation problem.

2.2.1 E-Step: Forward-Backward Algorithm—

In the E-step, the major task of the forward-backward procedure is to compute the conditional state probabilities for the two states:

$$\begin{aligned} \Pr(S_k = 1|y_{1:K}, \boldsymbol{\theta}) &= \frac{\Pr(y_{1:K}, S_k = 1|\boldsymbol{\theta})}{\Pr(y_{1:K}|\boldsymbol{\theta})} \\ &= \frac{\Pr(y_{1:K}, S_k = 1|\boldsymbol{\theta})}{\sum_{l=0}^{1}\Pr(y_{1:K}, S_k = l|\boldsymbol{\theta})} \end{aligned} \tag{2.10}$$

$$\Pr(S_k=0|y_{1:K},\boldsymbol{\theta})=1-\Pr(S_k=1|y_{1:K},\boldsymbol{\theta}), \tag{2.11}$$

as well as the conditional state joint probability:

$$\begin{aligned}\Pr(S_{k-1}=i,\ S_k=j|y_{1:K},\boldsymbol{\theta})\\ =\frac{\Pr(y_{0:K},S_{k-1}=i,S_k=j|\boldsymbol{\theta})}{\Pr(y_{1:K}|\boldsymbol{\theta})}\\ =\frac{\Pr(y_{0:K},S_{k-1}=i,S_k=j|\boldsymbol{\theta})}{\sum_{l=0}^{1}\sum_{m=0}^{1}\Pr(y_{1:K},S_{k-1}=l,S_k=m|\boldsymbol{\theta})}.\end{aligned} \tag{2.12}$$

To make the notation simple, in the derivation below, we let the conditional $\boldsymbol{\theta}$ be implicit in the equation.

To estimate equations 2.10 and 2.11, we first factorize the joint probability as

$$\begin{aligned}\Pr(y_{1:K},\ S_k=l)=&\Pr(y_{1:k},\ S_k=l)\Pr(y_{k+1:K}|y_{1:k},\ S_k=l)\\ =&\Pr(y_{1:k},\ S_k=l)\Pr(y_{k+1:K}|y_{1:k}\ S_k=l)\\ \equiv&\ a_k(l)\,b_k(l)\quad\text{for }\ l=0,1,\end{aligned} \tag{2.13}$$

where

$$\begin{aligned}&a_k(l)=\Pr(y_{1:k},\ S_k=l)\quad\text{for }\ k=2,\dots,K\\ &a_1(l)=\Pr(y_{1:k},\ S_k=l)\Pr(y_1|S_1=l)\\ &b_k(l)=\Pr(y_{k+1:K}|S_k=l)\quad\text{for }\ k=1,\dots,K-1\\ &b_1(l)=1,\end{aligned}$$

and the forward and backward messages $a_k(l)$ and $b_k(l)$ can be computed recursively along the discrete-time index k (Rabiner, 1989):

$$\begin{aligned}a_k(l)=&\sum a_{k-1}(i)\,P_{il}\Pr(y_k|S_k=l)\\ =&\sum a_{k-1}(i)\,P_{il}\frac{\exp(-\lambda_k)\lambda_k^{y_k}}{y_k!}\\ b_k(l)=&\sum b_{k+1}(i)\,P_{li}\Pr(y_{k+1}|S_{k+1}=i)\\ =&\sum b_{k+1}(i)\,P_{li}\frac{\exp(-\lambda_{k+1})\lambda_{k+1}^{y_{k+1}}}{y_{k+1}!},\end{aligned}$$

where P_{il} denotes the transition probability from state i to l.

Given $\{a_k, b_k\}$, we can estimate equation 2.12 by

$$\Pr(S_{k-1}=i,\ S_k=j|y_{1:K},\ \boldsymbol{\theta})=a_k(i)\,P_{ij}\Pr(y_{k+1}|S_{k+1}=j)b_{k+1}(j). \tag{2.14}$$

Furthermore, we can compute the observed likelihood (of the incomplete data) by

$$p(y_{1:K})=\sum_{l=0}^{1}\Pr(y_{1:K},\ S_K=l)=\sum_{l=0}^{1}a_K(l). \tag{2.15}$$

Given equations 2.13 and 2.15, the state posterior conditional probability is given by Bayes' rule,

$$\Pr(S_k=i|y_{1:K})=\frac{\Pr(S_k=i,\ y_{1:K})}{p(y_{1:K})}=\frac{a_k(i)b_k(i)}{\sum_{l=0}^{1}a_K(l)}. \tag{2.16}$$

In the term of the computational overhead for the above-described two-state HMM, the forward-backward procedure requires a linear order of computational complexity $\mathcal{O}(4K)$ and memory storage $\mathcal{O}(2K)$.

2.2.2 M-Step: Reestimation and Newton-Ralphson Algorithm—

In the M-step, we update the unknown parameters (based on their previous estimates) by setting the partial derivatives of the Q-function to zeros: $\frac{\partial Q(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}=0$, from which we may derive either closed-form or iterative solutions.

Let $\xi_k(i, j) = \Pr(S_{k-1} = i, S_k = j \mid y_{1:K}, \boldsymbol{\theta})$ and $\gamma_k(i) = \Pr(S_k = i \mid y_{1:K}, \boldsymbol{\theta})$ denote, respectively, the conditional marginal and joint state probabilities (which are the sufficient statistics for the complete data log likelihood 2.9). From the E-step, we may obtain

$$\gamma_k(i)=\frac{a_k(i)b_k(i)}{\sum_{i=0}^{1}a_k(l)b_k(l)}=\sum_j\xi_k(j,i)=\sum_j\xi_{k+1}(i,\ j). \tag{2.17}$$

The transition probabilities are given by Baum's reestimation procedure:

$$\widehat{P}_{ij}=\frac{\sum_{k=2}^{K}\xi_k(i,\ j)}{\sum_{k=2}^{K}\sum_j\xi_k(i,\ j)}=\frac{\sum_{k=2}^{K}\xi_k(i,\ j)}{\sum_{k=2}^{K}\gamma_k(i)}. \tag{2.18}$$

Specifically the transition probabilities P_{01} and P_{10} are estimated by closed-form expressions,

$$\widehat{P}_{01}=\frac{\sum_{k=2}^{K}\xi_k(0,1)}{\sum_{k=2}^{K}\sum_{j=0}^{1}\xi_k(0,\ j)}=\frac{\sum_{k=2}^{K}\xi_k(0,1)}{\sum_{k=2}^{K}\gamma_k(0)}, \tag{2.19}$$

$$\widehat{P}_{10}=\frac{\sum_{k=2}^{K}\xi_k(1,0)}{\sum_{k=2}^{K}\sum_{j=0}^{1}\xi_k(1,\ j)}=\frac{\sum_{k=2}^{K}\xi_k(1,0)}{\sum_{k=2}^{K}\gamma_k(1)}. \tag{2.20}$$

Next, we need to estimate the other unknown parameters (μ, α, β) that appear in the likelihood model. Since there is no closed-form solution for μ, α, and β, we may use the iterative optimization methods, such as the Newton-Ralphson algorithm or the iterative weighted least squares (IWLS) algorithm (e.g., Pawitan, 2001), to optimize the parameters in the M-step.

Let $\widehat{S}_k=\sum_{i=0}^{1} i\gamma_k(i)$ denote the computed mean statistic of a hidden state at time k; by setting the derivatives of $\mathcal{L}$ with regard to the parameters α, μ, and β (and similarly for vector $\boldsymbol{\beta}$), to zeros, we obtain

$$\sum_{k=J}^{K}\widehat{S}_k y_k=\sum_{k=J}^{K}\widehat{S}_k \exp(\mu+\alpha\widehat{S}_k+\beta\tilde{n}_k), \tag{2.21}$$

$$\sum_{k=J}^{K}\tilde{n}_k y_k=\sum_{k=J}^{K}\tilde{n}_k \exp(\mu+\alpha\widehat{S}_k+\beta\tilde{n}_k), \tag{2.22}$$

$$\sum_{k=J}^{K} y_k=\sum_{k=J}^{K} \exp(\mu+\alpha\widehat{S}_k+\beta\tilde{n}_k), \tag{2.23}$$

respectively. Typically, a fixed number of iterations is preset for the Newton-Ralphson algorithm in the internal loop within the M-step.

Finally, the convergence of the EM algorithm is monitored by the incremental changes of the log likelihood as well as the parameters. If the absolute value of the change is smaller than 10^{-5}, the EM algorithm is terminated.

2.2.3 Viterbi Algorithm—Upon estimating parameters $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{P}, \mu, \alpha, \beta)$, we can run the Viterbi algorithm (Viterbi, 1967; Forney, 1973) for decoding the most likely state sequences. The Viterbi algorithm is a dynamical programming method (Bellman, 1957) that uses the “Viterbi path” to discover the single most likely explanation for the observations. Specifically, the maximum a posteriori (MAP) state estimate $\hat{S}_k$ at time k is

$$\widehat{S}_k^{\mathrm{MAP}}=\arg\max_{i\in\{0,1\}} \gamma_k(i) \quad 1 \le k \le K. \tag{2.24}$$

The computational overhead of the forward Viterbi algorithm has an overall time complexity $\mathcal{O}(4K)$ and space complexity $\mathcal{O}(2K)$.

3 Continuous-Time Markovian and Semi-Markovian State-Space Models

The discrete-time probabilistic model discussed in the previous section imposes strong assumptions on the transition between the UP and DOWN states. First, it is stationary in the sense that the transition probability is time invariant; second, the transition is strictly Markovian. In this section, we relax these assumptions and further develop continuous-time, data-driven (either Markovian or semi-Markovian) state-space models, which is more appropriate and realistic in characterizing the nature or statistics of the state transitions. In addition, the inference algorithm for the continuous-time models uses the estimation output from the discrete-time model (developed in section 2) as the initialization condition, which also helps to accelerate the algorithmic convergence process.

3.1 Continuous-Time Probabilistic Model

One important distinction between the discrete-time and continuous-time Markov chains is that the former allows state changes to occur only at regularly spaced intervals, whereas the latter is aperiodic, in that the time between state changes is exponentially distributed. Therefore, the notion of a “single-step transition probability” is no longer valid in continuous time since the “step” does not exist. In fact, the transition probability in continuous time is characterized by either the transition rate or the sojourn time probability density function (pdf) between the two state change events. Let us assume that the pdf of the sojourn time for state *j* characterized by a parameter vector θ_j. Hence, the transition probability between state 0 (DOWN) and 1 (UP) is characterized by the corresponding pdfs $p(\theta_0, z)$ and $p(\theta_1, z)$, respectively, where *z* is now the random variable in terms of holding time. For example, given the current state status (say, state *j*) and current time *t*, the probability of escaping or changing the current state (to other different state) will be computed from the cumulative distribution function (cdf):

$$F(z) \equiv \Pr[\,0 \le z \le t] = \int_0^t p(\theta_j, z)\, dz, \tag{3.1}$$

and the probability of remaining in the present state will be computed from

$$\Pr[\,z > t] = 1 - F(z) = \int_t^\infty p(\theta_j, z)\, dz, \tag{3.2}$$

which is known as the survival function in reliability and survival analysis. As seen, the transition probability matrix in continuous time now depends on the elapsed time (starting from the state onset) as well as the present state status. In general, we write the transition probability matrix as a parameterized form $\boldsymbol{P}(\boldsymbol{\theta})$, where $\boldsymbol{\theta} = (\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$ characterizes the associated sojourn time pdf parameters for the DOWN (0) and UP (1) states. As we will see, choosing different probability density models for the sojourn time results in different formulations of the continuous-time Markov or semi-Markov chain.

In modeling the neural spike train point processes, the CIF characterizes the instantaneous firing probability of a discrete event (i.e., spike). Specifically, the product between the CIF $\lambda(t)$ and the time interval Δ tells approximately the probability of observing a spike within the interval $[t, t + \Delta)$ (e.g., Brown et al., 2003):

$$\lambda(t) = \lim_{\Delta \to 0} \frac{\Pr\{N(t+\Delta) - N(t) = 1 | \mathscr{H}_{0:t}\}}{\Delta}.$$

For each spike train, we model the CIF in a parametric form,[5]

[5]Here we assume that the individual CIF $\lambda^c(t)$ can be modeled as a GLM with $\log(\cdot)$ as a link function (Truccolo et al., 2005; Paninski, Pillow, & Lewi, 2007). One can also use other functions as the link function candidate, such as $\log(1 + \exp(\cdot))$ or the sigmoidal (bell-shaped) function, whichever better reflects the neuron's firing properties (e.g., Paninski, 2004). The choice of the functional form for the CIF does not affect the inference principle or procedure described below.

$$\lambda^c(t) \equiv \lambda^c(t|\mathscr{H}_{0:t}) = \exp\left(\mu_c + \alpha_c S_t + \gamma_c \int_0^t e^{-\beta_c \tau} dN^c(t-\tau) d\tau\right), \tag{3.3}$$

where $\exp(\mu_c)$ denotes the baseline firing rate for the cth spike train and β_c denotes an exponential decaying parameter that takes into account the history dependence of firing from time 0 upto time t. The nonnegative term $\int_0^t e^{-\beta_c \tau} dN^c(t-\tau) d\tau$ defines a convolution between the exponential decaying window and the firing history of spike train c up to time t. Because of digitalized recording devices, all continuous-time signals are sampled and recorded in digital format with a very high sampling rate (32 kHz in our setup). Therefore, we still deal with the "discretized" version of a continuous-time signal. In this case, we sometimes use S_t and S_k interchangeably if no confusion occurs. However, as we see below, their technical treatments are rather different. In the context of continuous-time observations (Δ = 1 ms), every time interval has at most one spike from each spike train. For computational ease, we approximate the integral in equation 3.3 with a finite discrete sum of firing history as follows:

$$\lambda_k^c \equiv \lambda^c(k|\mathscr{H}_{1:k}) = \exp(\mu_c + \alpha_c S_k + \boldsymbol{\beta}_c^T \tilde{\mathbf{n}}_k), \tag{3.4}$$

where $\tilde{\mathbf{n}}_k$ is a vector containing the number of spike counts within the past intervals, and the length of the vector defines a finite number of windows of spiking history. By assuming that the spike trains are mutually independent, the observed data likelihood is given by (Brillinger, 1988; Brown et al., 2003)

$$p\left(dN_{1:K}^{1:C}|S_{1:K}, \boldsymbol{\theta}\right) = \prod_{c=1}^{C}\prod_{k=1}^{K} \exp\left(dN_k^c \log\left[\lambda_k^c \Delta\right] - \lambda_k^c \Delta\right). \tag{3.5}$$

The complete statistics of the continuous-time latent process may be characterized by a triplet: $\mathcal{S} = (n, \boldsymbol{\tau}, \boldsymbol{\chi})$, where $\boldsymbol{\tau} = (\tau_0, \tau_1, \ldots, \tau_n)$ is a vector that contains the duration lengths of sojourn times of $\mathcal{S}$, and $\boldsymbol{\chi} = (\chi_0, \chi_1, \ldots, \chi_n)$ represents the states visited in these sojourns. Let $\nu_0 = 0$, $\nu_j = \sum_{i=0}^{j-1} \tau_i \, (i = 1, 2, \ldots, n+1)$ and $\nu_{n+1} = T$. Alternatively, the complete data likelihood, equation 3.5, can be rewritten in another form,

$$p(dN_{1:K}^{1:C}|\mathcal{S}, \boldsymbol{\theta}) = \prod_{c=1}^{C}\prod_{l=1}^{n} \Pr(dN_l^{1:C}|\chi_l, \boldsymbol{\theta}), \tag{3.6}$$

where $dN_l^{1:C}$ denotes all of spike train observations during the sojourn time $[\nu_{l-1}, \nu_l]$ for the continuous-time (semi-)Markov process $\{S(t)\}$.

If we model each spike train as an inhomogeneous Poisson process with time-varying CIF $\lambda^c(t)$, the expected number of spike counts observed within the duration $[\nu_{l-1}, \nu_l]$ in the cth spike train is given by the integrated intensity (also referred to as cumulative hazard function):

$$\lambda_l^c = \int_{\nu_{l-1}}^{\nu_l} \lambda^c(t) dt. \tag{3.7}$$

Correspondingly, the observed data likelihood function, equation 3.5, is given by (Daley & Vere-Jones, 2002)

$$p\left(dN_{1:K}^{1:C}|\mathcal{S},\boldsymbol{\theta}\right)=\prod_{c=1}^{C}\prod_{l=1}^{n}\left(\exp\left(-\lambda_l^c\right)\prod_{i=1}^{y_l^c}\lambda^c(t_i)\right)$$
$$=\prod_{c=1}^{C}\prod_{l=1}^{n}\left(\exp\left(-\lambda_l^c\right)\frac{(\lambda_l^c)^{y_l^c}}{y_l^c!}\right),$$

where y_l^c denotes the spike counts of the cth spike train during the interval $(v_{l-1}, v_l]$, and t_i denotes the continuous-time index of the ith spike of a specific spike train during the interval $(v_{l-1}, v_l]$.

Ultimately, we can write the complete-data log likelihood in a compact form:[6]

$$\begin{aligned}\mathscr{L}=&\log p(S_{0:T}, y_{1:n}|\boldsymbol{\theta})\\ =&\sum_{l=1}^{n}\sum_{j=0}^{1}\left(\xi_l(j,j)\log[F(\boldsymbol{\theta}_j,\tau_l)]+\sum_{i\neq j}\xi_l(i,j)\log[1-F(\boldsymbol{\theta}_j,\tau_l)]\right)\\ &+\sum_{c=1}^{C}\sum_{l=1}^{n}\left(\int_{v_{l-1}}^{v_l}\log\lambda^c(t)dN^c(t)-\lambda_l^c\right),\end{aligned} \tag{3.8}$$

where $\boldsymbol{\theta}_j$ denotes the parameter(s) of the probability model of the sojourn time associated with state j.

3.2 Continuous-Time Markov Chain

In a continuous-time Markov chain (i.e., Markov process), state transitions from one state to another can occur at any instant of time. Due to the Markov property, the time that the system spends in any given state is memoryless, and the distribution of the survival time depends on the state (but not on the time already spent in the state); in other words, the sojourn time is exponentially distributed,[7] which can be characterized by a single rate parameter. The rate parameter, also known as the continuous-time state transition rate, defines the probability per time unit that the system makes a transition from one state to the other during an infinitesimal time interval:

$$q_{ij}=\lim_{\Delta t\to 0}\frac{\Pr(S_{t+\Delta t}=j|S_t=i)}{\Delta t},\quad i\neq j. \tag{3.9}$$

The total transition rate of state i satisfies the rate balance condition:

$$r_i\equiv q_{ii}=-\sum_{j\neq i}q_{ij}. \tag{3.10}$$

[6]In addition to the compact representation, another main reason for this reformulation is the efficiency and stability of numerical computation in calculating the observed data likelihood or likelihood ratio.

[7]The exponential distribution with mean 1/λ is the maximum entropy distribution among all continuous distributions with nonnegative support that have a mean 1/λ.

The holding time of the sojourn for state i follows an exponential distribution $\exp(-r_i\tau)$, or, equivalently, the transition times of state i are generated by a homogeneous Poisson process characterized by rate parameter r_i. For a two-state Markov chain, the transition rate matrix may be written as

$$Q=\begin{pmatrix} q_{00} & q_{01} \\ q_{10} & q_{11} \end{pmatrix}=\begin{pmatrix} r_0 & -r_0 \\ -r_1 & -r_1 \end{pmatrix}. \tag{3.11}$$

For an exponential random variable z, its cdf is computed as $\Pr[z \le t]=\int_{-\infty}^{t} re^{-rz}\,dz=1-e^{-rt}$, where re^{-rz} is the pdf of the exponential distribution with a rate parameter r. The reciprocal of the rate parameter, $1/r$, is sometimes called the survival parameter in the sense that the exponential random variable z that survives the duration of time has the mean $\mathbb{E}[z] = 1/r$. In light of equations 3.1 and 3.2, at a given specific time t, the probability of remaining within the current state sojourn is $\Pr[z>t]=1-\Pr[z \le t]=\int_{t}^{\infty} re^{-rz}\,dz=e^{-rt}$. Let r_0 and r_1 denote the transition rate for states 0 and 1, respectively. Let $\tau = t - v$ denote the elapsed time from the state transition up to the current time instant t; then the parameterized transition probability $\boldsymbol{P} = \{P_{ij}\}$ is characterized by

$$P_{ij}(\tau|r_i)=\begin{cases} \exp(-r_i\tau), & i=j \\ 1-\exp(-r_i\tau), & i \neq j \end{cases} \quad (i, j \in \{0, 1\}). \tag{3.12}$$

Now, the transition probability, instead of being constant, is a probabilistic function of the time after the Markov process makes a transition to or from a given state (the holding time from the last transition or the survival time to the next transition).

3.2.1 Imposing Physiological Constraints—Due to biophysical or physiological constraints, the sojourn time for a specific state might be subject to a certain range constraint, reflected in terms of the pdf. Without loss of generality, let $p(\tau)$ denote the standard pdf for a random variable τ, and let $\tilde{p}(\tau)$ denote the "censored" version of $p(\tau)$,[8] which is defined by

$$\tilde{p}(\tau)=\frac{1}{c}p(\tau)\mathbb{I}_{[a,b]}(\tau)=\begin{cases} \frac{1}{c}p(\tau), & \tau \in [a, b] \\ 0, & \text{otherwise} \end{cases},$$

where $\mathbb{I}(\cdot)$ is an indicator function and $a > 0$ and $b \in (a, \infty)$ are the lower and upper bounds of the constrained random variable τ (which is always positive for the duration of the sojourn time), respectively. The scalar c is a normalized constant determined by $c = F(b) - F(a)$, where $F(\cdot)$ denotes the corresponding cdf of the standard pdf $p(\tau)$ and $F(\infty) = 1$. Likewise, the censored version of the cdf is computed by $\tilde{F}(\tau)=\int_{-\infty}^{\tau}\tilde{p}(\tau)\,d\tau=\frac{1}{c}\int_{a}^{\tau}p(\tau)\,d\tau$.

Suppose the sojourn time τ of a continuous-time Markov chain follows a censored version of the exponential distribution; then we can write its censored pdf as

[8] *Censoring* is a term used in statistics that refers to the condition that the value of an observation is partially known or the condition that a value occurs outside the range of measuring tool.

$$\tilde{p}(\tau)=\begin{cases} \frac{r\exp(-r\tau)}{\exp(-ra)}, & \tau \geq a>0 \\ 0, & \text{otherwise} \end{cases}, \tag{3.13}$$

where the normalizing constant is given by $c = F(\infty) - F(a) = 1 - [1 - \exp(-ra)] = \exp(-ra)$.

3.3 Continuous-Time Semi-Markov Chain

In contrast to the Markov process, the semi-Markov process is a continuous-time stochastic process $\{S_t\}$ that draws the sojourn time $\{\nu_l\}$ spent in specific discrete states from a nonexponential distribution. In other words, the characterization of the sojourn time is no longer an exponential pdf. Table 2 lists a few examples of continuous-time probability models for characterizing the sojourn time duration for the interevent interval (Tuckwell, 1989). In general, the nonexponential censored pdf with a lower bound gives the flexibility to model the "refractory period" of the UP or DOWN state.

3.3.1 Two-Parameter Exponential Family of Distributions for the UP and DOWN State—To characterize the nonexponential survival time behavior of semi-Markov processes, here we restrict our attention to three probability distributions that belong to the two-parameter exponential family of continuous probability distributions. We define the censored versions of three pdfs as follows:

- The censored gamma distribution $\tilde{p}(\tau; s, \kappa)$:

$$\tilde{p}(\tau;s,\kappa)=\frac{1}{c}\mathbb{I}_{[a,b]}(\tau)\,p(\tau;s,\kappa)=\frac{1}{c}\mathbb{I}_{[a,b]}(\tau)\tau^{s-1}\frac{\exp(-\tau/\kappa)}{\Gamma(s)\kappa^{s}},$$

  where κ and s represent the scale and shape parameters, respectively. If s is an integer, then the gamma distribution represents the sum of s exponentially distributed random variables, each with a mean κ. Similarly, c is a normalized constant for the censored pdf $\tilde{p}(\tau; s, \kappa)$: $c = F(b) - F(a)$, and $F(\cdot)$ is the cdf of the standard gamma distribution.

- The censored log-normal distribution $\tilde{p}(\tau; \mu, \sigma)$:

$$\begin{aligned}\tilde{p}(\tau;\mu,\sigma)&=\tfrac{1}{c}\mathbb{I}_{[a,b]}(\tau)\,p(\tau;\mu,\sigma)\\ &=\tfrac{1}{c}\mathbb{I}_{[a,b]}(\tau)\frac{1}{x\sigma\sqrt{2\pi}}\exp\left(-\frac{[\ln(\tau)-\mu]^2}{2\sigma^2}\right),\end{aligned}$$

  where the mean, median, and variance of the log-normal distribution are $\exp(\mu + \sigma^2/2)$, $\exp(\mu)$, and $\exp(2\mu + \sigma^2)[\exp(\sigma^2) - 1]$, respectively.

- The censored inverse gaussian distribution $\tilde{p}(\tau; \mu, s)$:

$$\begin{aligned}\tilde{p}(\tau;\mu,s)&=\tfrac{1}{c}\mathbb{I}_{[a,b]}(\tau)\,p(\tau;\mu,s)\\ &=\tfrac{1}{c}\mathbb{I}_{[a,b]}(\tau)\sqrt{\frac{s}{2\pi\tau^3}}\exp\left(-\frac{s(\tau-\mu)^2}{2\mu^2\tau}\right),\end{aligned}$$

  where μ and s represent the mean and shape parameters, respectively.

The choice of the last two probability distributions is mainly motivated by the empirical data analysis published earlier (Ji & Wilson, 2007). Typically, for a specific data set, a smoothed histogram analysis is conducted to visualize the shape of the distribution, and the Kolmogorov-

Smirnov (KS) test can be used to empirically evaluate the fit of specific probability distributions. Mostly likely, none of parametric probability distribution candidate would fit perfectly (i.e., within 95% confidence interval) for the real-world data; we often choose the one that has the best fit.[9] In the simulation data shown in Figure 1, we have used the following constraints for the UP and DOWN states: [0.1, 3] (unit: second) for UP state and [0.05, 1] (unit: second) for DOWN state. The lower and upper bounds of these constraints are chosen in light of the results reported from Ji and Wilson (2007). Note that the shapes of the log-normal and inverse gaussian pdfs and cdfs are very similar, except that inverse gaussian distribution is slightly sharper when the value of the random variable is small (Takagi, Kumagai, Matsunaga, & Kusaka, 1997). In addition, the tail behavior of these two distributions differs; however, provided we consider only their censored versions (with finite duration range), the tail behavior is not a major concern here.

3.4 EM Algorithm

In the continuous-time model, we treat the individual spike trains separately and aim to estimate their own parameters. Let $\boldsymbol{\theta}=(\boldsymbol{\theta}_{up}, \boldsymbol{\theta}_{down}, \{\mu_c\}_{c=1}^{C}, \{\alpha_c\}_{c=1}^{C}, \{\boldsymbol{\beta}_c\}_{c=1}^{C},)$ denote the unknown parameters of interest, where θ_{up} and θ_{down} represent the parameters associated with the parametric pdfs of the UP and DOWN states, respectively; the rest of the parameters are associated with the CIF model for respective spike trains. For an unknown continuous-time latent process $\{S(t); 0 \le t \le T\}$ (where $S(t) \in \{0, 1\}$), let n be the number of jumps between two distinct discrete states. Let $\mathcal{S} = (n, \boldsymbol{\tau}, \boldsymbol{\chi})$ be a triplet of the Markov or semi-Markov process, where $\boldsymbol{\tau} = (\tau_0, \tau_1, \ldots, \tau_n)$ is a vector that contains the duration of the sojourn time of $\mathcal{S}$ and $\boldsymbol{\chi} = (\chi_0, \chi_1, \ldots, \chi_n)$ represents the states visited in these sojourns.

Let $\mathcal{Y}$ denote all the spiking timing information of the observed spike trains. Similar to the discrete-time HMM, we aim to maximize the Q-function, defined as follows:

$$Q(\boldsymbol{\theta})=\sum_{S} p\,(\mathcal{S}|\mathcal{Y},\boldsymbol{\theta}) \log p(\mathcal{Y},\mathcal{S}|\boldsymbol{\theta}). \tag{3.14}$$

The inference can be tackled in a similar fashion by the EM algorithm.

First, let us consider a simpler task where the transition time of the latent process is known and the number of state jumps is given. In other words, the parameters n and $\boldsymbol{\tau}$ are both available, so the estimation goal becomes less demanding. We need to estimate only $\boldsymbol{\chi}$ and $\boldsymbol{\theta}$.

Since the number of state transition, n, is known, $p(\mathcal{Y}, \mathcal{S} \mid \boldsymbol{\theta})$ is simplified to

$$p\,(\mathcal{S},\mathcal{Y}|\boldsymbol{\theta}) = \sum_{l=1}^{n} P_{\mathcal{S}_{l-1},\mathcal{S}_l}\,(\tau_l)\, P(\mathcal{Y}_l|\mathcal{S}(\chi_l)). \tag{3.15}$$

Let $\xi_l(i, j) = \Pr(\mathcal{S}(\chi_{l-1}) = i, \mathcal{S}(\chi_l) = j)$ and $\gamma_l(i) = \Pr(\mathcal{S}(\chi_l) = i)$. In the case of the continuous-time Markov chain, the complete data log likelihood is given by

[9]Alternatively, one can use a discrete nonparametric probability model for the sojourn time, which is discussed in section 5.

$$\begin{aligned}\mathcal{L}(\mathcal{S},\boldsymbol{\theta}) &= \log p(S_{0:T},\mathcal{Y}|\boldsymbol{\theta})\\ &= \sum_{l=1}^{n}\sum_{j=0}^{1}\left(\xi_l(j,j)(-r_j\tau_l)+\sum_{i\neq j}\xi_l(j,i)\log[1-\exp(-r_j\tau_l)]\right)\\ &\quad+\sum_{c=1}^{C}\sum_{l=1}^{n}\left(\int_{\nu_{l-1}}^{\nu_l}\log\lambda^c(t)dN^c(t)-\lambda_l^c\right),\end{aligned} \tag{3.16}$$

where $\boldsymbol{\theta} = (r_0, r_1, \mu_1\alpha_1, \beta_1, \ldots, \mu_C, \alpha_C, \beta_C)$ denotes the augmented parameter vector that contains all of the unknown parameters. In the case of the continuous-time semi-Markov chain where the sojourn time is modeled by a nonexponential probability distribution, we can write the log-likelihood function as

$$\begin{aligned}\mathcal{L} &= \log p(S_{0:T},\mathcal{Y}|\boldsymbol{\theta})\\ &= \sum_{l=1}^{n}\left(\sum_{j=0}^{1}\xi_l(j,j)\log[1-F(\tau_l;\boldsymbol{\theta}_j)]+\sum_{i\neq j}\xi_l(j,i)\log[F(\tau_l;\boldsymbol{\theta}_j)]\right)\\ &\quad+\sum_{c=1}^{C}\sum_{l=1}^{n}\left(\int_{\nu_{l-1}}^{\nu_l}\log\lambda^c(t)\,dN^c(t)-\lambda_l^c\right),\end{aligned} \tag{3.17}$$

where $F(\cdot)$ denotes the cdf of the nonexponential probability distribution.

Conditioned on the parameter $\boldsymbol{\theta}$, the posterior probabilities ξ_l and γ_l (for $l = 1, \ldots, n$) can be similarly estimated by the forward-backward algorithm as in the E-step for the HMM, whereas in the M-step, the new parameter $\boldsymbol{\theta}^{new}$ is obtained by

$$\boldsymbol{\theta}^{new} = \arg\max_{\boldsymbol{\theta}}\sum_{\mathcal{S}}\Pr\{\mathcal{S}|\mathcal{Y},\boldsymbol{\theta}^{old}\}\log(\Pr\{\mathcal{S},\mathcal{Y}|\boldsymbol{\theta}\}). \tag{3.18}$$

More specifically, for the parameters associated with the transition probability density model, we might, for example, assume that the UP and DOWN state durations are both log normal distributed with parameters $\boldsymbol{\theta}_j = \{\mu_j, \sigma_j\} (j = 0, 1)$, and they can be estimated by

$$\begin{aligned}\{\mu_j,\sigma_j\} = \arg\max_{\mu,\sigma} &\left\{\sum_{l=1}^{n}\xi_l(j,j)\log[1-F(\tau_l;\mu,\sigma)]\right.\\ &\left.+\sum_{l=1,i\neq j}^{n}\xi_l(j,i)\log[F(\tau_l;\mu,\sigma)]\right\}.\end{aligned} \tag{3.19}$$

In light of Table 2, setting the derivatives of μ_j and σ_j to zeros yields

$$\begin{aligned}0 &= \sum_{l=1}^{n}\frac{-1}{\sigma}\exp\left(-\frac{(\ln\tau_l-\mu)^2}{2\sigma^2}\right)\left(\frac{\xi_l(j,j)}{1-F(\tau_l;\mu,\sigma)}+\frac{\sum_{i\neq j}\xi_l(j,i)}{F(\tau_l;\mu,\sigma)}\right),\\ 0 &= \sum_{l=1}^{n}\frac{\mu-\ln\tau_l}{\sigma^2}\exp\left(-\frac{(\ln\tau_l-\mu)^2}{2\sigma^2}\right)\left(\frac{\xi_l(j,j)}{1-F(\tau_l;\mu,\sigma)}+\frac{\sum_{i\neq j}\xi_l(j,i)}{F(\tau_l;\mu,\sigma)}\right),\end{aligned}$$

where we have used $\frac{d\,\mathrm{erf}(z)}{dz} = \frac{2}{\sqrt{\pi}}\exp(-z^2)$ in light of Table 2. The above two equations can be solved iteratively with the Newton-Ralphson algorithm.

Once the state estimate $\hat{S}(t)$ is available from the E-step,[10] the parameters associated with the likelihood model can also be estimated by the Newton-Ralphson or the IWLS algorithm,

$$\{\mu_c, \alpha_c, \beta_c\} = \arg\max_{\mu,\alpha,\beta} \left(\sum_{l=1}^{n} \int_{\nu_{l-1}}^{\nu_l} \log \lambda^c(t)\ dN^c(t) - \lambda_l^c \right), \tag{3.20}$$

where $\lambda^c(t)$ and λ_l^c are defined by equations 3.3 (or 3.4) and 3.7, respectively.

Notably, the EM algorithm described above has a few obvious drawbacks. Basically, it assumes that the information as to when and how many state transitions occur during the latent process is given; once the number of state jumps (say, *n*) is determined, it does not allow the number *n* to change, so it is incapable of online model selection. Furthermore, it is very likely that the EM algorithm suffers from the local maximum problem, especially if the initial conditions of the parameters are far from the true estimates. In the following, we present an alternative inference method to overcome these two drawbacks, and the method can be regarded as a generalization of the EM algorithm, except that the E-step state estimation is replaced by a Monte Carlo sampling procedure. This method is often called the Monte Carlo EM (MCEM) algorithm (Chan & Ledolter, 1995; McLachlan & Krishnan, 1996; Tanner, 1996; Stjernqvist, Rydén, Sköld, & Staaf, 2007). The essence of MCEM is the theory of Markov chain Monte Carlo (MCMC), which will be detailed below.

3.5 Monte Carlo EM Algorithm

The basic idea of MCMC sampler is to draw a large number of samples randomly from the posterior distribution and then obtain a sample-based numerical approximation of the posterior distribution. Unlike other Monte Carlo samplers (such as importance sampling and rejection sampling), the MCMC method is well suited for sampling from a complex and high-dimensional probability distribution. Instead of drawing independent samples from the posterior distribution directly, MCMC constructs a Markov chain such that its equilibrium will eventually approach the posterior distribution. The Markov chain theory states that given an arbitrary initial value, the chain will ultimately converge to the equilibrium point provided the chain is run sufficiently long. In practice, determining the convergence as well as the “burn-in time” for MCMC samplers requires diagnostic tools (see Gilks, Richardson, & Spiegelhalter, 1995, for general discussions of the MCMC methods). Depending on the specific problem, the MCMC method is typically computationally intensive, and the convergence process can be very slow. Nevertheless, here we focus on methodology development, and therefore the computational demand is not the main concern. Specifically, constructing problem-specific MCMC proposal distributions (densities) is the key to obtain an efficient MCMC sampler that has a fast convergence speed and a good mixing property (Brooks, Guidici, & Roberts, 2003). For the variable-dimension RJMCMC method, we present some detailed mathematical treatments in appendix A.

The Monte Carlo EM (MCEM) algorithm works just like an ordinary EM algorithm, except that in the E-step, the expectation operations (i.e., computation of sufficient statistics) are replaced by Monte Carlo simulations of the missing data. Specifically, *M* realizations of the latent process $S(t)$ $(0 \le t \le T)$ are simulated, and in this case the Q-function can be written as

[10]Note that $\hat{S}(t)$ is not the same as χ_l $(l = 1, 2, \ldots, n)$ in that the former is stochastic and the latter is deterministic.

$$Q(\mathcal{S},\boldsymbol{\theta}) \approx \widehat{Q}(\mathcal{S},\boldsymbol{\theta}) = \frac{1}{M}\sum_{m=1}^{M} Q\left(\mathcal{S}^{(m)},\boldsymbol{\theta}\right), \tag{3.21}$$

where $\mathcal{S}^{(m)}$ denotes the mth simulated latent process for the unknown state (missing data).

The M-step of MCEM is the same as the conventional M-step in the EM algorithm. Specifically, the parameters of the CIF appearing in the likelihood model are estimated using equation 3.20. However, the estimation of the parameters for the UP or DOWN state sojourn depends on the type of probability distribution. Here we distinguish three different possible scenarios.

First, when the latent process is a continuous-time Markov chain and the sojourn time durations for the UP and DOWN states are both exponentially distributed, then $\boldsymbol{\theta}_{up}$ and $\boldsymbol{\theta}_{down}$ correspond to the rate parameters r_1 and r_0, respectively. The Q-function for a single Monte Carlo realization of $\mathcal{S}$ can be written as

$$\begin{aligned} Q(\mathcal{S},\boldsymbol{\theta}) = & \sum_{i=0, j\neq i}^{1} (\widehat{n}_{ij} \log q_{ij} + r_i T_i) \\ & + \sum_{c=1}^{C}\sum_{l=1}^{n} \left(\int_{\nu_{l-1}}^{\nu_l} \log \lambda^c(t)\, dN^c(t) - \lambda_l^c \right), \end{aligned} \tag{3.22}$$

where n_{ij} denotes the number of jumps from state i to state j during $[0, T]$, and $T_i = \int_0^T \mathbb{I}(S(t) = i)\, dt$ denote the total time or the sojourn length of $\{S(t)\}$ spent in state i during $[0, T]$. Let $n_i = \sum_{l=0}^{n} \mathbb{I}(\chi_l = i)$ denote the number of events that occur while $\{S(t)\}$ is in state i; then it is known that the transition rate matrix can be estimated by $\hat{q}_{ij} = n_{ij}/T_i$ and $r_i = q_{ii} = n_i/T_i$, where n_{ij}, n_i, and T_i are the sufficient statistics (Rydén, 1996). In this case, r_i corresponds to the MLE. With M Monte Carlo realizations, the rate parameter will be estimated by

$$r = \frac{1}{M}\sum_{m=1}^{M} \frac{\sum_{l=0}^{n} \mathbb{I}\left(\chi_l^{(m)} = 0\right)}{\int_0^T \mathbb{I}(S^{(m)}(t) = 0)\, dt}.$$

In the second scenario, when the latent process is a continuous-time semi-Markov chain where the sojourn time durations for both UP and DOWN states are nonexponentially distributed, the Q-function can be written as

$$\begin{aligned} Q(\mathcal{S},\boldsymbol{\theta}) = & \frac{1}{M}\sum_{m=1}^{M}\sum_{l=0}^{n-1} \left(\sum_{j=0}^{1} \sum_{j\to i}^{i\neq j} \log \left[1 - F\left(\tau_l^{(m)}; \boldsymbol{\theta}_j\right)\right]\right) \\ & + \sum_{c=1}^{C}\sum_{l=1}^{n} \left(\int_{\nu_{l-1}}^{\nu_l} \log \lambda^c \left(\overline{S}(t), \boldsymbol{\theta}\right) dN^c(t) - \lambda_l^c \right), \end{aligned} \tag{3.23}$$

where $\overline{S}(t) = \frac{1}{M}\sum_{m=1}^{M} \widehat{S}^{(m)}(t)$ is obtained from the Monte Carlo mean statistic of M simulated latent processes. Similarly, the parameters of the nonexponential probability distributions can be estimated by their MLE based on their Monte Carlo realizations. For instance, in the case of inverse gaussian distribution, the mean parameter is given by $\mu_j = \frac{1}{n_j M} \sum_{m=1}^{M} \sum_{l=0}^{n} \tau_l^{(m)} \mathbb{I}(\chi_l^{(m)} = j)$, and the shape parameter is given by $\sigma_j = [\frac{1}{n_j M} \sum_{m=1}^{M} \sum_{l=0}^{n} ((\tau_l^{(m)})^{-1} - \mu_j^{-1}) \mathbb{I}(\chi_l^{(m)} = j)]^{-1}$. In the case of log-normal distribution, the mean parameter is given by $\mu_j = \frac{1}{n_j M} \sum_{m=1}^{M} \sum_{l=0}^{n} \mathbb{I}(\chi_l^{(m)} = j) \ln \tau_l^{(m)}$, and the SD parameter is given by $\sigma_j = \frac{1}{n_j M} \sum_{m=1}^{M} \sum_{l=0}^{n} \mathbb{I}(\chi_l^{(m)} = j) |\ln \tau_l^{(m)} - \mu_j|$.

If, in the third scenario, one of the state sojourn time durations is exponentially distributed and the other is nonexponential, then the resulting latent process is still a semi-Markov chain, and the estimation procedure remains similar to that in the above two cases.

3.5.1 Initialization of the MCMC Sampler—It is important to choose sensible initial values for the simulated (semi-) Markov chain since a poor choice of the initial $\mathcal{S}^{(0)}$ can lead to a sampler that takes a very long time to converge or result in a poor mixing of the (semi-) Markov chain. In our experiment, we typically use a discrete-time HMM model (with a 10 ms bin size) to estimate the hidden state sequence and then interpolate it to obtain an initial estimate of the continuous-time state process (with 1 ms bin size), from which we obtain the initial values of $\{n, \boldsymbol{\tau}, \boldsymbol{\chi}\}$.

3.5.2 Algorithmic Procedure—In summary, the MCEM algorithm is run as follows:

- Initialize the MCMC sampler state for $\mathcal{S} = \{n, \boldsymbol{\tau}, \boldsymbol{\nu}\}$.
- Iterate the MCEM algorithm's E and M steps until the log likelihood reaches a local maximum or saddle point.
  1. Monte Carlo E-step: Given an initial state $\mathcal{S}^{(0)}$, run the RJMCMC sampling procedure to draw M Monte Carlo samples $\{\mathcal{S}^{(m)}\}_{m=1}^{M}$, based on which to compute the necessary Monte Carlo sufficient statistics.
  2. M-step: estimate the parameters $\{\boldsymbol{\theta}_{up}, \boldsymbol{\theta}_{down}\}$ with their MLE.
  3. M-step: optimize the parameters $\{\mu_c, \alpha_c, \boldsymbol{\beta}_c\}_{c=1}^{C}$ according to equation 3.20.
- Upon convergence, reconstruct the hidden state $\hat{S}(t)$ in the continuous-time domain.
- Compute $\lambda^c(t)$ for each spike train c, and conduct goodness-of-fit tests (see below) for all spike trains being modeled.

3.5.3 Reconstruction of Hidden State—There are two ways to reconstruct the hidden state of the latent process. The first is to apply the Viterbi algorithm once the MCEM inference is completed (when n and $\boldsymbol{\tau}$ have been determined). In the second, and simpler, approach, we can determine the state by the following rule (Ball et al., 1999). For $m = 1, 2, \ldots, M$, let

$$\widehat{S}^{(m)}(t) = \begin{cases} 1, & \text{if } \chi_t^{(m)} \in \text{UP}, \\ 0, & \text{if } \chi_t^{(m)} \in \text{DOWN}, \end{cases} \tag{3.24}$$

and let $\overline{S}(t) = \frac{1}{M} \sum_{m=1}^{M} \widehat{S}^{(m)}(t)$, and the point estimate of the hidden state is

$$\widehat{S}(t) = \begin{cases} 1 & \text{if } \overline{S}(t) \geq 0.5, \\ 0 & \text{if } \overline{S}(t) < 0.5. \end{cases} \tag{3.25}$$

Furthermore, the marginal prior probability of the hidden state is given by

$$\widehat{\pi}_i = \frac{1}{M} \sum_{m=1}^{M} \mathbb{I}\left(S^{(m)}(0) = i\right). \tag{3.26}$$

3.5.4 Goodness-of-Fit Tests—Upon estimating the CIF model $\lambda^c(t)$ for each spike train (see equation 3.3), the goodness of fit of the estimated model is tested in light of the time-rescaling theorem (Brown, Barbieri, Ventura, Kass, & Frank, 2002). Specifically, given a point process specified by J discrete events: $0 < u_1 < \cdots < u_J < T$, define the random variables $z_j = \int_{u_{j-1}}^{u_j} \lambda(\tau)\, d\tau$ for $j = 1, 2, \ldots, J - 1$. Then the random variables z_js are independent, unit-mean exponentially distributed. By introducing the variable of transformation $v_j = 1 - \exp(-z_j)$, v_js are independent and uniformly distributed within the region [0, 1]. Let $g_j = \Phi^{-1}(v_j)$ (where $\Phi(\cdot)$ denotes the cdf of the standard gaussian distribution); then g_js will be independent standard gaussian random variables. Furthermore, the standard Kolmogorov-Smirnov (KS) test is used to compare the cdf of v_j against that of the random variables uniformly distributed within [0, 1]. The KS statistic is the maximum deviation of the empirical cdf from the uniform cdf. To compute it, the v_js are sorted from the smallest to the largest value; then we plot values of the cdf of the uniform density defined as $\frac{j - 0.5}{J}$ against the ordered v_j s. The points should lie on the 45 degree line. In a Cartesian plot of the empirical cdf as the y-coordinate versus the uniform cdf as the x-coordinate, the 95% confidence interval lines are $y = x \pm \frac{1.36}{(J-1)^{1/2}}$. The KS distance, defined as the maximum distance between the KS plot and the 45 degree line, is used to measure the lack of fit between the model and the data.

Furthermore, we measure the independence of the time-rescaled time series by computing the autocorrelation function of g_js: $\mathrm{ACF}(m) = \frac{1}{J - m} \sum_{j=1}^{J-m} g_j g_{j+m}$. Since g_js are normally distributed, if they are independent, then they are also uncorrelated; hence, ACF(m) shall be small for all values of m, and the associated 95% confidence interval is $0 \pm \frac{1.96}{(J-1)^{1/2}}$.

3.5.5 Implementation and Convergence—Let the triple $\mathcal{S} = (n, \boldsymbol{\tau}, \boldsymbol{\chi})$ denote all the information of the continuous-time latent process, which contains n state jumps and $n + 1$ durations of corresponding sojourn times, and the discrete states that are visited in the sojourns.

To simulate the Markov chain, we first obtain the initial conditions for both the state and parameters $\{\mathcal{S}^{(0)}, \boldsymbol{\theta}^{(0)}\}$. Next, we run the MCMC sampler (see appendix A for details) to generate a sequence of Monte Carlo samplers $\{\mathcal{S}^{(k)}\}$ for $k = 1, 2, \ldots, M$, and the realizations $\{\mathcal{S}^{(k)}\}$ can be viewed as the samples drawn from the conditional posterior $p(\mathcal{S} \mid \mathcal{Y}, \boldsymbol{\theta})$. At each MCEM step, the parameter vector $\boldsymbol{\theta}$ is updated in the Monte Carlo M-step using the sufficient statistics obtained from $p(\mathcal{S} \mid \mathcal{Y}, \boldsymbol{\theta})$. Typically, to reduce the correlation of the simulated samples, a "burn-in" period is discarded at the beginning of the simulated (semi-) Markov chain. Even after the burn-in period, the successive realizations of $\{\mathcal{S}^{(k)}\}$ might still be highly correlated. This problem can be alleviated by using the “thinning” or subsampling technique: every N_p simulated samples in the chain is used. Although the thinning technique can increase the Monte Carlo variance of the estimate (Geyer, 1992), it is widely used in practice to reduce the correlation among the samples. In our experiment, we typically chose $N_p = 10$.

At the end of each Monte Carlo sampling step, the sufficient statistics for computing the acceptance probability and updating the parameter $\boldsymbol{\theta}$ (in the M-step) need to be stored, and all new information $(n, \boldsymbol{\tau}, \boldsymbol{\chi})$ needs to be updated each time $\mathcal{S}$ changes.

In terms of convergence, Markov chain theory tells us that when $M \to \infty$, the samples are asymptotically drawn from the desired target (equilibrium) distribution. However, choosing the proper value of M is often problem dependent, and the convergence diagnosis of the MCMC sampler remains an active research topic in the literature (e.g., Gelman & Rubin, 1992; Cowles & Carlin, 1996).

4 Experimental Results

4.1 Synthetic Data

First, we simulate four spike trains with the time-rescaling theorem (see Figure 2 for a snapshot of one realization). The latent state variable is assumed to be drawn from a two-state discrete space: $S(t) \in \{0, 1\}$. The simulation is conducted with a 1 ms time bin size with the following model:

$$\lambda^c(t)=\exp(\mu_c+\alpha_c S(t)+\beta_c \tilde{n}(t)), \quad (c=1, \dots, 4),$$

where $\tilde{n}(t)$ denotes the number of spike counts across all spike trains during the previous 100 ms prior to the current time index t, and the parameters of individual spike trains are set as follows:

$$\begin{array}{llll} \mu_1=-3.5, & \mu_2=-4.0, & \mu_3=-3.8, & \mu_4=-3.8, \\ \alpha_1=7.0, & \alpha_2=8.0, & \alpha_3=7.6, & \alpha_4=7.6, \\ \beta_1=0.06, & \beta_2=0.05, & \beta_3=0.03, & \beta_4=0.05. \end{array}$$

For the simulated hidden latent process, the total duration is $T = 30$ s, and the number of jumps varies from 35 to 45, yielding an average occurrence rate of state transitions of about 80 min^{-1}. Furthermore, we assume that the sojourn time durations for both UP and DOWN states follow a log-normal distribution. For the UP state, the survival time length is randomly drawn from logn(−0.4005, 0.8481) (such that the mean and median value of the duration are 0.67 s and 0.96 s, respectively), with a lower bound of 0.15 s; and for the DOWN state, the survival time length is randomly drawn from logn(−1.9661, 0.6231) (such that the mean and median value of the duration are 0.14 s and 0.17 s, respectively), with a lower bound of 0.05 s.

To test the discrete-time HMM model, the spike trains are binned in 10 ms and collected by spike counts. We employed the EM algorithm with the following initialization parameters: $\pi_0 = \pi_1 = 0.5, P_{00} = P_{11} = 0.9, P_{01} = P_{10} = 0.1$. For the synthetic data, the EM algorithm typically converges within 200 iterations. The forward-backward algorithm computes all necessary sufficient statistics. Upon convergence, the Viterbi algorithm produced the ultimate state sequence output, yielding an average decoding error rate of 1.52% (averaged over 10 independent runs). In this case, since the CIF model is given (no model mismatch issue is involved), the decoding error rate is reasonably low even with the discrete-time HMM. As a comparison, we also employed the threshold-based method (Ji & Wilson, 2007; see appendix B for brief descriptions) to classify the UP and DOWN states using the simulated spike trains. It was found (see Table 3) that the discrete-time HMM method yields better performance than the threshold-based method. Figure 3 plots a snapshot of hidden state estimation obtained from the discrete-time HMM in our experiments.

Next, we applied the MCEM algorithm to refine the latent state estimation in the continuous-time domain. Naturally, with a smaller bin size, the continuous-time model allows precisely segmenting the UP and DOWN states for identifying the location of state transition. With the initial conditions obtained from the discrete-time EM algorithm, we simulated the Markov chains for 20,000 iterations and discarded the first 1000 iterations (“burn-in” period). For the synthetic data, the MCEM algorithm converged after 20 to 30 iterations, and we were able to further improve the decoding accuracy by reducing the average error rate to 1.26%. As seen in Table 3, the continuous-time model outperformed the discrete-time HMM model in terms of the lower estimation error. However, in terms of estimating the correct number of state transitions, the HMM obtained nearly 100% accuracy in all 10 Monte Carlo trials (except for two trials that miscount two more jumps); in this sense, the HMM estimation result can be treated as a very good initial state as the input to the continuous-time semi-Markov chain model. In addition, the continuous-time model yields a 10 times greater information transmission rate (1 bit/ms) than the discrete-time model (1 bit/10 ms). We also computed the probability distribution statistics of the UP and DOWN states from both estimation methods. In the discrete-time HMM, we used the sample statistics of the UP and DOWN state durations as the estimated results. These results were also used as the initial values for the continuous-time semi-Markov process, where the MCEM algorithm was run to obtain the final estimate. The results are summarized in Table 4.

Once the estimates of $\{S(t)\}$ and $\{\mu_c, \alpha_c, \beta_c\}$ become available, we compute $\lambda^c(t)$ for the simulated spike trains in continuous time (with $\Delta = 1$ ms). Furthermore, the goodness-of-fit tests are employed to the rescaled time series, and the KS plots and the autocorrelation plots for the simulated spike trains are shown in Figure 4. As seen from the figure, these plots fall almost within the 95% confidence bounds, indicating the model fit is sufficiently satisfactory.

Finally we also do extra simulation studies by examining the sensitivity of different methods regarding the change of two factors: the modulation gains of the hidden state and the number of observed spike trains. The first issue examines the impact of the global network activity during the UP state, that is, the α_c component appearing in $\lambda^c(t)$. Specifically, we modify the gain parameters of individual spike trains (while keeping remaining parameters unchanged) as follows:

$$\alpha_1=6.7, \quad \alpha_2=7.8, \quad \alpha_3=7.3, \quad \alpha_4=7.3,$$

such that each λ_c is reduced about 20% during the UP state period. It appears that the average performance of the threshold-based method degraded from the original 2.91% to 3.62% (with a trial-and-error selected threshold), while the performance of the probabilistic models remained almost unchanged. This is partly because of the fact that a correct generative model is used and the uncertainties of the hidden state were taken into account during the final estimation (see Figure 5). In the meantime, if α_c is decreased more and more, the mean MUA firing rate will be significantly decreased, the rate difference between UP and DOWN periods is reduced, and therefore the ambiguity between them increases. At some point, it can be imagined that all methods will break down unless the bin size is increased accordingly (at the cost of loss of accuracy in the classification boundary). Due to space limitations, we do not explore this issue further here.

The second issue examines the estimation accuracy of the missing variable against the number of observed variables. It is expected that as more and more observations are added, the resulting discrepancy between the threshold-based method and the probabilistic models will also become smaller, since the uncertainty of the hidden state is less (or the posterior of the hidden variable is larger with more spike train observations). In our simulations, we did extra experiments by either reducing or increasing the number of simulated spike trains, followed by rechecking the results across different setups for different methods. The estimation error results are summarized in Figure 6. Specifically, for the threshold-based method, as more and more spike trains are added, its estimation error gradually improves. This is expected since the threshold selection criterion (see appendix B) heavily depends on the number of the spike train observations, and adding more spike train observations help to disambiguate the boundary between the UP and DOWN states. Meanwhile, for the probabilistic models, the estimation performance either slightly improves (in the discrete-time HMM) or remains roughly the same (in the continuous-time model). This is partly because adding more observations will also increase the number of parameters to be estimated in the continuous-time model, so the difficulty of inference also increases; whereas the HMM performance is likely to saturate quickly due to either the insufficiency of the model or the local minimum problem inherent in the EM algorithm. This observation implies that the probabilistic models are particularly valuable when the number of spike train observations is relatively small and that the simple threshold-based method becomes more and more reliable in terms of estimation accuracy—yet its performance is still worse than that of two probabilistic models. This is probably because it is difficult to find an optimal kernel smoothing parameter or the two threshold values (see appendix B). Moreover, the threshold-based method cannot produce the statistics of interest (e.g., posterior probability, transition probability).

4.2 Real-World Spike Train Data

Next, we apply our models to validate some real-world simultaneously recorded spike trains collected from one behaving rat (Vijayan, 2007), where the MUA, cortical and hippocampal EEGs, and EMG have been simultaneously recorded (see Figure 7 for a snapshot). We presented the spike train data of a single animal (in one day), recorded from the primary somatosensory cortex (SI) during SWS. Neurophysiological studies of neural spike trains and EEGs across different rats and different recording days, as well as the comparison between the cortex and hippocampus, will be presented elsewhere. In this study, 20 clearly identified cortical cells from eight tetrodes were recorded and sorted. All spiking activity from these eight tetrodes was used to determine the UP and DOWN states.

For this study, we selected about 15.7 minutes of recordings from a total of 11 (interrupted) SWS periods of one rat,[11] where the mean ± SD length of SWS periods is 85.7 ± 35.8 s (maximum 156.1 s, minimum 30.7 s). We pulled out the multiunit spikes from eight tetrodes (no spike sorting is necessary here). For each spike train (from one tetrode), we empirically chose the following CIF model, as defined in the continuous-time domain:[12]

[11]The sleep stage classification was based on the recordings of electromyography (EMG) and hippocampal and cortical EEGs (ripple power, theta and delta power). SWS is characterized as having low EMG, high hippocampal ripple, low hippocampal theta (4–8 Hz), and high cortical delta (2–4 Hz). In practice, we varied the cut-off thresholds of those parameters (via grid search) to obtain a suboptimal SWS classification for a specific rat.

[12]The model was empirically verified by model selection based on the GLM fit of a small data set using the glmfit function in Matlab. The model with the lowest deviance (defined by twofold negative log likelihood) or the smallest Akaike's information criterion (AIC) value will be chosen.

$$\begin{aligned}\lambda^{c}(t)&=\exp\left(\mu_c+\alpha_c S_t+\gamma_c\int_0^t e^{-\beta_c\tau}dN(t-\tau)d\tau\right)\\&\approx\exp\left(\mu_c+\alpha_c S_t+\gamma_c\int_0^{\bar{\tau}} e^{-\beta_c\tau}dN(t-\tau)d\tau\right)\\&\approx\exp\left(\mu_c+\alpha_c S_t+\gamma_c\boldsymbol{\beta}_c^T\mathbf{N}_{t-\bar{\tau}:t}\right),\end{aligned}$$

where the exponential decaying parameter β_c is initially set to a value that lets $e^{-\beta_c\tau}\approx 0$ for $\tau > \bar{\tau} = 100$ ms, which leads to the second line of approximation; the third line of approximation appears when we replace the continuous convolution with a discrete vector product, in which $\boldsymbol{\beta}$ denotes the vector containing 100 coefficients sampled from $e^{-\beta_c\tau}$ with a 1 ms temporal resolution, and $\boldsymbol{N}_{t-\bar{\tau}:t}$ denotes the vector containing 100 0 or 1 elements that indicate, respectively the absence or presence of spikes. For the initial values, we set $\gamma_c = 0$ 01 for all spike trains; μ_c and α_c were hand-tuned based on a small data set.[13]

Since $\boldsymbol{\theta}$ will be largely dependent on $\mathcal{S}$ in the MCEM algorithm, a sensible choice of initial state $\mathcal{S}^{(0)}$ is important for the convergence of the MCMC sampler. We initialized the state with the results obtained from the discrete-time HMM (10 ms bin size) and interpolated the intermediate missing values in the continuous-time domain (1 ms bin size). The rate parameter defined for the HMM (see equation 2.5) was assumed as follows:[14]

$$\lambda_k=\exp(\mu+\alpha S_k+\beta_1 n_{k-2:k-1}+\beta_2 n_{k-4:k-2}+\beta_3 n_{k-6:k-4}),$$

and $n_{k-2:k-1}$ defines the number of spike counts (across all spike trains) within the previous 10 ms time interval prior to time index k or t_k (with bin size 10 ms). Therefore, the spiking history dependence is described by the number of spike counts in the past three history windows: 10–20 ms, 20–40 ms, and 40–60 ms. The initial parameters were set as $\mu = -0.5$, $\alpha = 1$, $\boldsymbol{\beta} \equiv [\beta_1, \beta_2, \beta_3]^T = \mathbf{0}$. The discrete-time HMM converged after about 100 iterations when the log likelihood stops to increase. After that, we ran the Viterbi algorithm to obtain an initial guess of $\{n, \boldsymbol{\tau}, \boldsymbol{\chi}\}$ for the continuous-time model. It is assumed that if $S_k^{(0)}=S_{k+1}^{(0)}$, the same state spans the region $[k\Delta, (k + 1)\Delta]$ ($\Delta = 10$ ms), while if $S_k^{(0)} \neq S_{k+1}^{(0)}$, then a single jump occurs at time $(k + 0.5)\Delta$. Furthermore, we initialized the parameters of individual spike trains that were obtained from a GLM fit (based on about 500 ms of empirical data analysis; see note 12). The HMM estimation results are summarized in Tables 5 and 6. Since there is no ground truth about the latent process for the real-world data, we compared the HMM's state estimate with that obtained from the threshold-based method. It appears that the HMM tends to discover more state transitions than the threshold-based method (see Table 5), some of which might be false alarms. Figure 8 presents an illustrated example.[15] In order to determine which estimation result (from both methods) is correct, we might require other available information (such as the cortical EEG or hippocampal EEG) to help determine the "true" state.[16] Direct comparison of different methods is difficult for real data since there is no single ground truth. Typically it was found that the HMM method yields more frequent state jumps than the threshold method

[13]A sensible initial parameter value will be an important factor for obtaining a fast convergence of the simulated Markov chain. In practice, one can fit a small spike train data set (with preidentified hidden state values) with a GLM.

[14]We computed the mean and variance of spike counts given all 10 ms time bins and obtained a mean 2.42 and a variance 4.45. The deviance between the mean and variance statistics suggested that the inhomogeneous Poisson probability model is inaccurate, and this fact motivated us to include history-dependent covariates in the rate parameter.

[15]A video demo file is provided online (https://neurostat.mit.edu/zhechen/UpDownDemo.avi) for readers interested in a detailed result comparison for a selected 5 minute recording.

[16]The cortical EEG averages have special waveforms triggered by the start and the end times of the UP state; furthermore, ripple events (150–300 Hz) occur much more frequently during the UP state (Ji & Wilson, 2007).

(see Table 5); this is simply due to the fact that while estimating the hidden state, the algorithm does not consider the neighboring state information and evaluates only the individual likelihood within each single time interval (of 10 ms); this tends to yield many single-state jumps with short durations. In contrast, the threshold-based method is designed to merge those short silent intervals with their neighboring sojourns (see step 3 in appendix B). However, the selection of the threshold is rather ad hoc, and the classification results require many hand-tuned parameter setups (such as kernel smoothing, bin size, and minimum SWS cut-off length), which does not offer a robust and consistent criterion across different data sets.

In order to choose a proper parametric model for the sojourn time duration for the UP and DOWN states, we used the state classification result from the discrete-time HMM (see Figure 9). Based on the histogram data analysis, we chose the exponential pdf as the probability model for the sojourn duration of the DOWN state and the log-normal pdf as the probability model for the sojourn duration of the UP state. We also computed their sample statistics (mean, SD) and used them as the initial parameters for the continuous-time probabilistic model. The lower bounds for the UP and DOWN state duration lengths are both set as 40 ms.

Since the recording time of the real-world spike trains data is rather long (about 60 times longer than the synthetic data), the computational overhead is much greater for the MCEM algorithm. In RJMCMC sampling, 300,000 iterations were run to simulate the Markov chain,[17] and the first 3000 iterations were discarded as the burn-in period. After that, we fed the obtained parameter estimates using the complete data set. After an additional 100 MCEM iterations, the algorithm converged (when the iterative log-likelihood increase is sufficiently small), and we obtained the final parameter estimates. With these estimates, we simulated another 1000 realizations of the semi-Markov chain $\mathcal{S}$ and used them for the final hidden state reconstruction (see equations 3.22 and 3.23). The convergence plots of the semi-Markov chain and the MCEM algorithm are shown in Figure 10.

Several noteworthy comments are in order:

- As a comparison, we also used the estimated hidden state $\{S(t)\}$ to fit a GLM model (using glmfit function in Matlab, $\Delta = 1$ ms) by modeling the history dependence with eight discrete windows (1–5, 5–10, 10–15, 15–20, 20–30, 30–40, 40–50 ms). Upon fitting the GLM model, we obtained the estimated spiking history dependence coefficients for the individual spike trains (see Figure 11); as seen from the results, their curves all have an approximately exponential-decaying shape. Finally, the KS plots and autocorrelation plots are shown in Figures 12 and 13, respectively. Overall, the goodness of fit is quite satisfactory.
- In comparison with the discrete-time HMM-EM method (see Tables 5 and 6), the continuous-time MCEM method yields less frequent state jumps. As a consequence, the MCEM result is accompanied with less short sojourn durations since it allows a potential merge of neighboring sojourns during the RJMCMC procedure (see move type 3 in appendix A) that considers the joint likelihoods of the neighboring sojourns. Furthermore, in comparison with the threshold-based method, the continuous-time semi-Markov model is more powerful in representing the uncertainty as well as inferring the underlying neural dynamics. Its estimated model parameters (the shape of the transition and duration probability density) might reveal the some neural mechanism or physiology behind the transitory dynamics (e.g., the inhibitory period after the last transition event). In our experiments, the MCEM method obtained the lowest estimate of the number of state transitions (see Table 5), yielding a transition

[17]In implementation by Matlab version 7.0, that roughly amounts to about 15 hours of CPU time in a personal computer equipped with an Intel Core 2 Du processor.

occurrence rate about 82 min^{-1} (slightly greater than the rate reported in visual cortex; Ji & Wilson, 2007). Despite its flexibility, the MCEM method is much more computationally intensive than the HMM-EM method. The implementation of HMM is simpler and has a faster convergence speed (the EM algorithm typically converged within 100–200 steps, although the local maximum problem remains). In contrast, the MCEM method relies on simulation of state sequences at every iteration and is required to evaluate the point-process joint likelihood (see equation 3.5) for each possible move. The calculation of every single spike's likelihood contribution is time-consuming and is a bottleneck in the computation. In addition, the convergence speed of the MCEM algorithm becomes slower in the end. This is because when the Markov chain gradually approaches the equilibrium, many moves are rejected and a small modification of the hidden state $\mathcal{S}$ or the parameter $\boldsymbol{\theta}$ would not change very much in terms of the joint log likelihood of the data. This can be seen in the flat plateau of the log-likelihood curve near the end of the convergence in Figure 10b. For the real-world spike train data, the simulation of Markov chain needs to be very long in order to pass through all of move possibilities, especially if the number of potential state transitions is large (here, on the order of thousands). Even so, no optimum stop criterion can be provided with a guarantee; hence, the trade-off between the computation cost and the estimation accuracy remains in any Monte Carlo optimization problem.

- To compare these three classification methods, we also computed the cortical EEG averages (mean ± standard error of mean) triggered by the their UP state starting and ending time stamps, respectively (recall note 15). The results are compared in Figure 14. Although the figures all look similar (due to large timescale), on close examination of the plots, it appears that the EEG averages from the MCEM method result in a more accurate detection of the onset of the UP state.
- Given the MUA spike train data analyzed for the behaving rat, the latent process S_t stays longer during the UP state than the DOWN state, indicating that the population neurons remained dominantly active during SWS.[18] Whether these neuronal firing patterns contain any “memory replay” compared with the earlier firing pattern during the RUN behavior will be the subject of future investigation.

4.3 Firing Pattern Analysis Within the UP States

As observed from the analysis of the recorded multiple spike trains, the somatosensory cortical neurons undergo near-synchronous transitions between the UP and DOWN states, from every tens of milliseconds to a few seconds or so. The neuronal firing activities inside the UP state are mainly characterized by duration length and spiking rate. It would be interesting to see if there are any firing “patterns” embedded in these UP-state periods, in either multiunit or single-unit activity (e.g., Luczak et al., 2007; Ji & Wilson, 2007).

On estimating the latent state process, we obtain two features: one is the log (natural basis) of duration, the other the number of spikes per second per tetrode. After collecting these two features from the experimental data shown earlier, we resort to the clustering tool for feature visualization. The soft-clustering algorithm we use here is a greedy clustering algorithm (Verbeek, Vlassis, & Kröse, 2003) based on fitting a gaussian mixture model. In the greedy learning algorithm, the optimal number of mixtures is automatically determined during the learning process. The algorithm was run 20 times, and the best result (with the highest log likelihood) was chosen (see Figure 15). Hence, the neuronal firing pattern can be characterized by a finite number of parameters (mean and covariance), from which we can compare the different neuronal firing across different animals, different days, different brain regions (cortex

[18]This is in contrast to the anesthetized animals, in which the DOWN states occupy a larger fraction of time than the UP states.

versus hippocampus), different sleep stages (Pre-RUN versus Post-RUN), and so on. Since this article mainly focuses on the statistical modeling methodology, further quantitative data analysis and its link to neurophysiology will be presented and discussed elsewhere.

5 Discussion

5.1 Model Mismatch or Misspecification for the Spike Train

When investigating the real-world recording spike trains data, an important task of computational modeling is to identify the functional form of CIF (see equations 2.3 and 3.3 or 3.4). Unlike the simulated spike train data, the CIF of the real-world spike trains is not known in advance and needs to be identified before the probabilistic inference is carried on. In this article, for simplicity, we have presumed that the CIF can be approximated by a GLM (McCullagh & Nelder, 1989; Truccolo et al., 2005) which includes the hidden state and firing history as variables. We have also assumed that the spike trains across tetrodes are mutually independent. Most likely, the neuronal spiking is influenced not only by its own firing history but also by the other spike trains. Despite these simplifications, we think the models presented here still serve as a valuable first step to represent the temporal dynamics of the observed MUA spike trains. Above all, to quote George Box, “All models are wrong, but some are useful.” In addition, theoretically, given sufficient data and under some regular conditions (Pawitan, 2001), the MLE for a GLM is consistent even when the model (e.g., the link function) is chosen incorrectly (Paninski, 2004). From a practical point of view, we have ignored the possible estimation bias here.

For the real-world spike train data, there is no ground truth available for $\mathcal{S}$. A common practice is to select a small data set, and the UP and DOWN states are first identified by the threshold-based or the HMM method and reconfirmed by human inspection (with extra help of EEG measurements). Based on that information and the assumption that the CIF might be identified by a GLM, we can use the GLM fit for model selection. The model fit would be shown by the deviance and validated by the KS test. If the KS plot falls inside the 95% confidence intervals, it indicates that the CIF model fits well with the given spike train data. Unfortunately, in practice, this is not always the case given only a limited amount of the observed data and an economical size of parameter space for the GLM, indicating a lack of discrepancy between the model and the data.

5.2 Discrete Probability Model for the Sojourn Time

In this article, the sojourn time survival function for the UP and DOWN states is assumed and modeled as being continuous and parametric. More generally, if the histogram analysis of the data indicates that the true distribution is far away from any parametric (exponential or nonexponential) probability density, we might also employ a discretized probability model for the survival probability of the sojourn time. Specifically, let $[a, b]$ denote the range for the sojourn time; we may split the range evenly into L bins and model the discrete probability at each piece as $P_i(x) = \Pr\{a + (i - 1)(b - a)/L \le x < a + i(b - a)/L\}$ $(i = 1, 2, \ldots, L)$. Then the probabilistic model of the sojourn time will be fully characterized by two sets of the parameters, $\{P_i^{up}\}$ and $\{P_j^{down}\}$, where $\sum_{i=1}^{L} P_i^{up}=1$ and $\sum_{j=1}^{L} P_j^{down}=1$. In this case, the inference algorithm will be slightly different in that the M-step of the MCEM algorithm will be modified with a reestimation procedure (see equation 3.18), but the E-step remains unchanged.

5.3 Adding Intermediate States

Although in this article, we have exclusively discussed a two-state (0 and 1) Markov model, it is easy to extend the framework to a general N-state Markovian model. Indeed, it is quite possible to add an intermediate state between DOWN and UP as the transitory state. The reason for this argument arises from the observation from the real-world MUA spike trains: in many circumstances, there is no clear evidence that the MUA are either up-modulated or completely silent. Nor does the EEG in any obvious fashion help to differentiate these ambiguous periods. Unfortunately, how to define a transitory state presumably remains a nontrivial problem, and no attempt was made to explore this direction here.

5.4 Fully Bayesian Inference

In the MCEM algorithm discussed above, we consider Monte Carlo sampling only in the E-step, whereas the M-step uses a standard deterministic optimization procedure. Potentially we can use the MCMC procedure for both state and parameter estimation $\{\mathcal{S}^{(k)}, \boldsymbol{\theta}^{(k)}\}$ for $k = 1, 2, \ldots, M$, from which we can obtain the full posterior distribution $p(\mathcal{S}, \boldsymbol{\theta} \mid \mathcal{Y})$ instead of the marginal $p(\mathcal{S} \mid \boldsymbol{\theta}, \mathcal{Y})$. Take, for example, the parameters associated with the sojourn time pdf; we can define the gamma prior for the exponential distribution or a conjugate prior for the inverse gaussian (Banerjee & Bhattacharyya, 1979); for the parameters associated with the CIF model, we may define a gaussian prior. In this case, the M-step would be replaced by iterative Gibbs sampling. The detailed exploration of such a fully Bayesian inference approach, however, is beyond the scope of this article.

5.5 Limitation of Our Approach

There are several obvious assumptions used in our statistical modeling approach. First, the statistical mutual independence is assumed across neural spike trains, without explicit modeling of the recurrent network activity.[19] Second, the observed data are assumed to be stationary in the sense that the state transition and the CIF parameters are estimated from a long period of spike train recordings when those parameters are assumed to remain constant. Finally, an identical CIF model is also assumed across all neural spike trains. Nevertheless, these limitations by no means diminish the value of the models and methods proposed here, since this article can be treated as a pilot study toward the ultimate modeling goal.

5.6 Future Work

We have considered several future investigation efforts in the line with the work reported here. From a computational modeling point of view, we can extend the model by including continuous-valued observations (e.g., Srinivasan, Eden, Mitter, & Brown, 2007). For instance, the LFP or EEG measurements have been simultaneously recorded from both cortical and hippocampal regions. The detection of K-complexes from the cortical EEG and detection of the sharp wave-ripple complexes (SPW-Rs) from the hippocampal EEG would be beneficial to the identification of UP and DOWN states (Siriota et al., 2003; Battaglia et al., 2004; Ji & Wilson, 2007). Furthermore, it is possible to build a more complex SSM by allowing both continuous- and discrete-valued hidden variables—for instance, a switching SSM where the two latent processes interact with each other (e.g., Ghahramani & Hinton, 2000; Srinivasan et al., 2007).

From a neurophysiological point of view, we are also interested in studying the population neuronal firing causality and latency between the cortex and hippocampus, as well as their spike patterns relevant to the rat's RUN behavior. It is well known that sleep is a key factor that may promote the transfer of memory from the hippocampus to the cortex, and during sleep, replays in these two regions occur synchronously (Mehta, 2007; Ji & Wilson, 2007). Based on the extracelluar recordings (MUA and LFP), it would be interesting to investigate the UP and DOWN activities during multiple processing stages and sites in the cortico-hippocampal

[19]Recently, complementary work has been reported in modeling self-organized recurrent network model of excitatory and inhibitory neurons for spontaneous UP and DOWN state transitions (Kang, Kitano, & Fukai, 2008).

circuits, and the UP and DOWN state transitions can be used to quantify the functional connectivity of the neural circuits. An in-depth exploration of the LFP (e.g., K-complexes, SPW-Rs), with single- and multiunit firing activities in both cortical and hippocampal regions, would be key to understanding memory consolidation during sleep.

6 Conclusion

We have developed both discrete- and continuous-time probabilistic models and inference algorithms for inferring population neurons' UP and DOWN states, using the MUA spike trains. Compared to the deterministic threshold-based method (see appendix B) used in the literature, our probabilistic paradigms offer a stochastic approach to analyze the spike trains as well as provide a generative model to simulate the spike trains. Furthermore, the hidden state estimate is treated as a random variable with certain uncertainty (encoded by its posterior probability), whereas the threshold-based method cannot represent such uncertainties.

The discrete-time HMM provides a reasonable state estimate with a rather fast computing speed. However, the model is restricted to locate the UP and DOWN state transition with a relatively large time bin size (here, 10 ms). Another drawback of the HMM is that it is prone to get stuck in the local solution; in other words, the number of state transitions typically remains unchanged after a few EM iterations. In contrast, one advantage of the continuous-time probabilistic model is that it allows estimating the exact locations of state jumps. By using the RJMCMC sampling technique, the number of jumps as well as the locations of the jumps are allowed to be modified during the inference procedure, which offers a way to escape from the local minimum and tackle the model selection problem. The only shortcoming of the RJMCMC method is its greater computational complexity and the tremendous demand of computational power. In practice, the number of steps required to reach equilibrium often demands sensible initial conditions and diagnostic monitoring during the convergence process. We found that the inference solution obtained from the discrete-time HMM yields a reasonable initial state sequence to feed into the MCMC sampler. Once the number and the locations of the state jumps are determined, we can use the Monte Carlo statistics to infer the latent process. For practitioners who are more concerned about the processing speed than the accuracy of hidden state estimation, the discrete-time HMM might offer a reasonable guess (depending on the data characteristics). Nevertheless, no claim is made here that our proposed models and algorithms could always produce a correct UP or DOWN state classification result. The final justification might still rely on careful human inspection, but our estimation results certainly provide a good starting point with high confidence for follow-up.

In analyzing the simultaneously recorded spike trains, identifying an accurate CIF model is crucial to the probabilistic inference. However, there is no free-lunch-recipe to obtain the ultimate answer. In practice, it often requires some empirical data analysis (Brown, Kass, & Mitra, 2004; Kass, Ventura, & Brown, 2005), such as the interspike interval histogram, firing-rate trend dependence analysis, or fitting a GLM (Truccolo et al., 2005) or a non-Poisson model (Barbieri, Quirk, Frank, Wilson, & Brown, 2001; Brown et al., 2003).

Finally, we hope that our proposed statistical models can shed some light on developing physiologically plausible mechanistic models. A better understanding of the transition mechanism between the UP and DOWN states would also help to improve the statistical description of the data.

Acknowledgments

The research reported here was supported by NIH/NIDA grant R01-DA015644 to E.N.B. and M.A.W., an NIH Director Pioneer Award DP1-OD003646 to E.N.B., and an NIH/NHLBI grant R01-HL084502 to Z.C. and R.B.S.V. was supported by NIH institutional NRSA grant T32 HL07901. We thank R. Haslinger, D. Ji, and D. Nguyen for some helpful discussions. We also thank two anonymous reviewers for their valuable comments that helped to improve the presentation of this article.

Appendix A: Reversible-Jump MCMC

A.1 Background

Reversible-jump MCMC (RJMCMC) is a Metropolis-Hastings-type sampling algorithm with a transdimensional proposal. The term *reversible jump* refers to the ability of the Markov chain to “jump” between two parameter spaces that have different dimensions. The sampler explores the state spaces of variable dimensionality by various modifications through the Metropolis-Hastings proposals. Each Metropolis-Hastings proposal has a respective reverse proposal. For every proposal, the acceptance probability is computed according to a certain rule. The goal of the RJMCMC algorithm is to design efficient moves that allow the simulated Markov chain to reach the desired equilibrium (posterior) distribution within a reasonable amount of time. Unlike the fixed-dimensional MCMC algorithms, RJMCMC allows the state transitions to occur between spaces with different dimensions, say, $\mathcal{S} \rightarrow \mathcal{S}'$, where $dim(\mathcal{S}) \neq dim(\mathcal{S}')$.

In this appendix, we present a detailed elaboration of the RJMCMC algorithm in the context of simulating a continuous-time (semi-) Markov chain for the problem. In the following, we use similar notations and formulations of Ball et al. (1999).

A.2 Derivation

Let n denote the number of jumps between two distinct discrete states in the latent process $\{S(t); 0 \leq t \leq T\}$, where $S(t) \in \{0,1\}$. Let $\mathcal{S} = (n, \boldsymbol{\tau}, \boldsymbol{\chi})$ be a triplet of the (semi-) Markov process, where $\boldsymbol{\tau} = (\tau_0, \tau_1, \ldots, \tau_n)$ is a vector that contains the duration of the sojourn time of $\mathcal{S}$, and $\boldsymbol{\chi} = (\chi_0, \chi_1, \ldots, \chi_n)$ represents the states visited in these sojourns. Let $\nu_0 = 0$, $\nu_j = \sum_{i=0}^{j-1} \tau_i \; (j = 1, 2, \ldots, n+1)$, and $\nu_{n+1} = T$. Furthermore, we assume that for both UP and DOWN states, the sojourn time duration τ has a lower bound a ($\tau \geq a > 0$) but no upper bound; consequently, the associated pdf has to be rectified accordingly.

The following seven types of moves are considered in the Metropolis-type proposal:

1. Move a boundary between two successive sojourns of $\mathcal{S}$. First, decide which boundary to move by sampling j uniformly from $\{0, 1, \ldots, n-1\}$. Let $\chi_j = i_1$, $\chi_{j+1} = i_2$, and then we have two alternative sampling options:[20]

    - Sample u from a uniform distribution $\mathcal{U}(a_{i_1}, \tau_j + \tau_{j+1} - a_{i_2})$, where a_{i_1}, a_{i_2} denote the lower-bound constraints of the sojourn time of states χ_j and χ_{j+1}, respectively. The proposal $\mathcal{S}'$ is obtained by moving the boundary between the jth and $(j+1)$st sojourns from ν_{j+1} to $\nu_j + u$. In this case, $\mathcal{S}' = (n', \boldsymbol{\tau}', \boldsymbol{\chi}')$, where $\tau'_j = u, \tau'_{j+1} = \tau_j + \tau_{j+1} - u$.
    - Sample u from a gaussian distribution $\mathcal{N}(0, \sigma^2)$, where σ^2 denotes the (user-specified) variance parameter. The proposal $\mathcal{S}'$ is obtained by moving the boundary between the jth and $(j+1)$st sojourns from ν_{j+1} to $\nu_{j+1} + u$. In this case, $\mathcal{S}' = (n', \boldsymbol{\tau}', \boldsymbol{\chi}')$, $\tau'_j = \tau_j + u, \tau'_{j+1} = \tau_{j+1} - u$,

    For both sampling options, $n' = n$, $\tau'_l = \tau_l \; (l \neq j, j+1)$, and $\chi'_l = \chi_l \; (l = 1, 2, \ldots, n')$.

[20]Given a reasonably accurate initial state, the second sampling option would be more efficient since its rejection rate would be lower. For the current continuous-time estimation problem, the SD parameter σ in the second sampling option is chosen to be 2 ms, twice that of the bin size.

2. Insert a sojourn in one of the existing sojourns of $\mathcal{S}$. First, sample l^* uniformly from $\{0, 1, \ldots, n\}$, and let $i = \chi_{l^*}$. Determine the state for the inserted sojourn by sampling j from the probability $\tilde{p}_1(j \mid i, \boldsymbol{\theta})$ $(j \neq i)$. In the two-state space, $\tilde{p}_1(j \mid i) = 1$ (i.e., deterministic). Sample u from a uniform distribution $\mathcal{U}(a_i, \tau_{l^*} - a_i)$ (where a_i denotes the lower bound of the sojourn time for state i), and then sample v from the censored version of the exponential (or log-normal or inverse gaussian) distribution with parameters $\boldsymbol{\theta}_j \equiv r_j$ (or $\boldsymbol{\theta}_j \equiv \{\mu_j, \sigma_j\}$ for log normal, or $\boldsymbol{\theta}_j \equiv \{\mu_j, s_j\}$ for the inverse gaussian) truncated at $\tau_{l^*} - u - a_i$, that is, from the distribution that has the following conditional pdf $\tilde{p}(v \mid u)$

$$\tilde{p}\,(v|u) = \frac{p_v(\boldsymbol{\theta}_j; v)}{F_v(\boldsymbol{\theta}_j; \tau_{l*} - u - a_i) - F_v(\boldsymbol{\theta}_j; a_j)} \times (a_j \le v \le \tau_{l*} - u - a_i), \tag{A.1}$$

where a_j denotes the lower bound of the sojourn time for state j. Then a sojourn in state j $(j \neq i)$ of length v is inserted in the l^*th sojourn of $\mathcal{S}$. And the new proposal $\mathcal{S}' = (n', \boldsymbol{\tau}', \boldsymbol{\chi}')$ is given by $n' = n + 2$, $(\tau'_l, \chi'_l) = (\tau_l, \chi_l)(l = 0, 1, l*-1)$, $\tau'_{l*} = u$, $\tau'_{l*+1} = v$, $\tau'_{l*+2} = \tau_{l*} - u - v$, $(\tau'_l, \chi'_l) = (\tau_{l-2}, \chi_{l-2})(l = l*+3, l*+4, \ldots, n+2)$, $\chi'_{l*} = \chi'_{l*+2} = \chi_{l*}(=i)$, $\chi'_{l*+1} = j$. Note that if $\tau_{l^*} - u - 2a_i < a_j$, move 2 will not be executed.

3. Delete an intermediate sojourn of $\mathcal{S}$ whose two adjacent sojourns are in the same state (namely, merge one sojourn with its neighboring sojourns). Sample l^* uniformly from $\{0, 1, \ldots, n - 1\}$, and delete the l^*th sojourn of $\mathcal{S}$. The new $\mathcal{S}'$ is given by $n' = n - 2$, $(\tau'_l, \chi'_l) = (\tau_l, \chi_l)(l = 0, 1, \ldots, l*-2)$, $\tau'_{l*-1} = \tau_{l*-1} + \tau_{l*} + \tau_{l*+1}, \chi'_{l*-1} = \chi_{l*-1}$, $(\tau'_l, \chi'_l) = (\tau_{l+2}, \chi_{l+2})\,(l = l*, l*+1, \ldots, n-2)$.

4. Split the first sojourn of $\mathcal{S}$. First, let $i = \chi_0$, and sample u from the uniform distribution $\mathcal{U}(a_j, \tau_0 - a_i)$, where a_i and a_j denote the lower-bound constraints of the sojourn time for states i and j, respectively ($j \neq i$; in the two-state case, the choice of j is deterministic). The new proposal $\mathcal{S}'$ is given by $n' = n + 1$, $\tau'_0 = u$, $\chi'_0 = j$, $\tau'_1 = \tau_0 - u$, $\chi'_1 = i$, $(\tau'_l, \chi'_l) = (\tau_{l-1}, \chi_{l-1})\,(l = 1, 2, \ldots, n-1)$. Note that if $\tau_0 < a_i + a_j$, move 4 will not be executed.

5. Delete the first sojourn of $\mathcal{S}$. This move is deterministic. The new proposal $\mathcal{S}'$ is given by $n' = n - 1$, $\tau'_0 = \tau_0 + \tau_1$, $\chi'_0 = \chi_1$, $(\tau'_l, \chi'_l) = (\tau_{l+1}, \chi_{l+1})\,(l = 1, 2, \ldots, n-1)$.

6. Split the last sojourn of $\mathcal{S}$. First, let $i = \chi_n$, and sample u from a uniform distribution $\mathcal{U}(a_i, \tau_n)$, where a denotes the lower-bound constraint of the sojourn time for state i. Next, sample j from the distribution $\tilde{p}_3\,(j \mid i)$ ($j \neq i$; in the two-state state space, the choice of j is deterministic). The new proposal $\mathcal{S}'$ is given by $n' = n - 1$, $(\tau'_l, \chi'_l) = (\tau_l, \chi_l)\,(l = 1, 2, \ldots, n-1)$, $\tau'_n = u$, $\chi'_n = \chi_n$, $\tau'_{n+1} = \tau_n - u$, $\chi'_{n+1} = j$. Note that if $\tau_n < a_i$, move 6 will not be executed.

7. Delete the last sojourn of $\mathcal{S}$. This move type is deterministic, and the new proposal $\mathcal{S}'$ is given by $n' = n - 1$, $(\tau'_l, \chi'_l) = (\tau_l, \chi_l)\,(l = 1, 2, \ldots, n-2)$, $\tau'_{n-1} = \tau_{n-1} + \tau_n$, $\chi'_{n-1} = \chi_{n-1}$.

Of the above moves, moves 2 to 7 are discrete and transdimensional, and move 1 is continuous. For the convenience of technical treatment, we classify the seven moves into three classes: $A = \{1\}$, $B = \{2, 4, 6\}$, $C = \{3, 5, 7\}$, which correspond to *boundary move*, *insertion*, and *deletion*, respectively. Specifically, the individual moves in class B are the respective inverses of those in class C.

The move types chosen for the $\mathcal{S}$ update are obtained by sampling independently from the distribution of q_i $(i = 1, 2, \ldots, 7)$ as follows. First, the class of move type is determined by sampling from the distribution (q_A, q_B, q_C), where, for example, q_B represents the probability that a class B move is chosen. Second, if a class B or class C move is chosen, the specific individual move is found by sampling $\tilde{j}$ from a distribution ($\tilde{q}_1$, $\tilde{q}_2$, $\tilde{q}_3$). By this setup, the probability of q_i will be determined by $q_1 = q_A$, $q_2 = q_B\tilde{q}_1$, $q_4 = q_B\tilde{q}_2$, $q_6 = q_B\tilde{q}_3$, $q_3 = q_C\tilde{q}_1$, $q_5 = q_C\tilde{q}_2$, $q_7 = q_C\tilde{q}_3$. For the problem here, we may use the following setup: $(q_A, q_B, q_C) = (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$, and $\tilde{q}_1 \propto \frac{n-1}{n+1}$, $\tilde{q}_2 \propto \frac{1}{n+1}$, $\tilde{q}_3 \propto \frac{1}{n+1}$, such that the normalization constraint $\sum_{i=1}^{7} q_i = 1$ is satisfied. The $\tilde{q}_2$ and $\tilde{q}_3$ correspond to probabilities for selecting the first and last sojourns, respectively, and $\tilde{q}_1$ for the other sojourns.

To compute the acceptance probabilities: for $i = 1, 2, \dots, 7$, let $R^{(i)}(\mathcal{S} \to \mathcal{S}'; \boldsymbol{\theta})$ be the density of the transition kernel associated with the proposal $\mathcal{S}'$ for the move type (i) and let q_i be the probability of choosing the move type (i). Since moves 2 to 7 are discrete (unlike move type 1), their $R^{(i)}$ ($\mathcal{S} \to \mathcal{S}'$; $\boldsymbol{\theta}$) are probabilities instead of probability densities. Hence, the density of the transition kernel for a new proposal $\mathcal{S}'$ is given by

$$R(\mathcal{S} \to \mathcal{S}'; \boldsymbol{\theta}) = \sum_{i=1}^{7} q_i R^{(i)}(\mathcal{S} \to \mathcal{S}'; \boldsymbol{\theta}).$$

Among the seven move proposals, move type 1 does not change the dimension of $\mathcal{S}$, and its acceptance probability is $\mathcal{A} = \min(1, \mathcal{B})$, where

$$\mathcal{B} = \frac{p(\mathcal{S}', \boldsymbol{\theta}, y) R(\mathcal{S}' \to \mathcal{S}; \boldsymbol{\theta})}{p(\mathcal{S}, \boldsymbol{\theta}, y) R(\mathcal{S} \to \mathcal{S}'; \boldsymbol{\theta})}. \tag{A.2}$$

On the other hand, move types 2 to 7 change the dimension of $\mathcal{S}$, and their acceptance probabilities are given by $\mathcal{A} = \min(1, \mathcal{B})$, where

$$\mathcal{B} = \frac{p(\mathcal{S}', \boldsymbol{\theta}, y) R(\mathcal{S}' \to \mathcal{S}; \boldsymbol{\theta})}{p(\mathcal{S}, \boldsymbol{\theta}, y) R(\mathcal{S} \to \mathcal{S}'; \boldsymbol{\theta})} |\mathcal{J}|, \tag{A.3}$$

where $|\mathcal{J}|$ denotes the determinant of the Jacobian. The Jacobian measures the ratio of the volume of two state spaces.

For presentation convenience, we often factorize the acceptance probability as $\mathcal{B} = \mathcal{B}_1\mathcal{B}_2\mathcal{B}_3$ (prior ratio × likelihood ratio × proposal probability ratio),[21] where

$$\mathcal{B}_1 = \frac{p(\mathcal{S}'|\boldsymbol{\theta})}{p(\mathcal{S}|\boldsymbol{\theta})}, \quad \mathcal{B}_2 = \frac{p(y|\mathcal{S}', \boldsymbol{\theta})}{p(y|\mathcal{S}, \boldsymbol{\theta})}, \quad \mathcal{B}_3 = \frac{R(\mathcal{S}' \to \mathcal{S}; \boldsymbol{\theta})}{R(\mathcal{S} \to \mathcal{S}'; \boldsymbol{\theta})}. \tag{A.4}$$

In the context of two-state continuous-time (semi-) Markov chain that is used in this article, we compute these three probability ratios for these seven moves in the following.

[21]To avoid numerical problems, it is more convenient to calculate the ratios in the logarithm domain.

A.3 Move Types

A.3.1 Move Type 1

For the first sampling option, let $i_1 = \chi_j$ and $i_2 = \chi_{j+1}$, and let $\lambda_{i_1}^c = \int_{\nu_j}^{\nu_{j+1}} \lambda^c(t; S_{\nu_j:\nu_{j+1}} = i_1)\, dt$, $\lambda_{i_2}^c = \int_{\nu_{j+1}}^{\nu_{j+2}} \lambda^c(t; S_{\nu_{j+1}:\nu_{j+2}} = i_2)\, dt$, $\tilde{\lambda}_{i_1}^c = \int_{\nu_j}^{\nu_j+u} \lambda^c(t; S_{\nu_j:\nu_j+u} = i_1)\, dt$, and $\tilde{\lambda}_{i_2}^c = \int_{\nu_j+u}^{\nu_{j+2}} \lambda^c(t; S_{\nu_j+u:\nu_{j+2}} = i_2)\, dt$, and let $y_{i_1}^c$, $y_{i_2}^c$, $\tilde{y}_{i_1}^c$, and $\tilde{y}_{i_2}^c$ denote the number of spike counts for spike train c during the time intervals $[\nu_j, \nu_{j+1}]$, $[\nu_{j+1}, \nu_{j+2}]$, $[\nu_j, \nu_j + u]$, and $[\nu_j + u, \nu_{j+2}]$, respectively. The probability ratios for move type 1 are calculated as follows

$$
\begin{aligned}
\mathscr{B}_1^{(1)} &= \frac{\tilde{p}(\boldsymbol{\theta}_{i_1}; u)}{\tilde{p}(\boldsymbol{\theta}_{i_1}; \tau_j)} \frac{\tilde{p}(\boldsymbol{\theta}_{i_2}; \tau_{j+1} + \tau_j - u)}{\tilde{p}(\boldsymbol{\theta}_{i_2}; \tau_{j+1})} \quad \text{(option 1)} \\
\mathscr{B}_2^{(1)} &= \prod_{c=1}^{C} \Bigg\{ \exp(\lambda_{i_1}^c + \lambda_{i_2}^c - \tilde{\lambda}_{i_1}^c - \tilde{\lambda}_{i_2}^c) \\
&\qquad \times \frac{\prod_{k=1}^{\tilde{y}_{i_1}^c} \lambda^c(t_k, \mathcal{S}'_{\nu_j:\nu_j+u}) \prod_{k=1}^{\tilde{y}_{i_2}^c} \lambda^c(t_k, \mathcal{S}'_{\nu_j+u:\nu_{j+2}})}{\prod_{k=1}^{y_{i_1}^c} \lambda^c(t_k, \mathcal{S}_{\nu_j:\nu_{j+1}}) \prod_{k=1}^{y_{i_2}^c} \lambda^c(t_k, \mathcal{S}_{\nu_{j+1}:\nu_{j+2}})} \Bigg\} \quad \text{(option 1)} \\
\mathscr{B}_3^{(1)} &= \frac{R(\mathcal{S}' \to \mathcal{S}; \boldsymbol{\theta})}{R(\mathcal{S} \to \mathcal{S}'; \boldsymbol{\theta})} = 1,
\end{aligned}
$$

where $\tilde{p}(\boldsymbol{\theta}; \cdot)$ is defined by the censored version of the parametric pdf of the sojourn time (for either UP or DOWN state). The ratios for the second sampling option are conceptually similar, and we show only $\mathscr{B}_1^{(1)}$ here:

$$
\mathscr{B}_1^{(1)} = \frac{\tilde{p}\,(\boldsymbol{\theta}_{i_1}; \tau_j + u)}{\tilde{p}\,(\boldsymbol{\theta}_{i_1}; \tau_j)} \frac{\tilde{p}\,(\boldsymbol{\theta}_{i_2}; \tau_{j+1} - u)}{\tilde{p}\,(\boldsymbol{\theta}_{i_2}; \tau_{j+1})} \quad \text{(option 2).}
$$

A.3.2 Move Type 2

Let $i_1 = i$, $i_2 = j$, and let $y_{i_1}^c$, $\tilde{y}_{i_1}^c$, $\tilde{y}_{i_2}^c$, $\widehat{y}_{i_1}^c$ denote the number of spike counts for spike train c during the time intervals $[\nu_{l*}, \nu_{l*+1}]$, $[\nu_{l*}, \nu_{l*} + u]$, $[\nu_{l*} + u, \nu_{l*} + u + v]$, and $[\nu_{l*} + u + v, \nu_{l*+1}]$, respectively. Let $\lambda_{i_1}^c = \int_{\nu_{l*}}^{\nu_{l*+1}} \lambda^c(t; S_{\nu_{l*}:\nu_{l*+1}} = i_1)\, dt$, $\tilde{\lambda}_{i_1}^c = \int_{\nu_{l*}}^{\nu_{l*}+u} \lambda^c(t; S_{\nu_{l*}:\nu_{l*}+u} = i_1)\, dt$, $\tilde{\lambda}_{i_2}^c = \int_{\nu_{l*}+u}^{\nu_{l*}+u+v} \lambda^c(t; S_{\nu_{l*}+u:\nu_{l*}+u+v} = i_2)\, dt$, and $\widehat{\lambda}_{i_1}^c = \int_{\nu_{l*}+u+v}^{\nu_{l*+1}} \lambda^c(t; S_{\nu_{l*}+u+v:\nu_{l*+1}} = i_1)\, dt$. Then

$$
\begin{aligned}
\mathscr{B}_1^{(2)} &= \frac{\tilde{p}(\boldsymbol{\theta}_{i_1}; u)\tilde{p}(\boldsymbol{\theta}_{i_2}; v)\tilde{p}(\boldsymbol{\theta}_{i_1}; \tau_{l*} - u - v)}{\tilde{p}(\boldsymbol{\theta}_{i_1}; \tau_{l*})} \\
\mathscr{B}_2^{(2)} &= \prod_{c=1}^{C} \Bigg\{ \exp(\lambda_{i_1}^c - \tilde{\lambda}_{i_1}^c - \tilde{\lambda}_{i_2}^c - \widehat{\lambda}_{i_1}^c) \\
&\quad \times \frac{\prod_{k=1}^{\tilde{y}_{i_1}^c} \lambda^c(t_k, \mathcal{S}'_{\nu_{l*}:\nu_{l*}+u}) \prod_{k=1}^{\tilde{y}_{i_2}^c} \lambda^c(t_k, \mathcal{S}'_{\nu_{l*}+u:\nu_{l*}+u+v}) \prod_{k=1}^{\widehat{y}_{i_1}^c} \lambda^c(t_k, \mathcal{S}'_{\nu_{l*}+u+v:\nu_{l*+1}})}{\prod_{k=1}^{y_{i_1}^c} \lambda^c(t_k, \mathcal{S}_{\nu_{l*}:\nu_{l*+1}})} \Bigg\} \\
\mathscr{B}_3^{(2)} &= \frac{R(\mathcal{S}' \to \mathcal{S}; \boldsymbol{\theta})}{R(\mathcal{S} \to \mathcal{S}'; \boldsymbol{\theta})} = \frac{\frac{q_3}{n+1}}{\frac{q_2}{n+1} \frac{1}{\tau_{l*} - 2a_i} \tilde{p}(v|u)} = \frac{\tau_{l*} - 2a_i}{\tilde{p}(v|u)},
\end{aligned}
$$

where in computing $\mathscr{B}_3^{(2)}$, we have used $q_2 = q_3$ and $p(u) \dfrac{1}{\tau_{l*} - 2a_i}$, and $\tilde{p}(v \mid u)$ is given by equation A.1. Since move type 2 changes the dimension of the $\mathcal{S}$, we need to compute the associated Jacobian's determinant. Note that $\boldsymbol{\tau}'$ is obtained from $\boldsymbol{\tau}$ from an invertible deterministic function $\boldsymbol{\tau}' \leftarrow \mathcal{T}(\boldsymbol{\tau}, u, v) = (\tau_0, \tau_1, \ldots, \tau_{l^*-1}, u, v, \tau_{l^*} - a_i - u - v, \tau_{l^*+1}, \tau_{l^*+2}, \ldots, \tau_n)$, whose Jacobian is then given by

$$
\begin{aligned}
|\mathcal{J}| &= \left| \frac{\partial \mathcal{S}'(u,v,\tau_l^* - a_i - u - v))}{\partial \mathcal{S}(u,v,\tau_l^*)} \right| \\
&= \begin{vmatrix} \frac{\partial u}{\partial \tau_{i*}} & \frac{\partial u}{\partial u} & \frac{\partial u}{\partial v} \\ \frac{\partial v}{\partial \tau_{i*}} & \frac{\partial v}{\partial u} & \frac{\partial v}{\partial v} \\ \frac{\partial(\tau_{l*} - a_i - u - v)}{\partial \tau_{l*}} & \frac{\partial(\tau_{l*} - a_i - u - v)}{\partial u} & \frac{\partial(\tau_{l*} - a_i - u - v)}{\partial v} \end{vmatrix} = \begin{vmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & -1 & -1 \end{vmatrix} = 1.
\end{aligned}
$$

A.3.3 Move Type 3

Moves 3 and 2 are inverses of each other. Let $i_1 = \chi_{l^*-1}$ and $i_2 = \chi_{l^*}$, and let $\lambda_{i_1}^c = \int_{\nu_{l*-1}}^{\nu_{l*}} \lambda^c(t; S_{\nu_{l*-1}:\nu_{l*}} = i_1)\,dt$, $\lambda_{i_2}^c = \int_{\nu_{l*}}^{\nu_{l*+1}} \lambda^c(t; S_{\nu_{l*}:\nu_{l*+1}} = i_2)\,dt$, $\widehat{\lambda}_{i_1}^c = \int_{\nu_{l*+1}}^{\nu_{l*+2}} \lambda^c(t; S_{\nu_{l*+1}:\nu_{l*+2}} = i_1)\,dt$, $\overset{\sim c}{\lambda}_{i_1} = \int_{\nu_{l*-1}}^{\nu_{l*+2}} \lambda^c(t; S_{\nu_{l*-1}:\nu_{l*+2}} = i_1)\,dt$, and let $y_{i_1}^c$, $y_{i_2}^c$, $\widehat{y}_{i_1}$, and $\overset{\sim c}{y}_{i_1}$ denote the numbers of spike counts for spike train c within the intervals $[\nu_{l^*-1}, \nu_{l^*}]$, $[\nu_{l^*}, \nu_{l^*+1}]$, $[\nu_{l^*+1}, \nu_{l^*+2}]$, and $[\nu_{l^*-1}, \nu_{l^*+2}]$, respectively. It is noted that $\boldsymbol{\tau}'$ can be obtained from an invertible deterministic function $\mathcal{T}(\boldsymbol{\tau}) = (\tau_0, \tau_1, \ldots, \tau_{l^*-1}, \tau_{l^*+1}, \tau_{l^*+2}, \ldots, \tau_n)$, and $|\mathcal{J}| = 1$. The probability ratios are given as

$$
\begin{aligned}
\mathscr{B}_1^{(3)} &= \frac{\tilde{p}(\boldsymbol{\theta}_{i_1}; \tau_{l*-1} + \tau_{l*} + \tau_{l*+1})}{\tilde{p}(\boldsymbol{\theta}_{i_1}, \tau_{l*-1})\tilde{p}(\boldsymbol{\theta}_{i_2}, \tau_{l*})\tilde{p}(\boldsymbol{\theta}_{i_1}, \tau_{l*+1})} \\
\mathscr{B}_2^{(3)} &= \prod_{c=1}^{C} \left\{ \exp(\lambda_{i_1}^c + \lambda_{i_2}^c + \widehat{\lambda}_{i_1}^c - \overset{\sim c}{\lambda}_{i_1}) \right. \\
&\quad \left. \times \frac{\prod_{k=1}^{\overset{\sim c}{y}_{i_1}} \lambda^c(t_k, \mathcal{S}'_{\nu_{l*-1}:\nu_{l*+2}})}{\prod_{k=1}^{y_{i_1}^c} \lambda^c(t_k, \mathcal{S}_{\nu_{l*-1}:\nu_{l*}}) \prod_{k=1}^{y_{i_2}^c} \lambda^c(t_k, \mathcal{S}_{\nu_{l*}:\nu_{l*+1}}) \prod_{k=1}^{\widehat{y}_{i_1}^c} \lambda^c(t_k, \mathcal{S}_{\nu_{l*+1}:\nu_{l*+2}})} \right\} \\
\mathscr{B}_3^{(3)} &= \frac{\frac{q_2}{n+1} \frac{1}{(\tau_{l*-1} + \tau_{l*} + \tau_{l*+1})} \tilde{p}_v(\tau_{l*} | \tau_{l*+1})}{\frac{q_3}{n+1}} = \frac{\tilde{p}_v(\tau_{l*} | \tau_{l*+1})}{\tau_{l*-1} + \tau_{l*} + \tau_{l*+1}}
\end{aligned}
$$

where $q_2 = q_3$, and $\tilde{p}_v(\tau_{l^*} \mid \tau_{l^*+1})$ is determined from

$$
\tilde{p}_v(\tau_{l*} | \tau_{l*+1}) = \frac{p(\boldsymbol{\theta}_{i_2}; \tau_{l*})}{\int_{a_{i_2}}^{\tau_{l*} + \tau_{l*+1}} p(\boldsymbol{\theta}_{i_2}; z)\,dz}.
$$

A.3.4 Move Type 4

Let $i_1 = j$, $i_2 = i$, and let $\lambda_{i_2}^c = \int_0^{\nu_1} \lambda^c(t; S_{0:\nu_1} = i_2)\,dt$, $\overset{\sim c}{\lambda}_{i_2} = \int_0^u \lambda^c(t; S_{0:u} = i_2)\,dt$, $\overset{\sim c}{\lambda}_{i_1} = \int_u^{\nu_1} \lambda^c(t; S_{u:\nu_1} = i_1)\,dt$, and let $y_{i_2}^c$, $\overset{\sim c}{y}_{i_2}$, and $\overset{\sim c}{y}_{i_1}$ denote the number of spike counts for spike train c observed within the intervals $[0, \nu_1]$, $[0, u]$, and $[u, \nu_1]$, respectively. Let π_{i_1} and π_{i_2} denote the prior probabilities of the initial sojourn in state i_1 and i_2, respectively. Then we have

$$\mathscr{B}_1^{(4)}=\frac{\pi_{i_1}}{\pi_{i_2}}\ \frac{\tilde{p}(\boldsymbol{\theta}_{i_1};u)\tilde{p}(\boldsymbol{\theta}_{i_2};\tau_0-u)}{\tilde{p}(\boldsymbol{\theta}_{i_2};\tau_0)}$$

$$\mathscr{B}_2^{(4)}=\prod_{c=1}^{C}\left\{\exp(\lambda_{i_2}^c-\tilde{\lambda}_{i_2}^c-\tilde{\lambda}_{i_1}^c)\frac{\prod_{k=1}^{\tilde{y}_{i_2}^c}\lambda^c(t_k,\mathcal{S}'_{0:u})\prod_{k=1}^{\tilde{y}_{i_1}^c}\lambda^c(t_k,\mathcal{S}'_{u:\nu_1})}{\prod_{k=1}^{y_{i_2}^c}\lambda^c(t_k,\mathcal{S}_{0:\nu_1})}\right\}$$

$$\mathscr{B}_3^{(4)}=\frac{q_5}{q_4\frac{1}{\tau_0-a_{i_2}-a_{i_1}}}=\tau_0-a_{i_2}-a_{i_1},$$

where $q_4 = q_B\tilde{q}_2 = q_C\tilde{q}_2 = q_5$. Note that $\boldsymbol{\tau}'$ is obtained from $\boldsymbol{\tau}$ from an invertible deterministic function $\boldsymbol{\tau}' \leftarrow \mathcal{T}(\boldsymbol{\tau}, u) = (u, \tau_0 - u, \tau_1, \tau_2, \ldots, \tau_n)$. Then it follows that

$$|\mathcal{J}|=\begin{vmatrix}\frac{\partial u}{\partial \tau_0} & \frac{\partial u}{\partial u}\\ \frac{\partial(\tau_0-u)}{\partial \tau_0} & \frac{\partial(\tau_0-u)}{\partial u}\end{vmatrix}=\begin{vmatrix}0 & 1\\ 1 & -1\end{vmatrix}=1.$$

A.3.5 Move Type 5

Moves 5 and 4 are inverses of each other. Let $i_1 = \chi_0$, $i_2 = \chi_1$, $\lambda_{i_1}^c=\int_0^{\nu_1}\lambda^c(t;S_{0:\nu_1}=i_1)\,dt$, $\lambda_{i_2}^c=\int_{\nu_1}^{\nu_2}\lambda^c(t;S_{\nu_1:\nu_2}=i_1)\,dt$, $\tilde{\lambda}_{i_2}^c=\int_0^{\nu_1}\lambda^c(t;S_{0:\nu_2}=i_2)\,dt$, and let $y_{i_1}^c$, $y_{i_2}^c$, and $\tilde{y}_{i_2}^c$ denote the observed number of spike counts for spike train c within the intervals $[0, \nu_1]$, $[\nu_1, \nu_2]$, and $[0, \nu_2]$, respectively. Then we have

$$\mathscr{B}_1^{(5)}=\frac{\pi_{i_2}}{\pi_{i_1}}\ \frac{\tilde{p}(\boldsymbol{\theta}_{i_2};\tau_0+\tau_1)}{\tilde{p}(\boldsymbol{\theta}_{i_1};\tau_0)\tilde{p}(\boldsymbol{\theta}_{i_2};\tau_1)}$$

$$\mathscr{B}_2^{(5)}=\prod_{c=1}^{C}\left\{\exp(\lambda_{i_1}^c+\lambda_{i_2}^c-\tilde{\lambda}_{i_2}^c)\frac{\prod_{k=1}^{\tilde{y}_{i_2}^c}\lambda^c(t_k,\mathcal{S}'_{0:\nu_2})}{\prod_{k=1}^{y_{i_1}^c}\lambda^c(t_k,\mathcal{S}_{0:\nu_1})\prod_{k=1}^{y_{i_2}^c}\lambda^c(t_k,\mathcal{S}_{\nu_1:\nu_2})}\right\}$$

$$\mathscr{B}_3^{(5)}=\frac{q_4\frac{1}{\tau_0+\tau_1}}{q_5}=\frac{1}{\tau_0+\tau_1},$$

where $q_4 = q_5$. Similarly, $\boldsymbol{\tau}' \leftarrow \mathcal{T}(\boldsymbol{\tau}) = (\tau_0 + \tau_1, \tau_2, \ldots, \tau_n)$, and $|\mathcal{J}| = 1$.

A.3.6 Move Type 6

Let $i_1 = i$, $i_2 = j$, $\lambda_{i_1}^c=\int_{\nu_n}^{\nu_{n+1}}\lambda^c(t;S_{\nu_n:\nu_{n+1}}=i_1)\,dt$, $\tilde{\lambda}_{i_1}^c=\int_{\nu_n}^{\nu_n+u}\lambda^c(t;S_{\nu_n:\nu_n+u}=i_1)\,dt$, $\tilde{\lambda}_{i_2}^c=\int_{\nu_n+u}^{\nu_{n+1}}\lambda^c(t;S_{\nu_n+u:\nu_{n+1}}=i_2)\,dt$, and let $y_{i_1}^c$, $\tilde{y}_{i_1}^c$, and $\tilde{y}_{i_2}^c$ denote the number of spike counts for spike train c observed within the intervals $[\nu_n, \nu_{n+1}]$, $[\nu_n, \nu_n + u]$, and $[\nu_n + u, \nu_{n+1}]$, respectively. Then we have

$$\mathscr{B}_1^{(6)}=\frac{\tilde{p}(\boldsymbol{\theta}_{i_1};u)\tilde{p}(\boldsymbol{\theta}_{i_2};\tau_n-u)}{\tilde{p}(\boldsymbol{\theta}_{i_1};\tau_n)}$$

$$\mathscr{B}_2^{(6)}=\prod_{c=1}^{C}\left\{\exp(\lambda_{i_1}^c-\tilde{\lambda}_{i_2}^c-\tilde{\lambda}_{i_2}^c)\times\frac{\prod_{k=1}^{\tilde{y}_{i_1}^c}\lambda^c(t_k,\mathcal{S}'_{\nu_n:\nu_n+u})\prod_{k=1}^{\tilde{y}_{i_2}^c}\lambda^c(t_k,\mathcal{S}'_{\nu_n+u:\nu_{n+1}})}{\prod_{k=1}^{y_{i_1}^c}\lambda^c(t_k,\mathcal{S}_{\nu_n:\nu_{n+1}})}\right\}$$

$$\mathscr{B}_3^{(6)}=\frac{q_7}{q_6\frac{1}{\tau_n-a_{i_1}}}=\tau_n-a_{i_1},$$

where $q_6 = q_7$. Similarly $\boldsymbol{\tau}' \leftarrow \mathcal{T}(\boldsymbol{\tau}, u) = (\tau_0, \tau_1, \ldots, \tau_{n-1}, u, \tau_n - u)$, and

$$|\mathcal{J}| = \begin{vmatrix} \frac{\partial u}{\partial u} & \frac{\partial u}{\partial \tau_n} \\ \frac{\partial(\tau_n - u)}{\partial u} & \frac{\partial(\tau_n - u)}{\partial \tau_n} \end{vmatrix} = \begin{vmatrix} 1 & 0 \\ -1 & 1 \end{vmatrix} = 1.$$

A.3.7 Move Type 7

Moves 7 and 6 are inverses of each other. Let $i_1 = \chi_{n-1}$, $i_2 = \chi_n$, $\lambda^c_{i_1} = \int_{v_{n-1}}^{v_n} \lambda^c(t; S_{v_{n-1}:v_n} = i_1)\, dt$, $\lambda^c_{i_2} = \int_{v_n}^{v_{n+1}} \lambda^c(t; S_{v_n:v_{n+1}} = i_2)\, dt$, $\overset{\sim c}{\lambda}_{i_1} = \int_{v_{n-1}}^{v_{n+1}} \lambda^c(t; S'_{v_{n-1}:v_{n+1}} = i_1)\, dt$, and let $y^c_{i_1}$, $y^c_{i_2}$, and $\overset{\sim c}{y}_{i_1}$ denote the observed number of spike counts for spike train c within the intervals $[v_{n-1}, v_n]$, $[v_n, v_{n+1}]$, and $[v_{n-1}, v_{n+1}]$, respectively. Then we have

$$\mathscr{B}^{(7)}_1 = \frac{\tilde{p}(\boldsymbol{\theta}_{i_1}; \tau_{n-1} + \tau_n)}{\tilde{p}(\boldsymbol{\theta}_{i_1}; \tau_{n-1})\tilde{p}(\boldsymbol{\theta}_{i_2}; \tau_n)}$$

$$\mathscr{B}^{(7)}_2 = \prod_{c=1}^{C} \left\{ \exp(\lambda^c_{i_1} + \lambda^c_{i_2} - \overset{\sim c}{\lambda}_{i_1}) \frac{\prod_{k=1}^{\overset{\sim c}{y}_{i_1}} \lambda^c(t_k, \mathcal{S}'_{v_{n-1}:v_{n+1}})}{\prod_{k=1}^{y^c_{i_1}} \lambda^c(t_k, \mathcal{S}_{v_{n-1}:v_n}) \prod_{k=1}^{y^c_{i_2}} \lambda^c(t_k, \mathcal{S}_{v_n:v_{n+1}})} \right\}$$

$$\mathscr{B}^{(7)}_3 = \frac{q_6 \frac{1}{\tau_{n-1} + \tau_n}}{q_7} = \frac{1}{\tau_{n-1} + \tau_n},$$

where $q_6 = q_7$. In addition, we have $\boldsymbol{\tau}' \leftarrow \mathcal{T}(\boldsymbol{\tau}) = (\tau_0, \ldots, \tau_{n-2}, \tau_{n-1} + \tau_n)$ and $|\mathcal{J}| = 1$.

A.4 Heuristics for Efficient RJMCMC Sampling

The experimental recordings are relatively long (varying 15–30 minutes for different rats or dates), and the MCMC sampling for the continuous-time model (with 1 ms bin size) is quite time-consuming. We need to design an efficient (problem-specific) sampling procedure for tackling this problem. One important issue is the initialization of state. As discussed earlier, this will be obtained from the estimation result of the discrete-time HMM. Another issue is to design data-driven MCMC proposals (e.g., Tu & Zhu, 2002) that “intelligently” select moves that also satisfy the detailed balance condition. Specifically, we use a few heuristics in carrying out RJMCMC sampling:

- Move type 1: Given a reasonably initialized state, use option 2 instead of option 1.
- Move type 2: Implement it favorably for those long sojourn time durations, and execute it only for those sojourn time durations with at least four times the minimum length.
- Move type 3: Implement it favorably for those short sojourn time durations.
- Move type 4: Execute it only when the initial sojourn time has at least four times the minimum length.
- Move type 6: Execute it only when the final sojourn time has at least four times the minimum length.

As far as the current UP and DOWN estimation problem is concerned, move types 1 and 3 are the most important ones. When implementing move type 3, we employ a heuristic importance sampling trick. Specifically, the probability of choosing a sojourn time to merge with its neighboring sojourns is inversely proportional to the sojourn length: the shorter the duration, the more likely to be picked out to be merged. Similarly, this trick can be utilized in move type 2 to determine where to split a DOWN state sojourn. The probability of the selected position will be inversely proportional to the observed number of instantaneous MUA spike counts.

A.5 Special Example

In what follows, we derive a special example, in which the sojourn time durations of both UP and DOWN states are modeled by a censored exponential distribution as given in equation 3.12. This example can be viewed as a special case of the result from Ball et al. (1999) in which no constraint was imposed for the pdf. Let r_0 and r_1 denote the rate parameters associated with the exponential distribution for the DOWN and UP states, respectively. And let $a_0 > 0$ and $a_1 > 0$ denote the lower bounds of the sojourn durations for the DOWN and UP states, respectively. The probability ratios $\mathcal{B}_1$ and $\mathcal{B}_3$ for the seven move types are as follows:

- Move type 1:

$$\begin{aligned}
\mathscr{B}_1^{(1)} &= \frac{c_1 r_{i_1} \exp(-r_{i_1} u) c_2 r_{i_2} \exp[-r_{i_2}(\tau_{j+1}+\tau_j-u)]}{c_3 r_{i_1} \exp(-r_{i_1}\tau_j) c_4 r_{i_2} \exp(-r_{i_2}\tau_{j+1})} \\
&= \frac{c_1 c_2 \exp[(r_{i_1}-r_{i_2})(\tau_i-u)]}{c_3 c_4} \qquad \text{(option 1)} \\
\mathscr{B}_1^{(1)} &= \frac{c_5 r_{i_1} \exp[-r_{i_1}(\tau_j+u)] c_5 r_{i_2} \exp[-r_{i_2}(\tau_{j+1}-u)]}{c_3 r_{i_1} \exp(-r_{i_1}\tau_j) c_4 r_{i_2} \exp(-r_{i_2}\tau_{j+1})} \\
&= \frac{c_5 c_6 \exp[(r_{i_1}-r_{i_2})u]}{c_3 c_4} \qquad \text{(option 2)} \\
\mathscr{B}_3^{(1)} &= 1 \qquad \text{(option 1 and 2)},
\end{aligned}$$

  where c_1, c_2, c_3, c_4, c_5, c_6 are the normalized coefficients (details are ignored here; see the description after equation 3.13)

- Move type 2:

$$\begin{aligned}
\mathscr{B}_1^{(2)} &= \frac{c_1 r_{i_1} \exp(-r_{i_1} u) c_2 r_{i_2} \exp(-r_{i_2} v) c_3 r_{i_1} \exp[-r_{i_1}(\tau_{l*}-u-v)]}{c_4 r_{i_1} \exp(-r_{i_1}\tau_{l*})} \\
&= \frac{c_1 c_2 c_3 r_{i_1} r_{i_2} \exp[(r_{i_1}-r_{i_2})v]}{c_4} \\
\mathscr{B}_3^{(2)} &= \frac{(\tau_{l*}-2a_{i_1})\left(\exp(-r_{i_2}a_{i_2})-\exp[-r_{i_2}(\tau_{l*}-u-a_{i_1})]\right)}{r_{i_2}\exp(-r_{i_2}v)},
\end{aligned}$$

  where c_1, c_2, c_3, c_4 are the normalized coefficients

- Move type 3:

$$\begin{aligned}
\mathscr{B}_1^{(3)} &= \frac{c_1 r_{i_1} \exp[-r_{i_1}(\tau_{l*-1}+\tau_{l*}+\tau_{l*+1})]}{c_2 r_{i_1} \exp(-r_{i_1}\tau_{l*-1}) c_3 r_{i_2} \exp(-r_{i_2}\tau_{l*}) c_4 r_{i_1} \exp(-r_{i_1}\tau_{l*+1})} \\
&= \frac{c_1 \exp[(r_{i_2}-r_{i_1})\tau_{l*}]}{c_2 c_3 c_4 r_{i_1} r_{i_2}} \\
\mathscr{B}_3^{(3)} &= \frac{r_{i_2}\exp(-r_{i_2}\tau_{l*})}{(\tau_{l*-1}+\tau_{l*}+\tau_{l*+1})\left(\exp(-r_{i_2}a_{i_2})-\exp[-r_{i_2}(\tau_{l*}+\tau_{l*+1})]\right)},
\end{aligned}$$

  where c_1, c_2, c_3, c_4 are the normalized coefficients

- Move type 4:

$$\begin{aligned}
\mathscr{B}_1^{(4)} &= \frac{\pi_{i_1}}{\pi_{i_2}} \frac{c_1 r_{i_1} \exp(-r_{i_1} u) c_2 r_{i_2} \exp[-r_{i_2}(\tau_0-u)]}{c_3 r_{i_2} \exp(-r_{i_2}\tau_0)} \\
&= \frac{\pi_{i_1}}{\pi_{i_2}} \frac{c_1 c_2 r_1 \exp[(r_{i_2}-r_{i_1})u]}{c_3} \\
\mathscr{B}_3^{(4)} &= \tau_0 - a_{i_1} - a_{i_2},
\end{aligned}$$

  where c_1, c_2, c_3 are the normalized coefficients

- Move type 5:

$$\begin{aligned}\mathscr{B}_1^{(5)}&=\frac{\pi_{i_2}}{\pi_{i_1}}\frac{c_1 r_{i_2}\exp[-r_{i_2}(\tau_0+\tau_1)]}{c_2 r_{i_1}\exp(-r_{i_1}\tau_0)c_3 r_{i_2}\exp(-r_{i_2}\tau_1)}\\&=\frac{\pi_{i_2}}{\pi_{i_1}}\frac{c_1\exp[(r_{i_1}-r_{i_2})\tau_0]}{c_2 c_3 r_1}\\\mathscr{B}_3^{(5)}&=\frac{1}{\tau_0+\tau_1},\end{aligned}$$

where c_1, c_2, c_3 are the normalized coefficients

- Move type 6:

$$\begin{aligned}\mathscr{B}_1^{(6)}&=\frac{c_1 r_{i_1}\exp(-r_{i_1}u)c_2 r_{i_2}\exp[-r_{i_2}(\tau_n-u)]}{c_3 r_{i_1}\exp(-r_{i_1}\tau_n)}\\&=\frac{c_1 c_2 r_2\exp[(r_{i_1}-r_{i_2})(\tau_n-u)]}{c_3}\\\mathscr{B}_3^{(6)}&=\tau_n-a_{i_1},\end{aligned}$$

where c_1, c_2, c_3 are the normalized coefficients

- Move type 7:

$$\begin{aligned}\mathscr{B}_1^{(7)}&=\frac{c_1 r_{i_1}\exp[-r_{i_1}(\tau_{n-1}+\tau_n)]}{c_2 r_{i_1}\exp(-r_{i_1}\tau_{n-1})c_3 r_{i_2}\exp(-r_{i_2}\tau_n)}=\frac{c_1\exp[(r_{i_2}-r_{i_1})\tau_n]}{c_2 c_3 r_2}\\\mathscr{B}_3^{(7)}&=\frac{1}{\tau_{n-1}+\tau_n},\end{aligned}$$

where c_1, c_2, c_3 are the normalized coefficients.

Appendix B: Threshold-Based Method for Classifying UP and DOWN States

The standard threshold-based method for determining the UP and DOWN states based on MUA spike trains (Ji & Wilson, 2007) consists of three major steps.

First, we bin the spike trains into 10 ms windows and calculate the raw spike counts for all time intervals. The raw count signal is smoothed by a gaussian window with an SD of 30 ms to obtain the smoothed count signal over time. We then calculate the first minimum (count threshold value) of the smoothed spike count histogram during SWS. As the spike count has been smoothed, the count threshold value may be a noninteger value.

Second, based on the count threshold value, we determine the active and silent periods for all 10 ms bins. The active periods are set to 1, and silent periods are set to 0. Next, the duration lengths of all silent periods are computed. We then calculate the first local minimum (gap threshold) of the histogram of the silent period durations.

Third, based on the gap threshold value, we merge those active periods separated by silent periods in duration less than the gap threshold. The resultant active and silent periods are classified as the UP and DOWN states, respectively. Finally, we recalculate the duration lengths of all UP and DOWN state periods and compute their respective histograms and sample statistics (min, max, median, mean, SD).

In summary, the choices of the spike count threshold and the gap threshold will directly influence the UP and DOWN state classification and their statistics (in terms of duration length and occurrence frequency). However, the optimal choices of these two hand-tuned parameters are rather ad hoc and dependent on several issues (e.g., kernel smoothing, bin size; see Figure 16 for an illustration). In some cases, no minimum can be found in the smoothed histogram, and then the choice of the threshold is problematic. Note that the procedure will need to be repeated for different data sets such that the UP and DOWN states statistics can be compared.

References

Abeles M, Bergman H, Gat I, Meilijson I, Seidemann E, Tishby N, et al. Cortical activity flips among quasi-stationary states. Proc Natl Acad Sci USA 1995;92:8616–8620. [PubMed: 7567985]

Achtman N, Afshar A, Santhanam G, Yu BM, Ryu SI, Shenoy KV. Free paced high-performance brain-computer interfaces. Journal of Neural Engineering 2007;4:336–347. [PubMed: 17873435]

Albert PS. A two-state Markov mixture model for a time series of epileptic seizure counts. Biometrics 1991;47:1371–1381. [PubMed: 1786324]

Ball FG, Cai Y, Kadane JB, O'Hagan A. Bayesian inference for ion-channel gating mechanisms directly from single-channel recordings, using Markov chain Monte Carlo. Proceedings of the Royal Society of London, A 1999;455:2879–2932.

Bannerjee AK, Bhattacharyya GK. Bayesian results for the inverse gaussian distribution with an application. Technometrics 1979;21(2):247–251.

Barbieri R, Quirk MC, Frank LM, Wilson MA, Brown EN. Construction and analysis of non-Poisson stimulus response models of neural spike train activity. Journal of Neuroscience Methods 2001;105:25–37. [PubMed: 11166363]

Battaglia FP, Sutherland GR, McNaughton BL. Hippocampal sharp wave bursts coincide with neocortical "up-state" transitions. Learning and Memory 2004;11:697–704. [PubMed: 15576887]

Baum L. An inequality and associated maximization technique in statistical estimation for probabilistic function of Markov processes. Inequalities 1972;3:1–8.

Baum LE, Petrie T, Soules G, Weiss N. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. Annals of Mathematical Statistics 1970;41(1):164–171.

Bellman, R. Dynamic programming. Princeton, NJ: Princeton University Press; 1957.

Brillinger DR. Maximum likelihood analysis of spike trains of interacting nerve cells. Biological Cybernetics 1988;59:189–200. [PubMed: 3179344]

Brooks S, Giudici P, Roberts G. Efficient construction of reversible jump MCMC proposal distributions (with discussion). Journal of the Royal Society of London, Series B 2003;65(1):3–56.

Brown, EN. Theory of point processes for neural systems. In: Chow, CC.; Gutkin, B.; Hansel, D.; Meunier, C.; Dalibard, J., editors. Methods and models in neurophysics. Amsterdam: Elsevier; 2005. p. 691-727.

Brown, EN.; Barbieri, R.; Eden, UT.; Frank, LM. Likelihood methods for neural data analysis. In: Feng, J., editor. Computational neuroscience: A comprehensive approach. London: CRC Press; 2003. p. 253-286.

Brown EN, Barbieri R, Ventura V, Kass RE, Frank LM. The time-rescaling theorem and its application to neural spike data analysis. Neural Comput 2002;14(2):325–346. [PubMed: 11802915]

Brown EN, Kass RE, Mitra PP. Multiple neural spike train data analysis: State-of-the-art and future challenges. Nature Neuroscience 2004;7:456–461.

Buzsáki, G. Rhythms of the brain. New York: Oxford University Press; 2006.

Chan KS, Ledolter J. Monte Carlo EM estimation for time series models involving counts. Journal of the American Statistical Association 1995;90:242–252.

Chung SH, Krishnamurthy V, Moore JB. Adaptive processing techniques based on hidden Markov models for characterizing very small channel currents buried in noise and deterministic interferences. Philosophical Transactions of the Royal Society, London B 1991:357–384.

Coleman, TP.; Brown, EN. A recursive filter algorithm for state estimation from simultaneously recorded continuous-valued, point process and binary observations. Proc. Asilomar Conference on Signals, Systems, and Computers; Piscataway, NJ: IEEE Press; 2006. p. 1949-1953.

Cowles MK, Carlin BP. Markov chain Monte Carlo convergence diagnostics: A comparative review. Journal of the American Statistical Association 1996;91:883–904.

Cox, DR.; Isham, V. Point processes. London: Chapman & Hall; 1980.

Daley, D.; Vere-Jones, D. An introduction to the theory of point processes, Vol. 1: Elementary theory and methods. New York: Springer-Verlag; 2002.

Danóczy, MG.; Hahnloser, RHR. Efficient estimation of hidden state dynamics from spike trains. In: Weiss, Y.; Schölkopf, B.; Platt, J., editors. Advances in neural information processing systems, 18. Cambridge, MA: MIT Press; 2006. p. 227-234.

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

Deng L, Mark JW. Parameter estimation for Markov modulated Poisson processes via the EM algorithm with time discretization. Telecommunication Systems 1993;1(1):321–338.

Durbin, R.; Eddy, S.; Krough, A.; Mitchison, G. Biological sequence analysis: Probabilistic models of proteins and nucleic acids. Cambridge: Cambridge University Press; 1998.

Eden, UT.; Brown, EN. Mixed observation filtering for neural data. Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'08); Piscataway, NJ: IEEE Press; 2008. p. 5201-5203.

Ephraim Y, Merhav N. Hidden Markov processes. IEEE Transactions on Information Theory 2002;48:1518–1569.

Escola, S.; Paninski, L. Hidden Markov models for the complex stimulus-response relationship of multi-state neurons. Conference abstract, Frontiers in Computational Neuroscience, Bernstein Symposium; 2008.

Forney GD. The Viterbi algorithm. Proceedings of the IEEE 1973;61(3):268–278.

Fredkin DR, Rice JA. Maximum likelihood estimation and identification directly from single channel recordings. Proceedings of the Royal Society of London, B 1992;249:125–132.

Fujisawa S, Matsuki N, Ikegaya Y. Single neurons can induce phase transitions of cortical recurrent networks with multiple internal states. Cerebral Cortex 2005;16(5):639–654. [PubMed: 16093564]

Gat I, Tishby N, Abeles M. Hidden Markov modeling of simultaneously recorded cells in the associated cortex of behaving monkeys. Networks: Computation in Neural Systems 1997;8(3):297–322.

Gelman A, Rubin DB. Inference from iterative simulation using multiple sequences. Statistical Sciences 1992;7:457–472.

Geyer CJ. Practical Markov chain Monte Carlo. Statistical Science 1992;7:473–511.

Ghahramani Z, Hinton GE. Variational learning for switching state-space models. Neural Comput 2000;12(4):831–864. [PubMed: 10770834]

Gilks, WR.; Richardson, S.; Spiegelhalter, DJ., editors. Markov chain Monte Carlo in practice. London: Chapman & Hall/CRC; 1995.

Green PJ. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. Biometrika 1995;82:711–732.

Guon Y. Estimating hidden semi-Markov chains from discrete sequences. Journal of Computational Graphical Statistics 2003;12:604–639.

Haider B, Duque A, Hasenstaub AR, McCormick DA. Neocortical network activity in vivo is generated through a dynamic balance of excitation and inhibition. Journal of Neuroscience 2006;26:4535–4545. [PubMed: 16641233]

Haider B, Duque A, Hasentaub AR, Yu Y, McCormick DA. Enhancement of visual responsiveness by spontaneous local network activity in vivo. Journal of Neurophysiology 2007;97:4186–4202. [PubMed: 17409168]

Haslinger R, Ulbert I, Moore CI, Brown EN, Devor A. Analysis of LFP phase predicts sensory response of barrel cortex. Journal of Neurophysiology 2006;96:1658–1663. [PubMed: 16775200]

Ji D, Wilson MA. Coordinated memory replay in the visual cortex and hippocampus during sleep. Nature Neuroscience 2007;10(1):100–107.

Jones LM, Fontanini A, Sadacca BF, Katz DB. Natural stimuli evoke analysis dynamic sequences of states in sensory cortical ensembles. Proc Natl Acad Sci USA 2007;104:18772–18777. [PubMed: 18000059]

Kang S, Kitano K, Fukai T. Structure of spontaneous UP and DOWN transitions self-organizing in a cortical network model. PLoS Computational Biology 2008;4(3):e1000022.10.1371/journal.pcbi. 1000022 [PubMed: 18369421]

Kass RE, Ventura V, Brown EN. Statistical issues in the analysis of neuronal data. Journal of Neurophysiology 2005;94:8–25. [PubMed: 15985692]

Kemere C, Santhanam G, Yu BM, Afshar A, Ryu SI, Meng TH, et al. Detecting neural-state transitions using hidden Markov models for motor cortical prostheses. Journal of Neurophysiology 2008;100:2441–2452. [PubMed: 18614757]

Le ND, Leroux BG, Puterman ML. Exact likelihood evaluation in a Markov mixture model for time series of seizure counts. Biometrics 1992;48:317–323. [PubMed: 1581489]

Luczak A, Barthó P, Marguet SL, Buzsáki G, Harris KD. Sequential structure of neocortical spontaneous activity in vivo. Proc Natl Acad Sci 2007;104:347–352. [PubMed: 17185420]

McCullagh, P.; Nelder, J. Generalized linear models. London: Chapman & Hall; 1989.

McLachlan, GJ.; Krishnan, T. The EM algorithm and extensions. Hoboken, NJ: Wiley; 1996.

Mehta MR. Cortico-hippocampal interaction during up-down states and memory consolidation. Nature Neuroscience 2007;10(1):13–15.

Paninski L. Maximum likelihood estimation of cascade point-process neural encoding models. Network: Computation in Neural Systems 2004;15:243–262.

Paninski, L.; Pillow, J.; Lewi, J. Statistical models for neural encoding, decoding, and optimal stimulus design. In: Cisek, P.; Drew, T.; Kalaska, J., editors. Computational neuroscience: Theoretical insights into brain function. Amsterdam: Elsevier; 2007.

Pawitan, Y. In all likelihood: Statistical modelling and inference using likelihood. New York: Oxford University Press; 2001.

Prerau MJ, Smith AC, Eden UT, Yanike M, Suzuki W, Brown EN. A mixed filter algorithm for cognitive state estimation from simultaneously recorded continuous-valued and binary measures of performance. Biological Cybernetics 2008;99(1):1–14. [PubMed: 18438683]

Rabiner LR. A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE 1989;77(2):257–286.

Radons G, Becker JD, Dülfer B, Krüger J. Analysis, classification, and coding of multielectrode spike trains with hidden Markov models. Biological Cybernetics 1994;71(4):359–373. [PubMed: 7948227]

Robert CP, Rydén T, Titterington DM. Bayesian inference in hidden Markov models through reversible jump Markov chain Monte Carlo method. Journal of the Royal Statistical Society, Series B 2000;62:57–75.

Roberts WJJ, Ephraim Y, Dieguez E. On Rydén's EM algorithm for estimating MMPPs. IEEE Signal Processing Letters 2006;13(6):373–376.

Rydén T. An EM algorithm for estimation in Markov-modulated Poisson processes. Computational Statistics and Data Analysis 1996;21:431–447.

Sanchez-Vives MV, McCormick DA. Cellular and network mechanisms of rhythmic recurrent activity in neocortex. Nature Neuroscience 2000;3:1027–1034.

Sirota A, Csicsvari J, Buhl D, Buzsáki G. Communication between neocortex and hippocampus during sleep in rodents. Proc Nat Acad Sci USA 2003;100:2065–2069. [PubMed: 12576550]

Smith AC, Brown EN. Estimating a state-space model from point process observations. Neural Comput 2003;15:965–991. [PubMed: 12803953]

Srinivasan L, Eden UT, Mitter SK, Brown EN. General-purpose filter design for neural prosthetic devices. Journal of Neurophysiology 2007;98:2456–2475. [PubMed: 17522167]

Stjernqvist S, Rydén T, Sköld M, Staaf J. Continuous-index hidden Markov modelling of array CGH copy number data. Bioinformatics 2007;23(8):1006–1014. [PubMed: 17309894]

Takagi K, Kumagai S, Matsunaga I, Kusaka Y. Application of inverse gaussian distribution to occupational exposure data. Annals of Occupational Hygiene 1997;41(5):505–514.

Tanner, MA. Tools for statistical inference. Vol. 3rd. New York: Springer-Verlag; 1996.

Truccolo W, Eden UT, Fellow M, Donoghue JD, Brown EN. A point process framework for relating neural spiking activity to spiking history, neural ensemble and covariate effects. Journal of Neurophysiology 2005;93:1074–1089. [PubMed: 15356183]

Tu Z, Zhu S. Image segmentation by data-driven Markov chain Monte Carlo. IEEE Transactions on Pattern Analysis and Machine Intelligence 2002;24(5):657–673.

Tuckwell, HC. Stochastic processes in the neurosciences. Philadelphia: SIAM; 1989.

Verbeek JJ, Vlassis N, Kröse B. Efficient greedy learning of gaussian mixture models. Neural Comput 2003;15(2):469–485. [PubMed: 12590816]

Vijayan, S. Unpublished doctoral dissertation. Harvard University; 2007. Characterization of activity in the primary somatosensory cortex during active behavior and sleep.

Viterbi J. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. IEEE Transactions on Information Theory 1967;13:260–269.

Volgushev M, Chauvette S, Mukovski M, Timofeev I. Precise long-range synchronization of activity and silence in neocortical neurons during slow-wave sleep. Journal of Neuroscience 2006;26:5665–5672. [PubMed: 16723523]

Wolansky T, Clement EA, Peters SR, Palczak MA, Dickson CT. Hippocampal slow oscillation: A novel EEG state and its coordination with ongoing neocortical activity. Journal of Neuroscience 2006;26:6213–6229. [PubMed: 16763029]

Xi J, Kass RE. Detection of bursts in neuronal spike trains using hidden semi-Markov point process models. 2008 Unpublished manuscript.

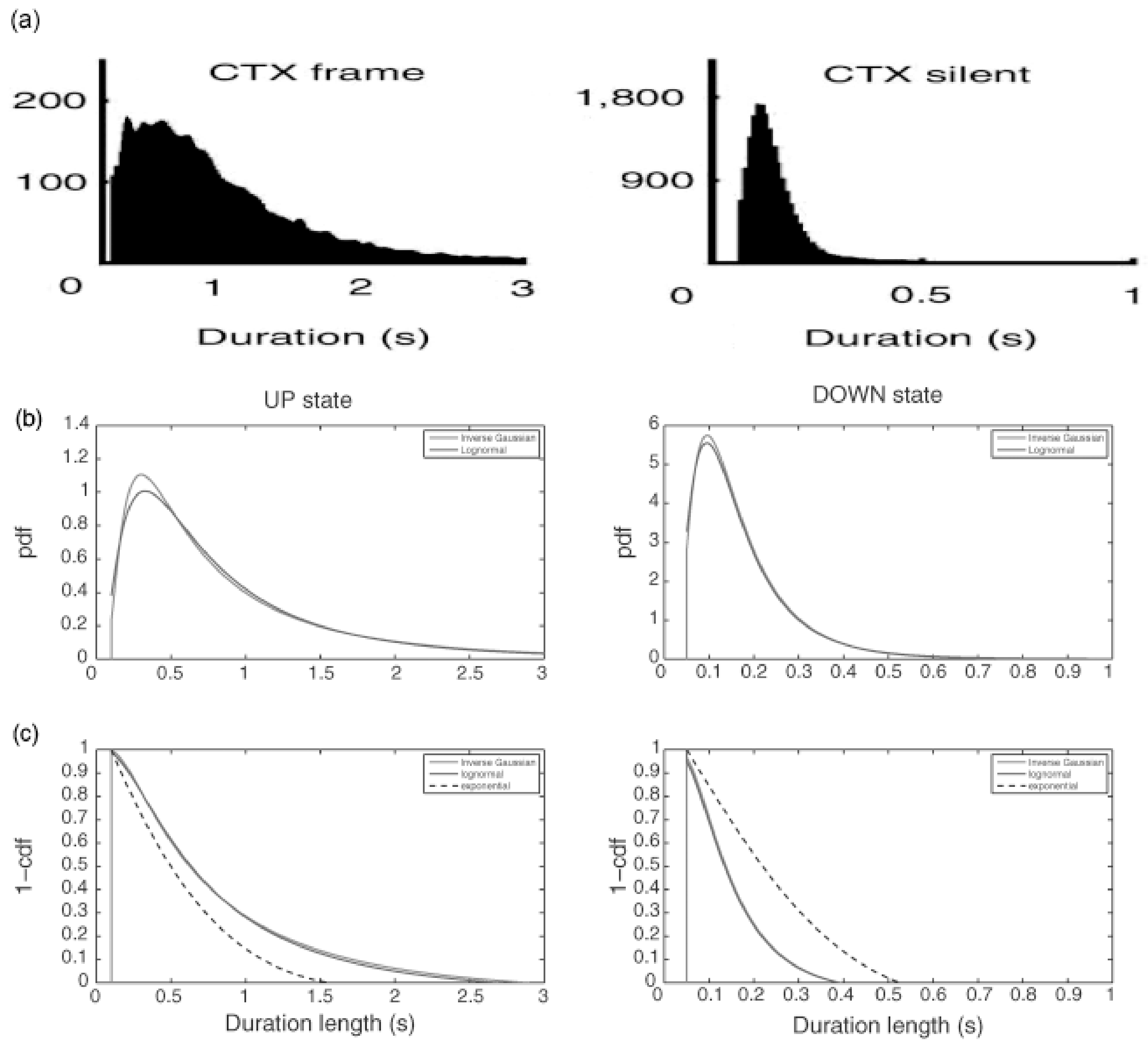

Figure 1.
(a) The duration histograms of the UP (mean 0.96, median 0.67, support [0.1, 3], unit: second) and DOWN (mean 0.17, median 0.13, support [0.05, 1], unit: second) states of spiking data recorded from four behaving rats' visual cortex during SWS. Note that the statistics used in *b* are identical to those in (Ji & Wilson, 2007, Figure 2b). The *y*-axis in both plots shows the count statistics of all cortical UP and DOWN durations. (b) Censored versions of the log-normal and inverse gaussian pdfs for the UP (left panel) and DOWN (right panel) states. (c) Censored versions of the log-normal and inverse gaussian survival functions (1-cdf) for the UP (left panel) and DOWN (right panel) states. As comparison, the dashed lines in *b* and *c* show the holding time probability for an exponential distribution. Note that in the UP state, the holding time probability in both two-parameter distributions decays more slowly than that of the exponential distribution, whereas in the DOWN state, the holding time probability of exponential distribution decays more slowly than the others.

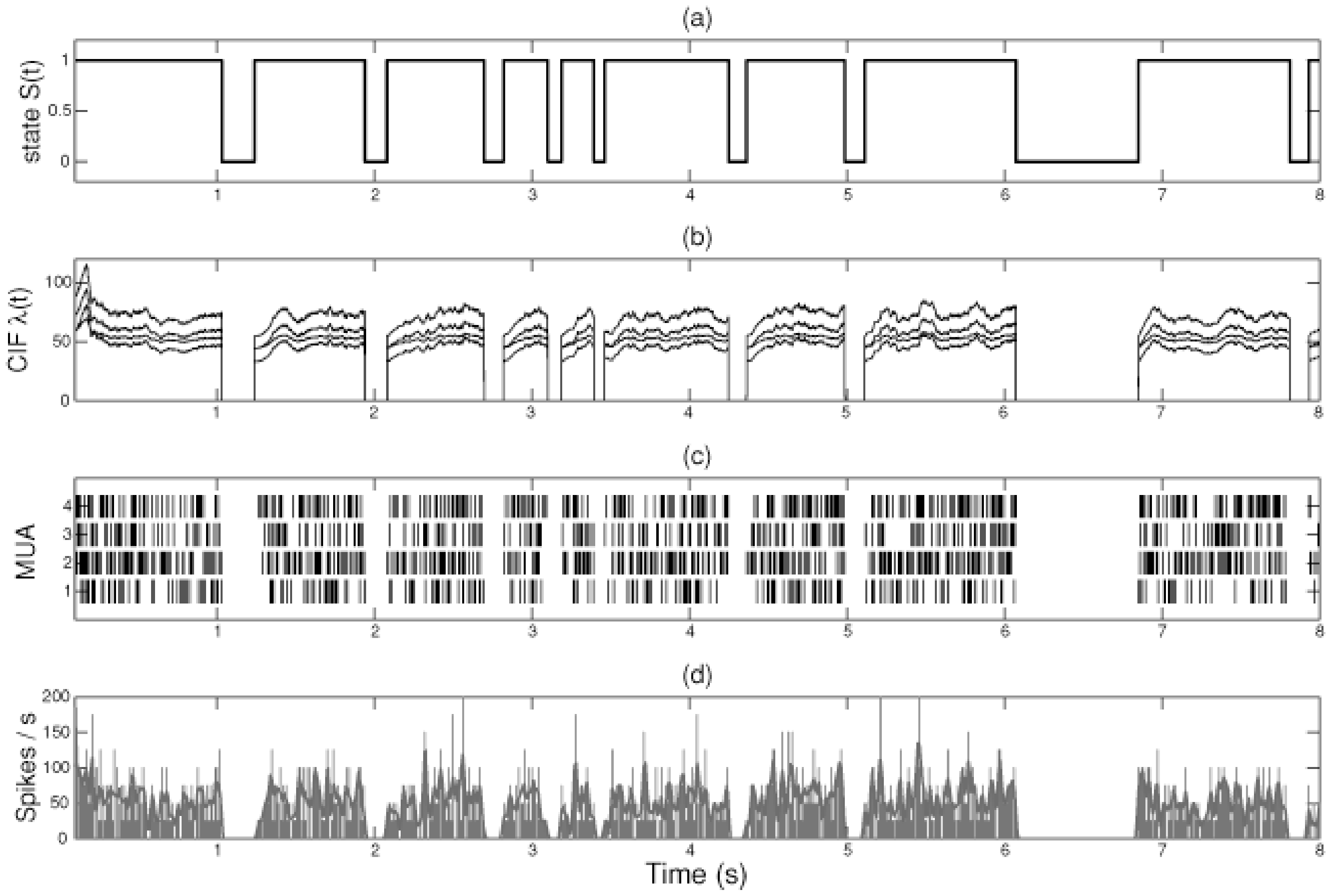

Figure 2.
Synthetic data. (a) The simulated UP and DOWN hidden state process. (b) The simulated time-varying traces of conditional intensity function (CIF) $\lambda^c(t)$ ($c = 1, \ldots, 4$). (c) The four simulated spike trains. (d) The averaged firing rate across four spike trains (the solid gray curve corresponds to the temporally smoothed firing rate using a 30 ms width gaussian kernel).

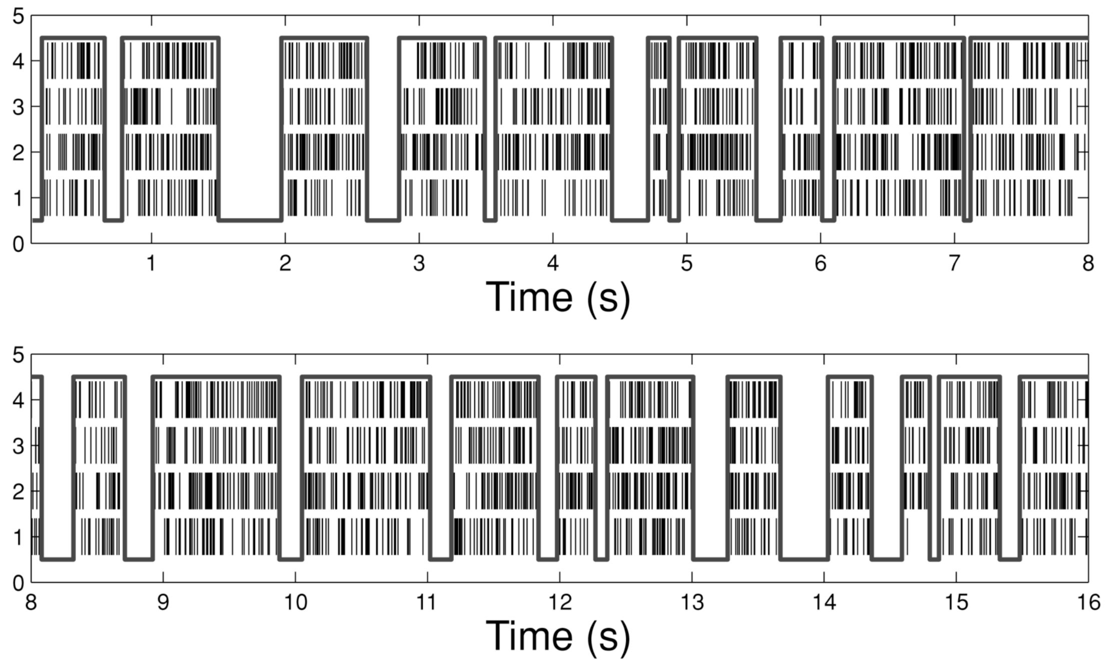

Figure 3.

A snapshot of UP and DOWN state estimation obtained from the discrete-time HMM for the simulated spike train data.

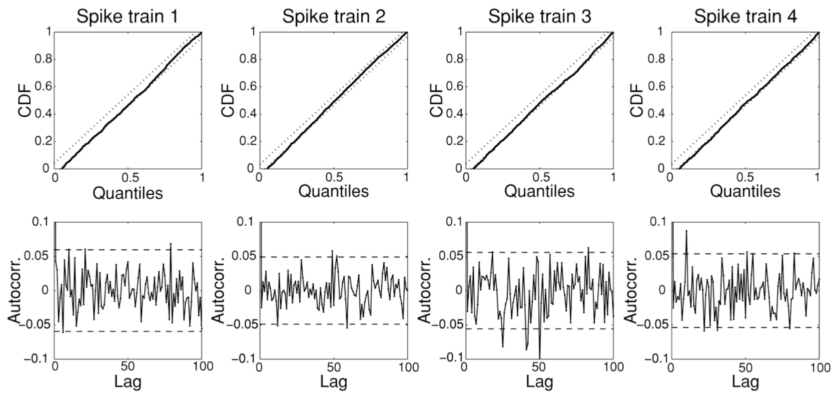

Figure 4.
The fitted KS plots (top row) and autocorrelation plots (bottom row) for the four simulated spike trains from one Monte Carlo experiment (dotted and dashed lines in the plots indicate the 95% confidence bounds).

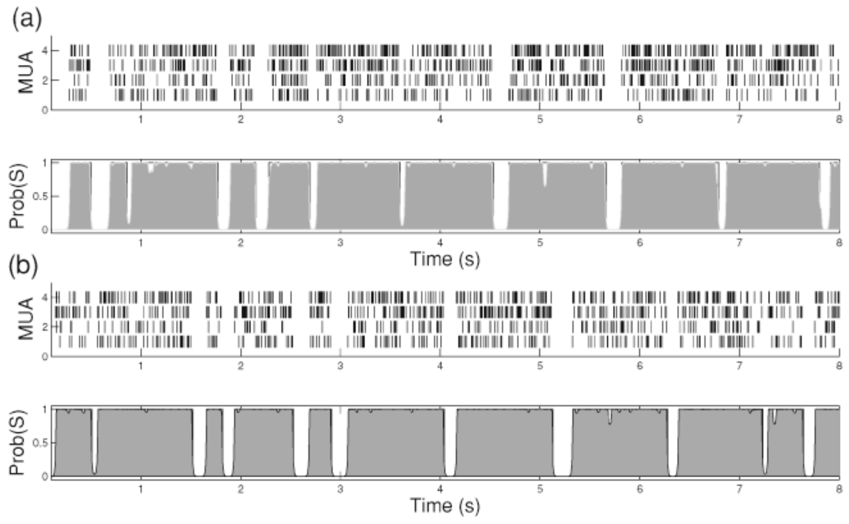

Figure 5.
Snapshot illustrations of simulated synthetic spike trains and the estimated state posterior probability from the (a) HMM and (b) continuous-time semi-Markov model (b). The shaded area denotes the posterior probability of the hidden state being in an UP state. The estimation error rates (compared with the ground truth) in these two cases are 1.9% and 1.4%, respectively.

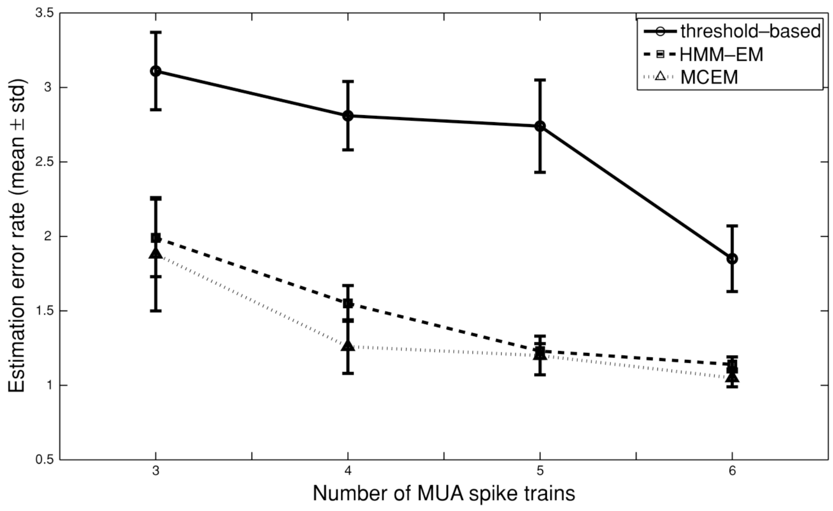

Figure 6.
The estimation error comparison of different methods by varying the number of spike train observations (the statistics are computed based on five independent simulated trials). In all conditions, the spike trains are generated using the same conditions: $\mu_c = -3.6$, $\alpha_c = 7.2$, and $\beta_c = 0.05$.

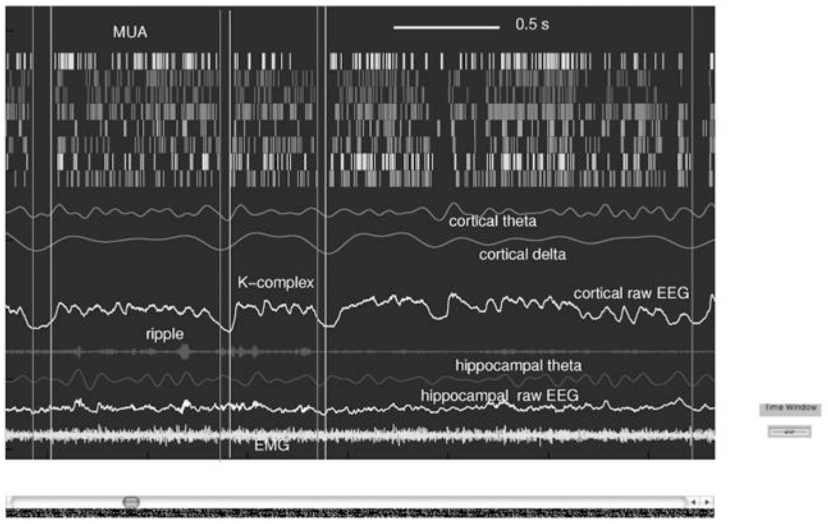

Figure 7.
A snapshot of recordings of cortical MUA, raw cortical EEG, cortical theta wave (4–8 Hz), cortical delta wave (2–4 Hz), raw hippocampal EEG, hippocampal ripple power (more than 100 Hz), hippocampal theta wave, and EMG.

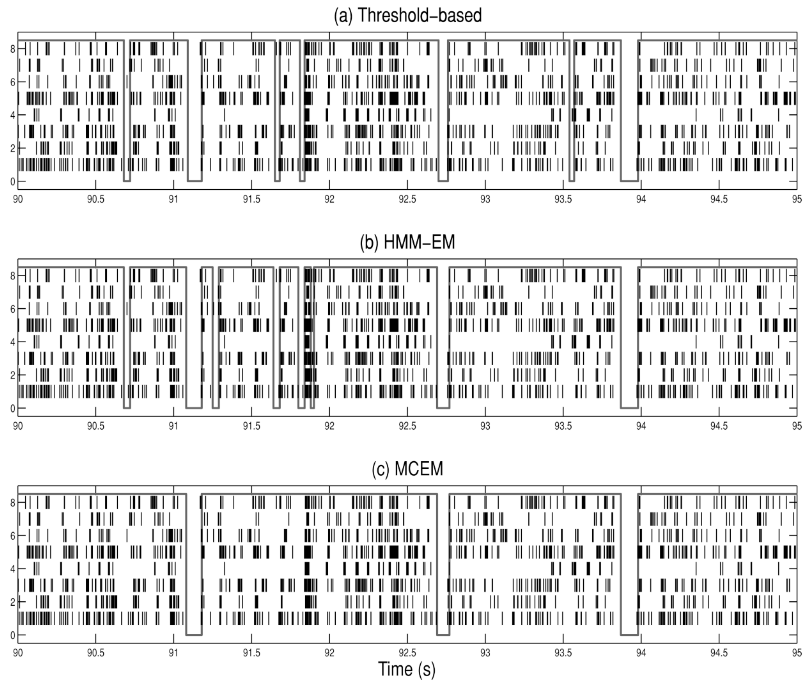

Figure 8.

Real-world MUA spike trains of eight tetrodes recorded from the primary somatosensory cortex of one rat (note that each tetrode might contain varying number of single cells). (a) A selected 5 s segment of the MUA spike trains during SWS and its UP and DOWN state classification via the threshold-based method (segmented by the thick solid line). (b) The hidden state estimation result obtained from the discrete-time HMM (used as the initial state for the continuous-time RJMCMC sampler). (c) The hidden state estimation obtained from the MCEM algorithm. In this example, the MCEM algorithm merged several neighboring sojourns that were decoded differently from the HMM.

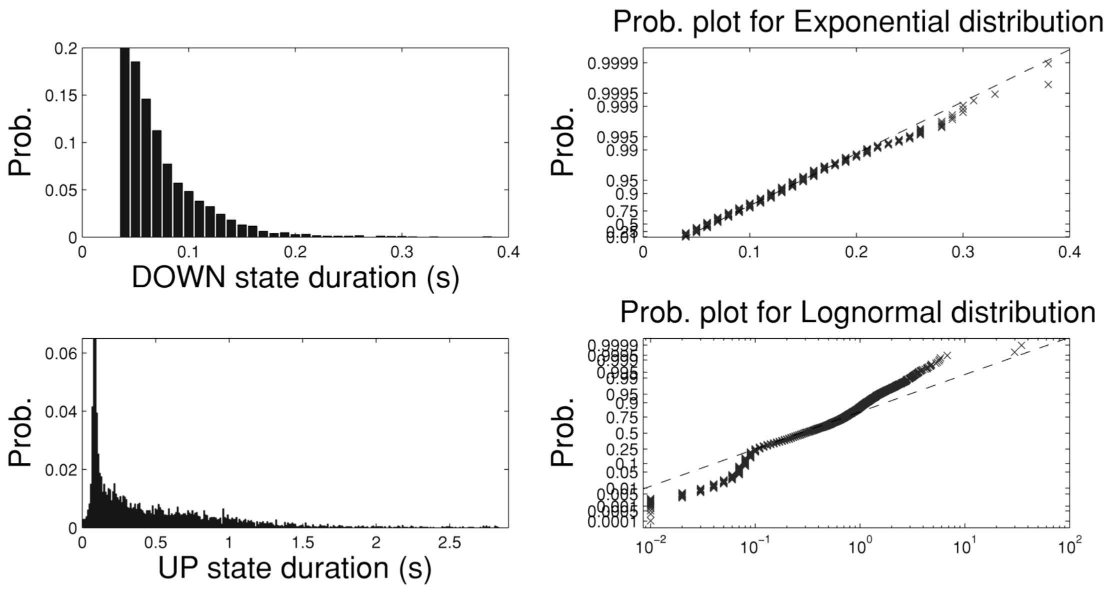

Figure 9.
Fitting the real-world sojourn-time duration length for the DOWN and UP states, where the UP or DOWN state classification is obtained from the discrete-time HMM estimation result. (Left panels) Histograms. (Right panels) Fitting the sojourn durations with exponential (for the DOWN state) and log-normal (for the UP state) distributions. If the sample data fit the tested probability density, the data points will approximately match the straight line in the plot.

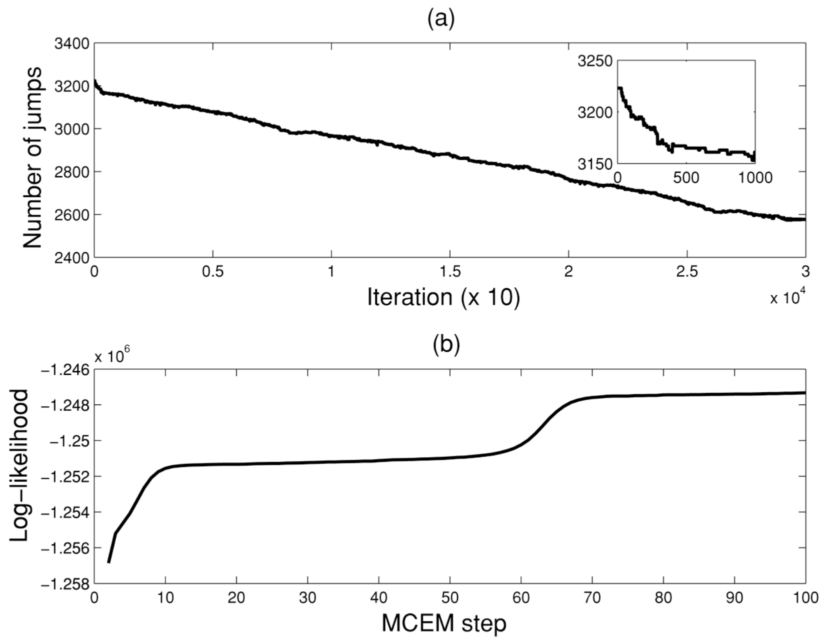

Figure 10.

Convergence plot of the simulated Markov chain. (a) Trajectory of the number of state jumps (inset: the trajectory within the first 1000 iterations). (b) Trajectory of the log likelihood in running the MCEM algorithm.

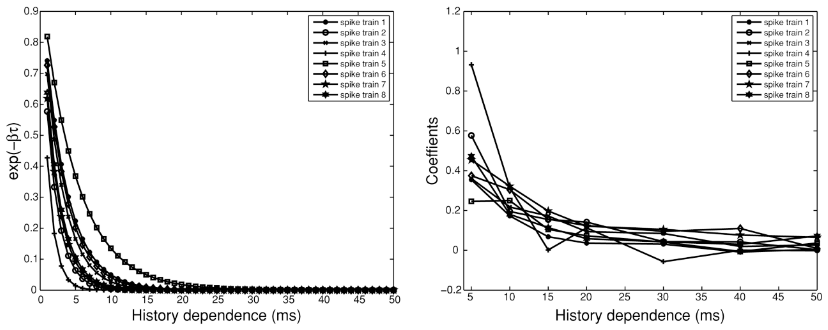

Figure 11.
(Left panel) Estimated exponential decaying filters $e^{-\beta_c \tau}$ for the recorded spike trains shown in Figure 8. (Right panel) Estimated history dependence coefficients estimated for the eight spike trains (based on GLM fit using seven discrete windows of history spike counts: 1–5, 5–10, 10–15, 15–20, 20–30, 30–40, 40–50 ms). The estimated history-dependent firing coefficients exhibit an exponential-like decaying curve (for all eight spike trains).

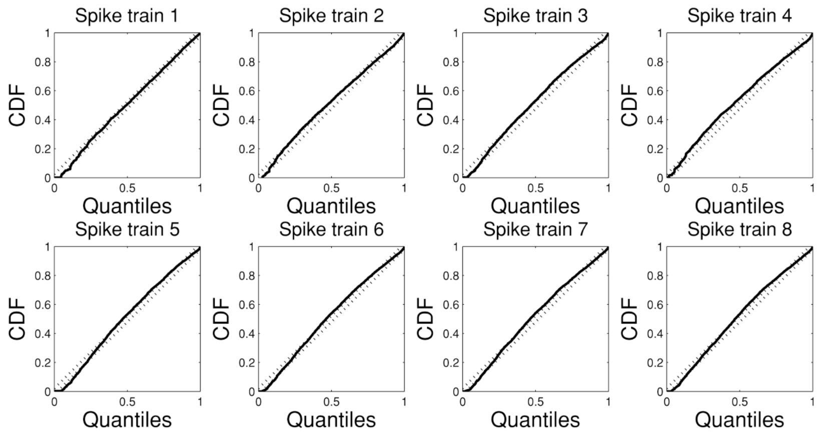

Figure 12.
Fitted KS plots of the real-world MUA spike trains (dotted lines along the diagonal indicate the 95% confidence bounds).

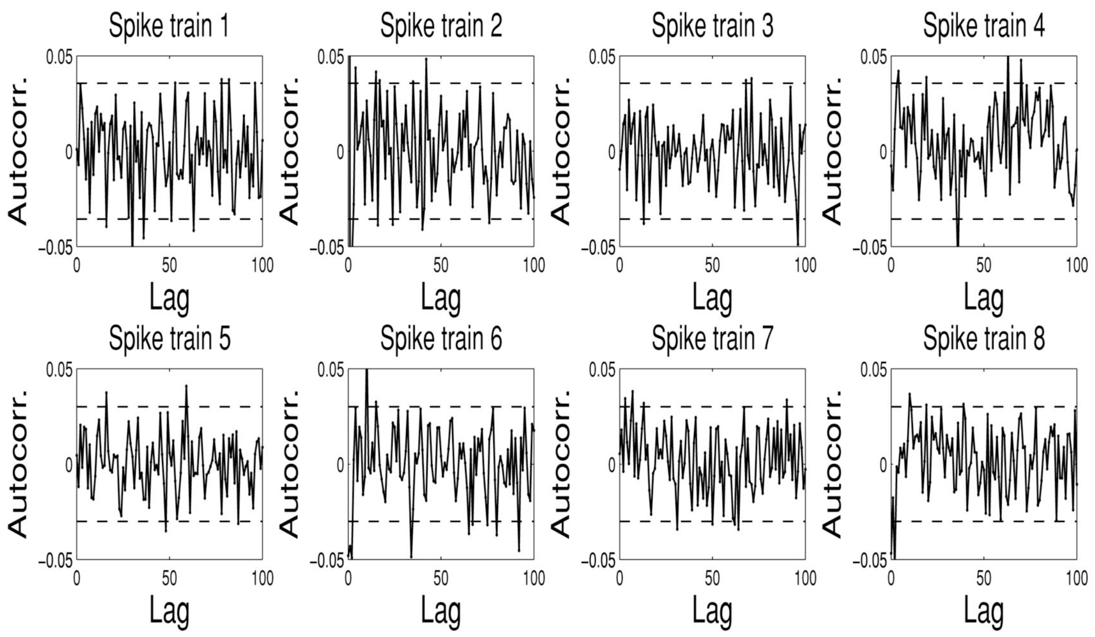

Figure 13.
Autocorrelation plots for the real-world MUA spike trains (dashed lines indicate the 95% confidence bounds).

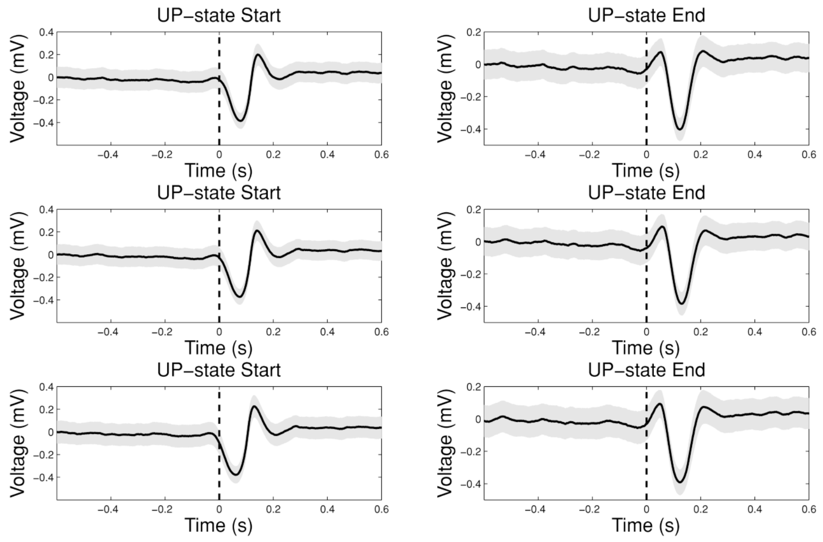

Figure 14.
Cortical EEG averages (mean ± standard error of the mean, shown by trace width) triggered by the classified UP state start and end time stamps (for visualization purposes, the standard error of the mean in all plots is amplified by 10 times its original value). From top to bottom: Results from the threshold-based method, the HMM method, and the MCEM method. The start of the UP state is aligned with the K-complex signal that has a biphasic wave switching from a negative dip to a positive peak, which lasts about 200 ms.

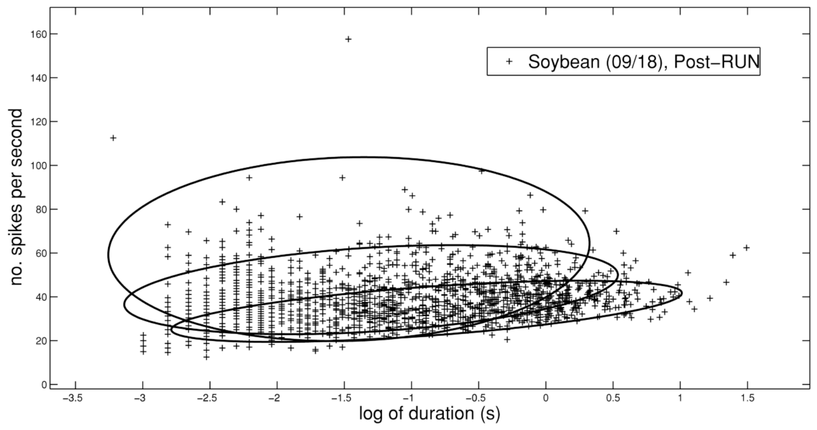

Figure 15.
Gaussian mixture clustering for the two firing features (log of duration and number of spikes per second). Here, the optimal number of mixtures is 3; the ellipses represent the two-dimensional gaussian shapes with different covariance structures.

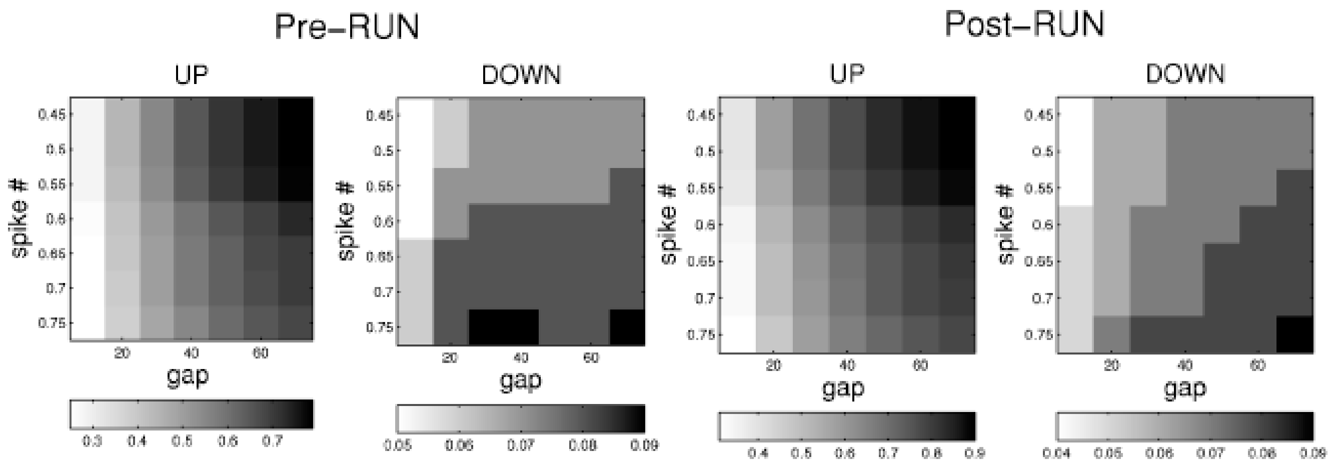

Figure 16.
Threshold-based method for estimating the median duration length of the UP and DOWN states (data from the same rat on a different day) in which two thresholds are chosen by grid search. The abscissa represents the gap threshold (in millisecond), and the ordinate represents the smoothed spike count threshold. The map's units are shown in seconds.

Table 1

Summary of Notation.

c	index of spike trains $c = 1, 2, \dots, C$
m	index of simulated Markov chains $m = 1, 2, \dots, M$
t	continuous-time index $t \in [0, T]$
t_i	spike timing of the ith spike in continuous time
Δ	smallest time bin size
k	discrete-time index $k = 1, 2, \dots, K$, $K\Delta = T$
y_k	number of counts observed from discrete-time Markov chain, $y_k \in \{0, \mathbb{N}\}$
S_k	discrete-time first-order Markov state, $S_k \in \{1, \dots, L\}$
S_0	initial Markov state at time 0
$S_{0:T}$, $S_{1:k}$	history of the Markov state from time 0 to T (or 1 to k)
n	number of state jumps within the latent process $S_{0:T}$
l	index of state jumps $l = 1, 2, \dots, n$
$\{S(t); 0 \le t \le T\}$	realization of hidden Markov process
$\mathcal{S} = (n, \boldsymbol{\tau}, \boldsymbol{\chi})$	triplet that contains all information of continuous-time Markov chain $\{S(t)\}$
$\boldsymbol{\tau} = (\tau_0, \dots, \tau_n)$	$(n + 1)$-length vector of the sojourn times of $\{S(t)\}$
$\boldsymbol{\chi} = (\chi_0, \dots, \chi_n)$	$(n + 1)$-length vector of visited states in the sojourn times of $\{S(t)\}$
$\mathcal{S}^{(0)}$	initial state of MCMC sampler
v_l	$v_0 = 0$, $v_l = \sum_{r=0}^{l-1} \tau_r$ $(l = 1, 2, \dots, n + 1)$
$\mathcal{H}_{0:T}$, $\mathcal{H}_{1:K}$	history of point-process observations from time 0 to T (or 1 to k)
$N(t)$, N_k	counting process in continuous and discrete time, $N(t), N_k \in \{0, \mathbb{N}\}$
$dN(t)$, dN_k	indicator of point-process observations, 0 or 1
P_{ij}	transition probability from state i to j for a discrete-time Markov chain, $\Sigma_j P_{ij} = 1$
q_{ij}	transition rate from state i to j for a continuous-time Markov chain, $\Sigma_j q_{ij} = 0$
$r_i = q_{ii}$	total transition rate of state i for a continuous-time Markov chain, $r_i = \Sigma_{j \neq i} q_{ij}$
π_i	initial prior probability $\Pr(S_0 = i)$
$a_k(i)$	forward message of state i at time k
$b_k(i)$	backward message of state i at time k
$\gamma_k(i)$	marginal conditional probability $\Pr(S_k = i \mid \mathcal{H}_{0:T})$
$\xi_k(i, j)$	joint conditional probability $\Pr(S_{k-1} = i, S_k = j \mid \mathcal{H}_{0:T})$
$\mathcal{L}$	log likelihood of the complete data
$R(\mathcal{S} \to \mathcal{S}')$	proposal transition density from state $\mathcal{S}$ to $\mathcal{S}'$
$\mathcal{B} = \mathcal{B}_1 \mathcal{B}_2 \mathcal{B}_3$	prior ratio × likelihood ratio × proposal probability ratio
$\mathcal{A}$	acceptance probability, $\mathcal{A} = \min(1, \mathcal{B})$
$\mathcal{J}$	Jacobian
λ_k	conditional intensity function of the point process at time k
$\boldsymbol{\theta}$	parameter vector that contains all unknown parameters
$p(x)$	probability density function

$F(x)$	cumulative distribution function, $F(x) = \int_{-\infty}^{x} p(z)\,dz$
$\Phi(x)$	gaussian cumulative distribution function
$\mathrm{erf}(x)$	error function
$\mathbb{I}(\cdot)$	indicator function
$\mathcal{U}(a, b)$	uniform distribution within the region (a, b)

Table 2

Summary of Continuous-Time Probability Models for the Transition Probability Density Function $p(\tau)$ (Where τ Is a Nonnegative or Positive Random Variable That Denotes the Holding Time), Cumulative Distribution Function $F(\tau)$, and Survival Function $1 - F(\tau)$.

	pdf $p(\tau)$	**cdf $F(\tau)$**	**l– $F(\tau)$**	**Domain**
Exponential	$r\exp(-r\tau)$	$1-\exp(-r\tau)$	$\exp(-r\tau)$	$\tau\in[0,\infty), r>0$
Weibull	$r\alpha(r\tau)^{\alpha-1}\exp[-(r\tau)^{\alpha}]$	$1-\exp[-(r\tau)^{\alpha}]$	$\exp[-(r\tau)^{\alpha}]$	$\tau\in[0,\infty), r>0, \alpha>0$
Gamma	$\tau^{s-1}\frac{\exp(-\tau/\theta)}{\Gamma(s)\theta^{s}}$	$\frac{\gamma(s,\tau/\theta)}{\Gamma(s)}$	$1-\frac{\gamma(s,\tau/\theta)}{\Gamma(s)}$	$\tau\in[0,\infty), s>0, \theta>0$
Log normal	$\frac{1}{\tau\sqrt{2\pi\sigma^{2}}}\exp\left(-\frac{(\ln\tau-\mu)^{2}}{2\sigma^{2}}\right)$	$\frac{1}{2}+\frac{1}{2}\mathrm{erf}\left[\frac{\ln\tau-\mu}{\sqrt{2}\sigma}\right]$	$\frac{1}{2}-\frac{1}{2}\mathrm{erf}\left[\frac{\ln\tau-\mu}{\sqrt{2}\sigma}\right]$	$\tau\in(0,\infty), \mu>0, \sigma>0$
Inverse gaussian	$\sqrt{\frac{s}{2\pi\tau^{3}}}\exp\left(-\frac{s(\tau-\mu)^{2}}{2\mu^{2}\tau}\right)$	$\Phi\left(\sqrt{\frac{s}{\tau}}\left(\frac{t}{\mu}-1\right)\right)+\exp\left(\frac{2s}{\tau}\right)\Phi\left(-\sqrt{\frac{s}{\tau}}\left(\frac{t}{\mu}+1\right)\right)$	–	$\tau\in(0,\infty), \mu>0, s>0$

Note: $\mathrm{erf}(z)=\frac{2}{\sqrt{\pi}}\int_{0}^{z}\exp(-t^{2})\,dt$ denotes the error function, $\Phi(z;\mu,\sigma)=\int_{-\infty}^{z}\exp\left(-\frac{(t-\mu)^{2}}{2\sigma^{2}}\right)dt$ denotes the gaussian cumulative distribution function, and these two functions relate to each other by $\Phi(z)=\frac{1}{2}\left[1+\mathrm{erf}\left(\frac{z}{\sqrt{2}}\right)\right]$.

Table 3

Comparison of Experimental Results on the Simulation Data.

	State Estimation Error Rate		
Method	**Mean ± SD**	**Best**	**Worst**
Threshold based	2.91 ± 0.31%	2.41%	3.48%
Discrete HMM-EM	1.52 ± 0.34	1.07	2.07
Continuous MCEM	1.26 ± 0.42	0.74	1.95

Note: Mean performance is averaged over 10 independent random trials.

Table 4

Duration Length Statistics of the UP and DOWN States from the Simulation Data.

	True	**Sample Statistics (from HMM)**	**Estimated (from MCEM)**
Mean (UP)	−0.4005	−0.4468	−0.4212
SD (UP)	0.8481	0.6827	0.7735
Mean (DOWN)	−1.9661	−2.1708	−2.0256
SD (DOWN)	0.6231	0.6335	0.6301

Table 5

State Estimation Discrepancy Between the Proposed Algorithms and the Threshold-Based Method for the Real-World Spike Trains Data.

Algorithm	Discrepancy Percentage	Number of Jumps, *n*	Bin Size
Threshold-based	—	2986	10 ms
Discrete HMM-EM	4.42%	3223	10 ms
Continuous MCEM	3.04%	2576	1 ms

Table 6

Estimated Statistics of the UP and DOWN State Durations (in msec) for the Real-World Spike Train Data.

	UP State			DOWN State		
Sojourn Duration Length	**Threshold Based**	**HMM-EM**	**MCEM**	**Threshold Based**	**HMM-EM**	**MCEM**
Minimum	40	40	58	40	20	53
Maximum	8510	4450	5059	270	290	302
Median	510	330	446	60	60	77
Mean ± SD	739 ± 727	507 ± 493	644 ± 638	67 ±31	74 ± 36	83 ± 37