# Estimating Conditional Intensity Function of a Neural Spike Train by Particle Markov Chain Monte Carlo and Smoothing

by

Haixu (Alex) Wang

B.Sc., Simon Fraser University, 2015

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
Department of Statistics and Actuarial Science
Faculty of Science


SIMON FRASER UNIVERSITY
Summer 2017

# Approval

| | |
|---|---|
| **Name:** | Haixu (Alex) Wang |
| **Degree:** | **Master of Science (Statistics)** |
| **Title:** | **Estimating Conditional Intensity Function of a Neural Spike Train by Particle Markov Chain Monte Carlo and Smoothing** |
| **Examining Committee:** | **Chair:** Tim Swartz<br>Professor |
| | **Jiguo Cao**<br>Senior Supervisor<br>Professor |
| | **Qian (Michelle) Zhou**<br>Supervisor<br>Associate Professor |
| | **Liangliang Wang**<br>Internal Examiner<br>Associate Professor |
| **Date Defended:** | **August 14, 2017** |

# Abstract

Understanding neural activities is fundamental and challenging in decoding how the brain processes information. An essential part of the problem is to define a meaningful and quantitative characterization of neural activities when they are represented by a sequence of action potentials or a neural spike train. The thesis approaches to use a point process to represent a neural spike train, and such representation provides a conditional intensity function (CIF) to describe neural activities. The estimation procedure for CIF, including particle Markov Chain Monte Carlo (PMCMC) and smoothing, is introduced and applied to a real data set. From the CIF and its derivative of a neural spike train, we can successfully observe adaption behavior. Simulation study verifies that the estimation procedure provides reliable estimate of CIF. This framework provides a definite quantification of neural activities and facilitates further investigation of understanding the brain from neurological perspective.

**Keywords:** Neural data analysis; neural spike train; point process; state-space model; particle Markov Chain Monte Carol; smoothing

# Acknowledgements

I would like to thank my parents and wife for unlimited support during the past 2 years.

I would also like to thank my supervisors Jiguo and Michelle for their guidance.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

A definite and quantitative characterization of neural activities provides foundations to recognizing how neurons process information received from external stimulations. It can also help us to understand how signals are being transmitted within the nervous system. One fundamental assumption of neural data analysis is that information is represented by the sequence of action potentials or spikes generated by neuron. Hence, we need to decode the sequence for understanding the language that neurons use for transmitting and processing information. Rate coding scheme is the most popular way to describe neuron behaviours, but there are several different definitions for the rate code (**Gerstner**). First, the rate code can be referred to as a spike count rate which corresponds to the average number of spikes occurred within an experimental period. The use of spike count rate is limited to the case when neuron's response is stationary across the experiment. However, that case is ideal and only reason. The spike count rate is not desirable in current researches, since we are interested in how neurons respond to complex and dynamic stimulus. Another rate code is calculated by averaging over repeated experiments. Neural response (sequence of action potentials) from several trials are aligned in parallel to form a raster plot. Define time bins with size of $\Delta$, the firing rate at each time bin of a neuron is calculated by diving the number of spikes from all trials by the number of trials during that time bin. This rate code can be summarized by a Peri-Stimulus-Time Histogram (PSTH). This coding scheme is easy to implement but ignores trial-to-trial variability. In addition, it cannot be applied to case when stimulation or experiment is unrepeatable. As a result, we need a reliable rate coding scheme utilizing probabilistic approach with single trial response.

An intuitive probability model for neural spike trains is point process model. A point process is a random process containing a sequence of binary events defined in continuous time (**Daley**), and

it has a conditional intensity function which can be considered as a rate function for quantifying observed neural spikes. When stimulus is not quantifiable or explicit, examining how neurons react and adapt to the stimulation can be done using the state-space model from signal processing framework (**SmithandBrown**). A state-space model contains two components: a latent process governing the system and an observation equation representing the output of the system. In this case, we assume that there exists a latent process modulating how a neuron generates spikes, and neuron's spike train corresponds to the consequence or output. Smith and Brown (2003) use approximate Expectation-Maximization algorithm to estimate parameters in the point process observations and latent states. **Yuan11235** suggest to employ a more powerful class of estimation method which is Markov Chain Monte Carlo (MCMC). **PMCMC** proposes a new variant of MCMC methods called the particle Markov Chain Monte Carlo (PMCMC), and it is particularly suitable for inference about state-space models.

The purpose of this work is to investigate how wide dynamic range (WDR) neurons respond to the pain stimulation. Neural spike trains are recorded for WDR neurons under pain stimulation, and we wish to calculate the conditional intensity function if we consider spike trains as point process observations. In addition, we assume that the conditional intensity, uniquely defining a point process, is governed by a latent process through a log-linear model, and the focus is to estimate the latent process and parameters of the log-linear model. The rest of the thesis is arranged as follows. Chapter 2 provides a literature review about works contributing to neural data analysis. Chapter 3 presents the methodology of PMMH used for parameter estimation and state inference given neural spike train data. A smoothing procedure is also introduced in the section for obtaining better estimation of intensity given PMMH sampling result. Moreover, the derivative of intensity is immediately available after smoothing. Chapter 4 focuses on the application of PMMH sampler and smoothing technique for a real data example. Chapter 5 is dedicated for simulation study'' to verify the variability and reliability of the estimation procedure. Finally, a conclusion and discussion is provided in section 6.

# Chapter 2

# Literature review

**Hubel1988** provides a well-organized and appealing introduction to how we can understand the brain from a neurological perspective with the example of visual system. He mentions that large parts of the brain are still mysteries to solve despite the knowledge we have about some regions like visual or motor cortex. He states that the main part of the brain is nerve cells, or neurons. Neurons are responsible for exchanging information by their chemical transmission. In order to understand the functionality of the brain, it may be better to start from studying how single neuron behaves in transmitting information. There are three components of a neuron: the cell body, dendrites for receiving information , and axon for delivering information. They are all enclosed by the cell membrane to build up the entire neuron. The form of information understood and transferred by neurons are impulses. As Hubel points out:

> In the first cell, an electrical signal, or impulse, is initiated on the part of an axon closest to the cell body. The impulse travels down the axon to its terminals. At each terminal, as a result of the impulse, a chemical is release into the narrow, fluid-filled gap between one cell and the next-the *synaptic cleft*-and diffuses across this 0.02-micrometer gap to the second, postsynaptic, cell. There it affects the membrane of the second cell in such a way as to make the second cell either more or less likely to fire impulses. (p.30)

First, we have to define a measurement quantifying neural activities, and the most common one is membrane potential indicating the difference in electrical potential between the inside and the outside of a neuron. Hubel gives a brief explanation about the importance and reason of membrane potential. He describes the membrane of a neuron as not a continuous and closed surrounding but a structure with many passages, pumps, or channels. In addition, neurons are filled with and embedded

in salt water enriched with charged ions, and channels are only responsible for allowing one specific kind of ions to flow in and out. Then, the membrane potential is determined by the concentration of positively charged ions inside of a neuron.

To understand the structure and functionality of neurons, we must recognize the series of 5 papers by Alan Hodgkin and Andrew Huxley in 1952. Their works, including one paper collaborated Bernard Katz, provide fundamental concepts with a quantitative model in neurophysiology. Neural impulses are also defined as spikes or action potentials in their works. **HH1952** provide evidence that initiation of an action potential depends three current components: sodium ions, potassium ions, and a combination of chloride and other ions. Even before their works in 1952, **HH1939** present the first intracellular recording of an action potential from a giant squid axon.

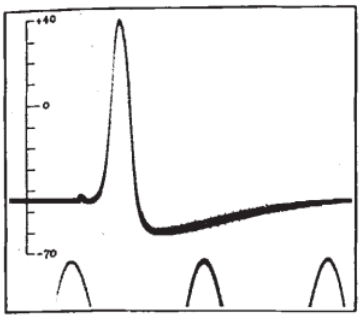


Figure 2.1: The first recording of an action potential. Reprinted from "*Action potentials recorded from inside a nerve fibre*", by A. Hodgkin and A. Huxley, 1939, *Nature*, *144*, 710-711. Reprinted with permission

In Hubel (1988), we can find a short but complete description of the generation an action potential from ionic movements.

> When the nerve is at rest, most but not all potassium channels are open, and most sodium channels are closed; the charge is consequently positive outside. During an impulse, a large number of sodium pores in a short length of the nerve fiber suddenly open, so that briefly the sodium ions dominate and that part of the nerve suddenly becomes negative outside, relative to inside. The sodium pores then reclose, and meanwhile even more potassium pores have opened than are open in the resting state. Both events-the sodium pores reclosing and additional potassium pores opening- lead to the rapid restoration of the positive-outside resting state. The whole sequence lasts about one-thousandth of

a second. All this depends on the circumstances that influence pores to open and close. (p. 16)

Knowledge of action potentials is fundamental for understanding the brain since it offers a way to define neural and brain activities. However, information perceived from the outside world to our brain is not reflected in each action potential but translated by a sequence of action potentials or spikes. The sequence of neural spikes is sometimes referred to as a neural spike train. If we assume that information received by sensory neurons are transformed into neural spike trains, understanding the brain can be accomplished by identifying the correspondence between information and neural spikes. **Rieke1999** focuses his attention on the problem of "what can the spike train of this one neuron tell us about events in the world". He claims that decoding the sequence of binary events can help us to recognize the functions of neurons, groups of neurons, and eventually the brain. The second chapter of his book provides quantitative foundations to describe and characterize neural spike trains.

Rieke investigates rate coding schemes to characterize neural responses. He starts from a rate code calculated from averaging number of spikes given a window of time from different trials. The rate function gives a probability of observing a spike in a small window. However, this coding scheme relies on the assumption that temporal information of spikes is not taken into account for decoding neural responses. The assumption of this rate code is contradictory to its calculation and result since it cannot provide a time-dependent rate function without using temporal information of spikes. In addition, he presents several experiments "suggesting that the elements of neural signal transmission and computation are capable of preserving precise temporal relationships." (**Rieke1999** p.36) He then proceeds to use a Poisson model to describe activities and statistics of a single neuron. The choice of a Poisson model adopts the idea of rate code, that is, Poisson model has its rate function tracks the probability of spike occurrence per unit time. Furthermore, Poisson models preserve temporal information of spikes for calculating statistics of spike train. Poisson model has a chance of working but is often expected to underperform in real spike trains, and Rieke has stated that:

At the very least we know that spikes cannot come too close together in time, because the spike generating mechanism of all cells is *refractory* for some short time following

the firing of an action potential. Thus the occurrence time of one spike cannot be completely independent of all the other occurrence times, as assumed in a Poisson model. When Poisson models give a good approximation to the data, it just means that the refractory time scale-or, more generally, any memory time scale in the spike generating mechanism-is short compared to the interesting time scales such as the mean interspike interval. (p. 53)

Poisson model provides a good basis for characterizing neural spike trains, hence, we can use a more general version of Poisson processes to relax the independence assumption of occurrences of spikes. Poisson process is the simplest example of a point process. A point process is used to represent a sequence of discrete events defined in continuous time. In this case, discrete events become binary events which indicating whether a spike occurs at a certain time. There are four equivalent specifications of a point process for a spike train: (1) spike times, (2) waiting times (interspike times), (3) counting function, and (4) sequence of binary indicators.

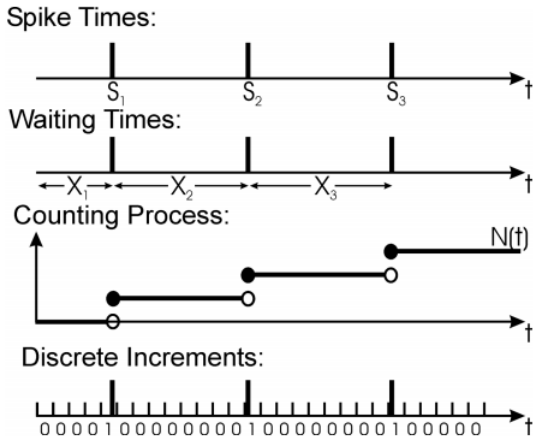


Figure 2.2: Four equivalent representations of a point process. Reprinted from "*Analysis of Neural Data (p.536)*", by Robert E. Kass, Uri Eden, and Emery N. Brown, 2014, New York: Springer-Verlag. Copyright 2014 by Springer Science+Business Media New York. Reprinted with permission

**Kass2014** serves as a thorough guide for understanding, interpreting, and analyzing neural data, and the book includes one chapter covers how to decode neural spike trains under point process framework. They suggests that the general point process improves the simple Poisson model by considering the history dependence in neural spike trains. The history dependence is included in

the conditional intensity function of a point process which generalizes the intensity functions of homogeneous and non-homogeneous Poisson processes. The conditional intensity function $\lambda(t|H(t))$ is defined as follows

$$\lambda(t|H(t)) = \lim_{\Delta \to 0} \frac{P(N(t+\Delta) - N(t) = 1|H(t))}{\Delta} \tag{2.1}$$

where $H(t)$ contains information of previous spikes up to time t. The conditional intensity function is an essential component for constructing the joint probability distribution or likelihood function for a neural spike train. Furthermore, Daley and Vere-Jones (2003) claims that "the conditional intensity function determines the probability structure of the point process uniquely." (p. 233). In that case, we can build a model for the conditional intensity in order to characterize a neural spike train.

Two approaches for modelling the conditional intensity function of a single neuron are generalized linear model (GLM) and signal processing method. **truccolo2005** proposes a method combining point process and GLM framework for decoding neural responses which is *point process-GLM framework*. First, they discuss three components to consider in modelling spiking activities: extrinsic stimulus applied to sensory neurons, past activities of a neuron, and interaction between neurons. An ideal statistical model should be able to evaluate how these three effects influence the spiking activities of a neuron. The paper proceeds to define the likelihood function of a neural spike train which corresponds to the joint probability of the sequence of binary events. The likelihood can also be expressed as a function the conditional intensity. Modelling the conditional intensity suffices to characterize the neural spike train, then we can use a log link function to establish a model for intensity based on history effect, extrinsic stimulus effect, and influence from other neurons. Eventually, the likelihood function of intensity is the same as a likelihood function for a GLM with a log link function. As a result, we can use maximum likelihood estimation for obtaining parameter estimates in the intensity model.

Another approach to study neural spike trains is to use signal processing method or filtering. It usually involves using a state-space model as explained in Smith and Brown (2003). The motivation of signal processing method is that the stimulus applied to experimental units are not always measurable or explicit. When stimulus is implicit, the previous point process-GLM framework cannot be applied

since we cannot examine how neural activities depend on the stimulus quantitatively. On the other hand, a state-space models can model neural activities with implicit stimulus since it can include stimulus effect into an arbitrary latent process. A state-space model can be decomposited into two parts: an observable process and a latent process assumed to modulate the evolution of observations. In their paper, the latent process is represented by a Gaussian autoregressive model driven by external stimulus. The observation process is a general point process characterized by its conditional intensity function. Given the state-space mode, estimations will be focused on latent process with its parameters and parameters of the point process. For that matter, they use approximate expectation-maximization (EM) method to obtain estimates.

Subsequently, Yuan, Girolami, and Niranjan (2012) suggest to use a more flexible and powerful class of method for inference about state-space model with point process observations (SSPP). The model description in Yuan, Girolami, and Niranjan (2012) is the same as that in Smith and Bronw (2003), and they replace approximate EM method with Markov Chain Monte Carlo (MCMC) methods. Two MCMC algorithms introduced in their paper are particle marginal Metropolis-Hasting (PMMH) by Andrieu, Doucet and Holenstein (2010) and the Riemann manifold Hamiltonian Monte Carlo (RMHMC) in **RMHMC** PMMH relies on a particle filtering or sequential Monte Carlo (SMC) method to obtain estimates of latent process and likelihood conditioned on a parameter candidate. A comprehensive guide to SMC is provided by **guildtoSMC**

Due to the fact that the latent process is assumed to follow a first order autoregressive model, resulting estimate of latent process or intensity function is often too 'wiggly' even with a constant stimulus effect. In that case, it does not obey the assumption that neural response should be moderate if there is no significant change in stimulation. To comply this assumption, the following methodology propose a smoothing step as introduced in **fdaRamsay** in addition to the previous work.

# Chapter 3

# Methodology

## 3.1 Model Description

Let $[0, T]$ be an observation interval which contains J spikes from a single neuron. We define $\{s_j\}_{j=1}^{J}$ to be the set of spike times such that $0 \leq s_1 \leq s_2 \leq ... \leq s_J \leq T$. Let $N(t)$ be a counting function which tracks the number of spikes that have been observed up to and including the current time $t$, and it contains all the information of the point process observations in the interval $[0, t]$. Then, a neural spike train can be denoted as $N_{0:T} = \{0 \leq s_1 \leq ... \leq s_J \leq T \ \cap N(T) = J\}$. Let $H(t)$ be the history information including spiking history $N_{0:t}$ or the effect of extrinsic stimulus applied to the neuron during time interval $[0, t]$, any point process can be completely characterized by its conditional intensity function (**Daley**) defined as

$$\lambda(t|H(t)) = \lim_{\Delta \to 0} \frac{P(N(t+\Delta) - N(t) = 1|H(t))}{\Delta}. \tag{3.1}$$

It follows that $\lambda(t|H(t))\Delta$ gives an approximate of the probability for observing a spike in the time interval $[t, t + \Delta)$ for small $\Delta$.

In addition, we assume that there exists a latent process $x(t)$ modulating the observed neural activities. Both processes can be defined in continuous time, but a discrete representation provides a simpler notation and facilitates particle filtering algorithms. We partition the observation interval $[0, T]$ into K subintervals with length of $\Delta = \frac{T}{K}$ given time points $\{t_k\}_{k=0}^{K}$. $\Delta$ or K is chosen to ensure that there is at most one spike in each subinterval $(t_{k-1}, t_k]$. Under discrete representation, $N(t) = N_k$, $N_{0:t} = N_{1:k}$, $\lambda(t) = \lambda_k$, and $x(t) = x_k$ at time $t \in (t_{k-1}, t_k]$. Let $\boldsymbol{\theta}$ be the vector

containing static parameters, the likelihood of a neural spike train is the joint probability density of observing J spikes in the time interval $[0, T]$ denoted as:

$$P(N_{0:T}|\boldsymbol{\theta}) = \prod_{j=1}^{J} \lambda(s_j|H(s_j), \boldsymbol{\theta}) \exp\left\{-\int_0^T \lambda(u|H(u), \boldsymbol{\theta}) du\right\}. \tag{3.2}$$

In discrete time, the above likelihood can be approximated by the following likelihood function

$$L(\boldsymbol{\theta}) = P(N_{1:k}|\boldsymbol{\theta}) = \exp\left\{\sum_{k=1}^{K} \log(\lambda_k \Delta) \mathrm{d}N_k - \sum_{k=1}^{K} \lambda_k \Delta\right\} \tag{3.3}$$

where $\mathrm{d}N_k = N_k - N_{k-1}$ representing whether a spike has occurred in the k-th subinterval.

Let the dependence between latent process and observations be defined through the following CIF model

$$\lambda_k = \exp(\mu + \beta x_k). \tag{3.4}$$

where $x_k$ represents latent states in discrete time bins. The latent states are assumed to follow an AR(1) model, that is,

$$x_k = \phi x_{k-1} + \epsilon_k \tag{3.5}$$

where $\epsilon_k$ represents a Gaussian noise with mean of 0 and variance of $\sigma^2$. $\beta$ and $\sigma$ are fixed as constants for identifiability of the model, hence the static parameter vector is $\boldsymbol{\theta} = (\mu, \phi)$. The latent states can be characterized by (1)initial density $f_{\boldsymbol{\theta}}(X_1)$, (2) transition density $f_{\boldsymbol{\theta}}(X_k|X_{k-1})$, and (3) conditionally independent observations $N_k$ with density $g_{\boldsymbol{\theta}}(N_k|X_{1:k})$ conditioned on the latent states. Joining the latent process with point process observations, we are able to utilize the state-space model framework for which has a graphic illustration as follows:

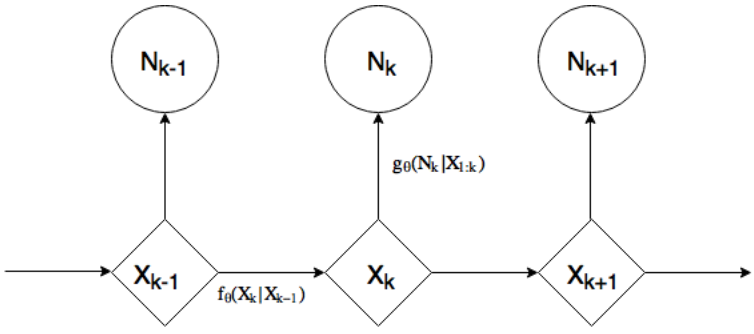


Figure 3.1: State-space model: Observation and latent process

Bayesian inference for state-space model is focused on exploring the posterior distributions of latent states

$$P(X_{1:K}|N_{1:K}, \boldsymbol{\theta}) \tag{3.6}$$

where $\boldsymbol{\theta}$ is known. When $\boldsymbol{\theta}$ is not available , inference relies on the the joint posterior distribution of $X_{1:k}$ and $\boldsymbol{\theta}$

$$P(X_{1:K}, \boldsymbol{\theta}|N_{1:K}) \propto P(X_{1:K}|N_{1:K}, \boldsymbol{\theta})P(\boldsymbol{\theta}) \tag{3.7}$$

given a prior distribution of $\boldsymbol{\theta}$ which is $P(\boldsymbol{\theta})$. For applications to real data, above posterior distributions do not possess a closed form when the state model does not have a Gaussian noise. Hence, only simulation-based approaches are applicable for approximating the posterior distributions.

## 3.2 Particle filtering

First, we consider the case where the static parameters are available. Based on Bayes' theorem, the posterior distribution $p(x_{1:k}|N_{1:k}, \boldsymbol{\theta})$ can be derived from

$$p(x_{1:k}|N_{1:k}, \boldsymbol{\theta}) = \frac{p(x_{1:k}, N_{1:k}|\boldsymbol{\theta})}{p(N_{1:k}|\boldsymbol{\theta})} \tag{3.8}$$

where the joint density of observed and hidden process is

$$p(x_{1:k}, N_{1:k}|\boldsymbol{\theta}) = f_{\boldsymbol{\theta}}(x_1) \prod_{j=1}^{k} f_{\boldsymbol{\theta}}(X_j|X_{j-1}) \prod_{i=1}^{k} g_{\boldsymbol{\theta}}(N_i|X_{1:i}) \tag{3.9}$$

It follows that the joint density can be written in a recursive form

$$p(x_{1:k}|N_{1:k}, \boldsymbol{\theta}) = p(x_{1:k-1}|N_{1:k-1}, \boldsymbol{\theta}) \frac{f_{\boldsymbol{\theta}}(x_k|x_{k-1}) g_{\boldsymbol{\theta}}(N_k|x_k)}{p(N_k|N_{1:k-1}, \boldsymbol{\theta})} \tag{3.10}$$

It is usually impossible to directly sample from the posterior $p(x_{1:k}|N_{1:k}, \boldsymbol{\theta})$. For that matter, importance sampling method proposes to draw samples from an alternative distribution which is less complex and has a parametric form. Importance sampling is an essential part of SMC, and it can be used to approximate an expectation of the target distribution with a weighted average of random samples from a known importance distribution (**ImportanceReview**). Let $f_t \equiv f(x_{1:k})$ be

some function of interest which is integratable with the posterior density $p(x_{1:k}|N_{1:k}, \boldsymbol{\theta})$. Then, the expectation of $f_t$ can be computed from evaluating the integral (**guildtoSMC**)

$$\mathrm{I}(f_t) = E_{p(x_{1:k}|N_{1:k},\boldsymbol{\theta})}[f_t] = \int f_t p(x_{1:k}|N_{1:k}, \boldsymbol{\theta}) dx_{1:k} \tag{3.11}$$

Given N i.i.d samples or particles $\{x_{1:k}^{(i)}\}_{i=1}^{N}$ from the importance distribution $\pi(x_{1:k}|N_{1:k})$, we can compute importance weights of particles as follows

$$w(x_{1:k}^{(i)}) = \frac{p(x_{1:k}^{(i)}|N_{1:k})}{\pi(x_{1:k}^{(i)}|N_{1:k})}. \tag{3.12}$$

Then, the integral can be approximated as a weighted average of functions evaluated

$$\hat{\mathrm{I}}(f_t) = \frac{1}{N}\sum_{i=1}^{N} f_t(i) \frac{w(x_{1:k}^{(i)})}{\sum_{i=1}^{N} w(x_{1:k}^{(i)})} = \frac{1}{N}\tilde{w}(x_{1:k}^{(i)}) \sum_{i=1}^{N} f_t^{(i)} \tag{3.13}$$

with normalized importance weights

$$\tilde{w}(x_{1:k}^{(i)}) = \frac{w(x_{1:k}^{(i)})}{\sum_{i=1}^{N} w(x_{1:k}^{(i)})}, \quad \sum_{i=1}^{N} \tilde{w}(x_{1:k}^{(i)}) = 1 \tag{3.14}$$

An approximation to the posterior $p(x_{1:k}|N_{1:k}, \boldsymbol{\theta})$ is also immediate and defined to be

$$\hat{P}_N(dx_{1:k}|N_{1:k}) = \sum_{i=1}^{N} \tilde{w}(x_{1:k}^{(i)}) \delta_{x_{1:k}^{(i)}}(dx_{1:k}) \tag{3.15}$$

One limitation of importance sampling in the state-space model setting is that it requires all observation $N_{1:k}$ up to the current time bin $(t_{k-1}, t_k]$ for any computation. The computational effort and complexity grow rapidly as time progresses, since we need to generate high-dimensional samples and compute their corresponding importance weights at each time point. Hence, we need a method that does not require to revisit past samples once importance sampling at a given time is finished. For that purpose, **SMCsampler** propose a method to investigate the posterior $p(x_{1:k}|N_{1:k}, \boldsymbol{\theta})$ with SMC sampler which is based on the importance sampling method but does not modify the existing particles.

First, we can observe that the posterior distribution (3.7) of latent states is in a recursive form. Ignoring the normalizing constant in the posterior, then the posterior becomes

$$p(x_{1:k}|N_{1:k}, \boldsymbol{\theta}) \propto p(x_{0:k-1}|N_{1:k-1}, \boldsymbol{\theta}) f_{\boldsymbol{\theta}}(x_k|x_{k-1}) g_{\boldsymbol{\theta}}(N_k|x_k). \tag{3.16}$$

SMC sampler modifies the importance sampling method in a way that it decomposites the importance distribution $\pi(x_{1:k}|N_{1:k})$ into a sequence of intermediate densities. As a result, we can generate samples at current time without browsing the entire history of each particle and compute importance weights. In addition, we let the importance distribution to mimic the state model, that is,

$$\pi(x_{1:k}|N_{1:k}) = \pi(x_k|x_{k-1}, N_k)\pi(x_{1:k-1}|N_{1:k-1}) = f_{\boldsymbol{\theta}}(x_1) \prod_{k=2}^{k} f_{\boldsymbol{\theta}}(x_k|x_{k-1}). \tag{3.17}$$

where $\pi(x_k|x_{k-1}, N_k) = f_{\boldsymbol{\theta}}(x_k|x_{k-1})$ and $\pi(x_1) = f_{\boldsymbol{\theta}}(x_1)$. Then, the importance weights $w_k^{(i)} \equiv w(x_{1:k}^{(i)})$ of particles $\{x_{1:k}^{(i)}\}_{i=1}^N$ are computed as

$$\begin{aligned} w_k^{(i)} &\propto \frac{p(x_{1:k-1}|N_{1:k-1}, \boldsymbol{\theta}) f_{\boldsymbol{\theta}}(x_k|x_{k-1}) g_{\boldsymbol{\theta}}(N_k|x_k)}{\pi(x_{1:k}|N_{1:k})} \\ &\propto \frac{p(x_{1:k-1}|N_{1:k-1}, \boldsymbol{\theta}) f_{\boldsymbol{\theta}}(x_k|x_{k-1}) g_{\boldsymbol{\theta}}(N_k|x_k)}{\pi(x_k|x_{k-1}, N_k)\pi(x_{1:k-1}|N_{1:k-1})} \\ &\propto w_{k-1}^{(i)} \frac{f_{\boldsymbol{\theta}}(x_k|x_{k-1}) g_{\boldsymbol{\theta}}(N_k|x_k)}{\pi(x_k|x_{k-1}, N_k)} \\ &\propto w_{k-1}^{(i)} g_{\boldsymbol{\theta}}(N_k|x_k) \end{aligned} \tag{3.18}$$

Letting the importance distribution to be the same as marginal density of state model does not guarantee a good approximation for the SMC sampler but only to facilitate the computation of importance weights.

SMC sampler decomposites a single importance distribution into a sequence of tractable proposal densities that allow us to sample from a complex posterior distribution. This sampling scheme becomes naturally suitable since the state-space model also has a similar decomposition. At time $t_k$, SMC sampler produces N particles $\{x_{1:k}^{(i)}\}_{i=1}^N$ with with importance weights $w(x_{1:k}^{(i)})$. However, these particles often encounter the degeneracy issue. As the sampler advances in time, importance weights are concentrated on a small group of significant particles since they are calculated proportional to the product of their past weights. On the other hand, most of the particles have really small

weights in sampling and negligible contributions in the later approximation. Degree of degeneracy can be measured by calculating the effective sample size

$$N_{\text{effective}} = \frac{1}{\sum_{i=1}^{N} w(x_{1:k}^{(i)})} \tag{3.19}$$

Effective size decreases as k increases given that the variance of importance weights also tends to increase (**SMCsampler**), eventually, particles fail to represent the posterior distribution given that the sampling distribution is severely skewed.

One solution of the degeneracy issue is to include a resampling filter, or a bootstrap filter, in the propagation of particles, and that allows interactions between different particles whereas particles are generated independently in before. Resampling filter prevents particles with small weights to be propagated further, and distribute important particles approximately according to the posterior distribution (**tutorialfilter**). Resampling step corresponds to take a bootstrap sample (sample with replacement) from current particles depending on their importance weights. Particle with large weights will survive through the filter with high probability, or equivalently, particle with small weights will be replaced by those with larger weights. Consequently, particles will have equal weights $\frac{1}{N}$ after filtering, and the posterior distribution is not approximated by the weighted average rather the unweighted average of filtered particles $\tilde{x}_{1:k}^{(i)}$, i.e.,

$$\hat{P}_N(dx_{1:k}|N_{1:k}) = \frac{1}{N}\sum_{i=1}^{N} \delta_{\tilde{x}_{1:k}^{(i)}}(dx_{1:k}). \tag{3.20}$$

An graphical illustration of the bootstrap filter is provided in **guildtoSMC** as the following

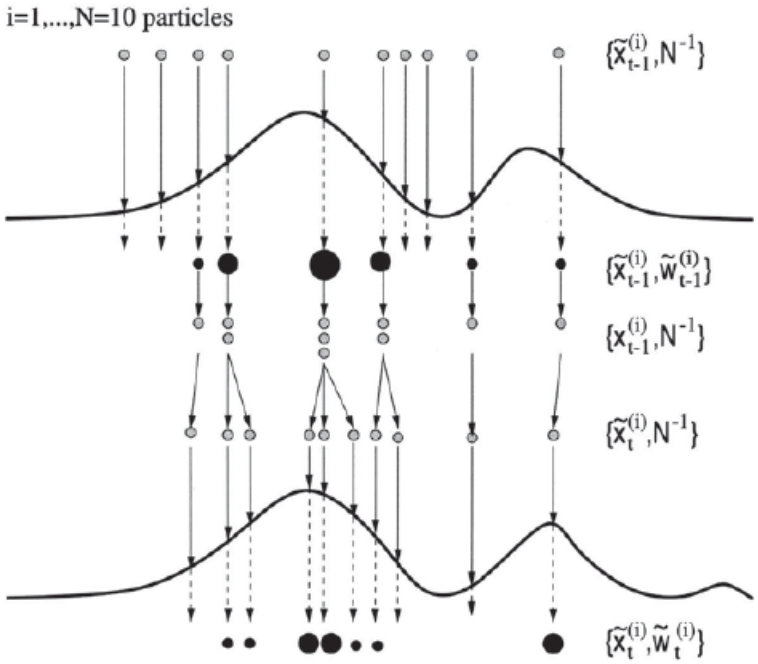


Figure 3.2: The bootstrap filter. Reprinted from *Sequential Monte Carlo Methods in Practice* (p. 12), by Doucet, Arnaud, De Freitas, Nando , and Gordon, Neil, 2001, New York: Springer-Verlag. Reprinted with permission

SMC sampler does not only provide an approximation to the posterior distribution $p(x_{1:k}|N_{1:k}, \boldsymbol{\theta})$ but also produces an estimate of the marginal likelihood $p(N_{1:k}|\boldsymbol{\theta})$. The marginal likelihood can be computed by integrating the latent states from the joint density:

$$p(N_{1:k}|\boldsymbol{\theta}) = \int p(N_{1:k}, x_{1:k}|\boldsymbol{\theta}) dx_{1:k}. \tag{3.21}$$

with a chain-like decomposition as

$$p(N_{1:k}|\boldsymbol{\theta}) = P(N_1|\boldsymbol{\theta}) \prod_{k=2}^{K} P(N_k|N_{1:k-1}, \boldsymbol{\theta}) \tag{3.22}$$

Each component can be estimated by the sum of importance weights $w(x^{(i)}t_{1:k})$ of particles $\{x_{1:k}^{(i)}\}_{i=1}^{N}$ as

$$\hat{P}(N_k|N_{1:k-1}, \boldsymbol{\theta}) = \sum_{i=1}^{N} w(x_{1:k}^{(i)}) \tag{3.23}$$

, and the marginal likelihood has an estimate

$$\hat{P}(N_{1:k}|\boldsymbol{\theta}) = \hat{P}(N_1|\boldsymbol{\theta}) \prod_{k=2}^{K} \hat{P}(N_k|N_{1:k-1}, \boldsymbol{\theta}). \tag{3.24}$$

The SMC algorithm for filtering can be summarized as follows:

**Algorithm 1:** Sequential Monte Carlo for partilce filtering

**Input:** $\boldsymbol{\theta}$

**Output:** $\hat{P}_N(x_{1:k}|N_{1:k}, \boldsymbol{\theta})$,$\hat{P}(N_{1:k}|\boldsymbol{\theta})$

**if** $k = 1$ **then** compute

Generate samples $\{x_1^{(i)}\}_{i=1}^N \sim f_{\boldsymbol{\theta}}(x_1)$

Compute importance weights $w_k^{(i)}$ and normalized weights $\tilde{w}(x_k^{(i)})$

Use resampling filter to obtain $N$ equally weighted particles $\tilde{x}_1^{(i)}$.;

**for** $k = 2 : K$ **do**

Generate samples $\{x_k^{(i)}\}_{i=1}^N \sim f_{\boldsymbol{\theta}}(x_k|\tilde{x}_{k-1}^{(i)})$

Compute importance weights $w_k^{(i)}$ and normalized weights $\tilde{w}(x_k^{(i)})$

Use resampling filter to obtain $N$ equally weighted particles $\tilde{x}_{1:k}^{(i)}$.

**end**

## 3.3 Particle marginal Metropolis-Hasting (PMMH) algorithm

Ideally, the joint posterior distribution $P(\boldsymbol{\theta}, x_{1:k}|N_{1:k})$ can be approximated by Markov Chain Monte Carlo methods. The proposal density of a marginal Metropolis-Hasting sampler for updating both static parameters and latent states can be defined as:

$$Q((\boldsymbol{\theta}^*, x_{1:k}^*)|(\boldsymbol{\theta}, x_{1:k})) = Q(\boldsymbol{\theta}^*|\boldsymbol{\theta})P(x_{1:k}^*|N_{1:k}, \boldsymbol{\theta}^*) \tag{3.25}$$

where $Q(\boldsymbol{\theta}^*|\boldsymbol{\theta})$ is the marginal proposal for $\boldsymbol{\theta}$ and $P(x_{1:k}^*|N_{1:k}, \boldsymbol{\theta}^*)$ produces updates $x_{1:k}^*$ conditioned on the proposed $\boldsymbol{\theta}^*$. Then, the proposal $(\boldsymbol{\theta}^*, x_{1:k}^*)$ is accepted based on the Metropolis-Hasting ratio

as

$$\begin{aligned} &\min\left\{1, \frac{P(\boldsymbol{\theta}^*, x^*_{1:k}|N_{1:k})Q(\boldsymbol{\theta}, x_{1:k}|\boldsymbol{\theta}^*, x^*_{1:k})}{P(\boldsymbol{\theta}, x_{1:k}|N_{1:k})Q(\boldsymbol{\theta}^*, x^*_{1:k}|\boldsymbol{\theta}, x_{1:k})}\right\} \\ =&\min\left\{1, \frac{P(N_{1:k}|\boldsymbol{\theta}^*)P(\boldsymbol{\theta}^*)Q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)}{P(N_{1:k}|\boldsymbol{\theta})P(\boldsymbol{\theta})Q(\boldsymbol{\theta}^*|\boldsymbol{\theta})}\right\} \end{aligned} \tag{3.26}$$

However, direct sampling from the posterior $P(x^*_{1:k}|\boldsymbol{\theta}^*, N_{1:k})$ is not feasible by standard MCMCs method, and the marginal likelihood of static parameters cannot be explicitly computed given that the state-space model is not Gaussian or linear.

**PMCMC** proposed a class of MCMC methods that can produce exact approximations to ideal MCMC algorithms, where an ideal algorithm can either sample directly from the the joint posterior or the posterior of latent states. One sampler of PMCMC methods is the particle marginal Metropolis-Hasting (PMMH) sampler. PMMH is designed to perform as an ideal marginal Metropolis-Hasting(MMH) sampler and utilizes the same proposal procedure (3.25). Ideal MMH sampler relies on the access of $P(x^*_{1:k}|\boldsymbol{\theta}^*, N_{1:k})$, whereas, PMMH sampler employs SMC to approximate the posterior of latent states as well as the marginal likelihood of static parameters $P(N_{1:k}|\boldsymbol{\theta})$. One advantage of the PMMH sampler in inferences for state-space models is that the sampler does not require great effort to construct the proposal density for states. If transition density of latent process is used to construct the importance distribution in SMC step, then the proposal of states only depends on the static parameters. That is, we only need to design a proposal density $Q(\boldsymbol{\theta}^*|\boldsymbol{\theta})$. In addition to easy implementation, PMMH sampler ensures that invariant distribution of samples $x^{(j)}_{1:k}, \boldsymbol{\theta}^{(j)}$ is the joint posterior $P(x^*_{1:k}|\boldsymbol{\theta}^*, N_{1:k})$. PMMH algorithm is described as follows

**Algorithm 2:** PMMH sampler

**Input:** $\boldsymbol{\theta}^{(0)}$, prior $P(\boldsymbol{\theta})$

**initilization**:

Use SMC to obtain samples $x_{1:k}^{(0)}$ from $P(x_{1:k}|\boldsymbol{\theta}^{(0)}, N_{1:k})$;

Compute marginal likelihood $\hat{P}(N_{1:k}|\boldsymbol{\theta}^{(0)})$

**for** $i = 1 :$ *maxiter* **do**

Propose $\boldsymbol{\theta}^*$ from proposal density $Q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(i-1)})$;

Use SMC to obtain samples $x_{1:k}^*$ from $P(x_{1:k}|\boldsymbol{\theta}^*, N_{1:k})$ and compute $\hat{P}(N_{1:k}|\boldsymbol{\theta}^*)$;

Compute the acceptance probability

$$\alpha = \frac{\hat{P}(N_{1:k}|\boldsymbol{\theta}^*)P(\boldsymbol{\theta}^*)Q(\boldsymbol{\theta}^{(i-1)}|\boldsymbol{\theta}^*)}{\hat{P}(N_{1:k}|\boldsymbol{\theta}^{(i-1)})P(\boldsymbol{\theta}^{(i-1)})Q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(i-1)})}$$

Generate $u$ from Uniform[0,1] ;

**if** $u < \alpha$ **then**

Update $\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^*$, $x_{1:k}^{(i)} = x_{1:k}^*$, $\hat{P}(N_{1:k}|\boldsymbol{\theta}^{(i)}) = \hat{P}(N_{1:k}|\boldsymbol{\theta}^*)$ ;

**else**

Keep $\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)}$, $x_{1:k}^{(i)} = x_{1:k}^{(i-1)}$, $\hat{P}(N_{1:k}|\boldsymbol{\theta}^{(i)}) = \hat{P}(N_{1:k}|\boldsymbol{\theta}^{(i-1)})$ ;

**end**

**end**

## 3.4 Smoothing

Since we have discretized the point process to associate it with a latent process, the intensity function will be represented by a sequence of discrete samples evaluated from the model

$$\lambda_k = \exp(\mu + \beta x_k) \tag{3.27}$$

given the estimates of latent states. However, the true intensity function should be able to be evaluated at anytime in the observation period rather than only a set of sample time points. If we join discrete samples to construct an estimate of the intensity function, the resulting curve is often

too wiggly. In addition, the unsteadiness of the estimate may overshadow any significant pattern we could observe as if we consider the intensity function as a single entity during the observation interval. Therefore, we can use interpolation or smoothing techniques from functional data analysis framework to obtain a smooth estimate of the intensity. Smoothing can reduce fluctuations in a curve estimate and help us to capture the entire perspective of the intensity function. Derivative of the intensity estimate is also immediately available if we use basis expansion methods for smoothing.

One approach is to express the intensity function by a set of basis functions. Let the discrete samples $\hat{\lambda}_k$ to be the pseudo-observations $y_k$ from the intensity function $\lambda(t)$ such that

$$y_k = \lambda(t_k) + \epsilon_k \tag{3.28}$$

where $\epsilon_k$ corresponds to the observational noise. The function $\lambda(t)$ is represented by a linear combination of known basis functions $\phi_c(t)$:

$$x(t) = \sum_{c=1}^{C} \beta_c \phi_c(t) \tag{3.29}$$

with basis coefficients $(\beta_1, ..., \beta_C)$. Then, the coefficients can be estimated by minimizing the sum of squared error, i.e.,

$$\sum_{k=1}^{N} [y_k - \sum_{c=1}^{C} \beta_c \phi_c(t_k)]^2 \tag{3.30}$$

Let $\boldsymbol{\beta}$ be the vector containing C basis coefficients and $\boldsymbol{\Phi}$ be a $N * C$ matrix storing evaluations of C basis functions at time points $(t_1, ..., t_N)$. Then, the sum of squared error can also be represented by

$$||\boldsymbol{y} - \boldsymbol{\beta}^T \boldsymbol{\Phi}||^2 = (\boldsymbol{y} - \boldsymbol{\beta}^T \boldsymbol{\Phi})^T (\boldsymbol{y} - \boldsymbol{\beta}^T \boldsymbol{\Phi}) \tag{3.31}$$

where $\boldsymbol{y}$ is the vector containing n observations.

Smoothing by basis expansion method is similar to a multiple regression problem and able to provide a smooth and well-behaved estimate of the intensity function. Employ cubic B-spline basis functions for its compact support property, and use $L_2$ penalization to alleviate the burden of choosing the number of knots and their locations. As a result, estimated intensity trajectory is more

stable and can avoid overfitting. The $L_2$ or roughness penalty is defined as

$$\gamma \int_0^T [\frac{d^2\lambda(t)}{dt^2}]^2 dt \tag{3.32}$$

with smoothing parameter $\gamma$ which controls the roughness of the trajectory given the total curvature of the function over time. Then, basis coefficients are estimated by minimizing the penalized mean squared error criterion as follows

$$l(\boldsymbol{\beta}) = (\boldsymbol{y} - \boldsymbol{\beta}^T\boldsymbol{\Phi})^T(\boldsymbol{y} - \boldsymbol{\beta}^T\boldsymbol{\Phi}) + \gamma \int_0^T [\frac{d^2\lambda(t)}{dt^2}]^2 dt. \tag{3.33}$$

Define a C * C matrix $\boldsymbol{V}$ with entries $v_{ij} = \int_0^T \phi_i''(t)\phi_j''(t)dt$, it follows that the roughness penalty can be evaluated by matrix multiplication as

$$\int_0^T [\frac{d^2\lambda(t)}{dt^2}]^2 dt = \int_0^T [\sum_{c=1}^{C} \beta_c \phi_c''(t)]^2 dt = \boldsymbol{\beta}^T\boldsymbol{V}\boldsymbol{\beta} \tag{3.34}$$

Integrals can be well approximated by the composite Simpson's rule. We define an even number Q of quadrature points $(t_0, t_1, ..., t_Q)$ for the integration interval $[0, T]$ such that $t_0 = 0$ and $t_q = t_{q-1} + h$ for $h = T/Q$. For any integral $\int_0^T \phi_i''(t)\phi_j''(t)dt$, the rule gives an approximation as

$$\int_0^T f_{ij}(t)dt = \frac{h}{3}[f_{ij}(t_0) + 2\sum_{s=1}^{\frac{Q}{2}-1} f_{ij}(t_{2s}) + 4\sum_{s=1}^{\frac{Q}{2}} f_{ij}(t_{2s-1}) + f_{ij}(t_Q)] \tag{3.35}$$

where $f_{ij}(t) = \phi_i''(t)\phi_j''(t)$. The penalized sum of squared error with matrix representation becomes

$$l(\boldsymbol{\beta}) = \frac{1}{N}||\boldsymbol{y} - \boldsymbol{\beta}^T\boldsymbol{\Phi}||^2 + \gamma\boldsymbol{\beta}^T\boldsymbol{\Phi}\boldsymbol{\beta} \tag{3.36}$$

Similar to least square estimation, we set the derivative of $l(\boldsymbol{\beta})$ to be 0 with respect to $\boldsymbol{\theta}$ and solve the estimating equation

$$(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \gamma\boldsymbol{V})\boldsymbol{\beta} = \boldsymbol{\Phi}\boldsymbol{y} \tag{3.37}$$

Solution to estimating equation has a closed form as

$$\boldsymbol{\beta} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \gamma\boldsymbol{V})^{-1}\boldsymbol{\Phi}^T\boldsymbol{y} \tag{3.38}$$

Knots can be equally spaced since the density of observations of intensity is constant through out the observational period. Then, there are two tuning parameters to consider: the number of basis function and the smoothing parameter. We need more basis functions or knots to ensure good local estimation and use penalization to avoid overfitting. We can consider several candidates for the number of basis functions, and choose the minimum when the estimated curves are similar. Once the number of basis function is fixed, smoothing parameter can be determined by the generalized cross validation (gcv) method with grid search. Given a choice of $\lambda$, the gcv scores is defined as

$$\text{gcv}(\lambda) = \frac{1}{N}\sum_{k=1}^{N}(y_k - \hat{\lambda}^{(-k)}(t_k))^2 \tag{3.39}$$

where $\hat{\lambda}^{(-i)}(t_k)$ represents the estimate of intensity with k-th sample being removed from estimation. Optimal $\gamma$ has the minimum gcv score.

# Chapter 4

# Real data application

## 4.1 Introduction to experiment and data

Neural data were acquired from the experiments conducted by Tianjin University in 2012. Experimental units were adult male Sprague-Dawley rats with weights of around 200 grams. Neurons of interest are wide dynamic range (WDR) neurons located in the dorsal horn of the spinal cord, since they are believed to be responding to pain stimuli applied to the Zusanli acupoint on the right leg. After anesthetizing the rat and exposing its lumbosacral spinal cord, micro-electrodes were then vertical inserted into spine for recording electric signals from multiple WDR neurons. Pain stimulation was done by lifting and thrusting stainless steel needles to the acupoint for 120 times per minute. The entire experiment lasted for 16 minutes: (1) 2 minutes without any stimulation for rat to be in a resting state, (2) 2 minutes for which the needle was only kept in the acupoint, (3) 2 minutes for which pain stimulation was applied to the acupoint, (4) 5 minutes after where pain stimulation is stopped with needle kept in the acupoint, and (5) 5 minutes with no stimulation for rats to return to a resting state. Under stimulation, we could observe fluctuations in neural activities.

Signals recorded by each electrode contain not only membrane potentials of several neurons but also noises, hence we need to detect individual spikes and assigning them to each neuron for further analysis. **sorting** have proposed a spike detecting and sorting algorithm which consists of three steps: (1) spike detection, (2) feature extraction, and (3) clustering. First, the raw signal is processed through a band-pass filter (300 - 3000 Hz). Let $x$ be the filtered signals, then a spike is

detected when signals pass a threshold $V$ defined as

$$\sigma_x = \text{median}\left\{\frac{|x|}{0.6745}\right\}, \quad V = 4\sigma_x. \tag{4.1}$$

Then, we take a sample of signals around each detected spike for feature extraction. Applying wavelet transform defined in **sorting** to each sample, we are able to obtain 64 wavelet coefficients that characterize the features of spike shapes at different scale and times. Selecting two coefficients that can separate different class of spikes the best, we can employ superparamagnetic clustering (SPC) algorithm to assign spikes with similar features to one cluster which represents one of the neuron being monitored.

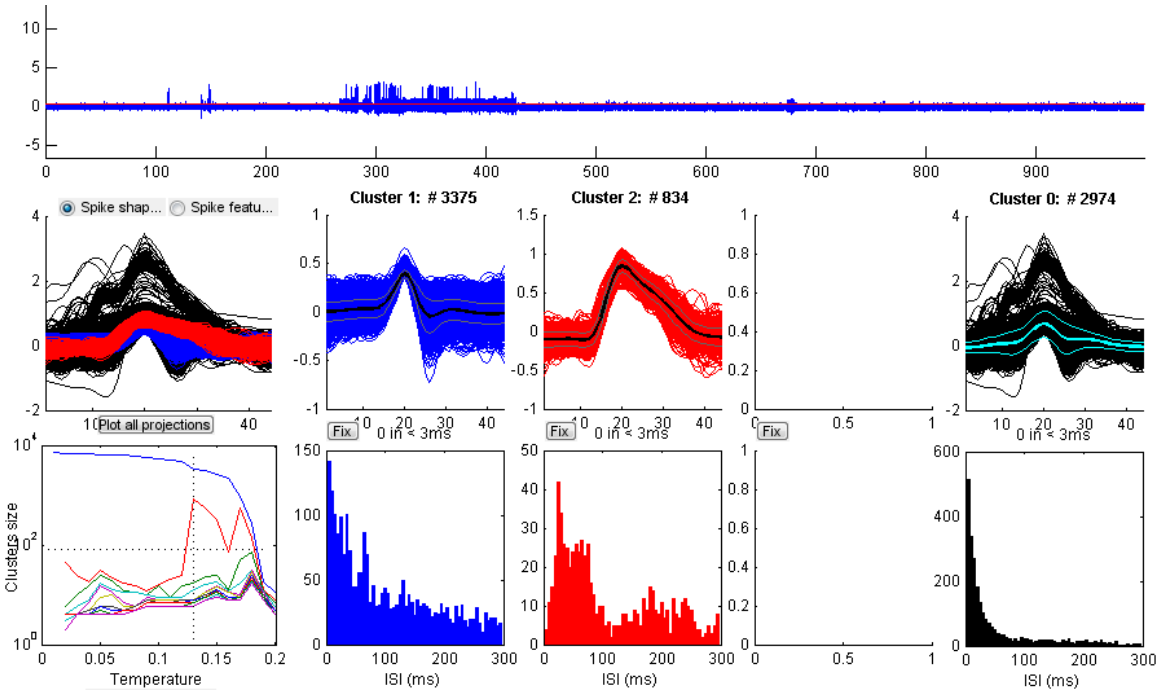


Figure 4.1: Detection and sorting of spikes

Top panel presents the entire signals recorded by the electrode after band-pass filtering, and the red line corresponds to the threshold $V$. All samples of spikes are collected for wavelet transform which provides wavelet coefficients. Running SPC algorithm on two coefficients that can separate different types of spikes, we obtain clusters of spikes where each cluster represents each neuron.

After sorting signals and detect spikes for all neurons , we are able to identify 12 neurons and their corresponding spike train data as shown in Figure 4.2. Spike train data of Neuron 1 is then used for particle filtering and intensity estimation.

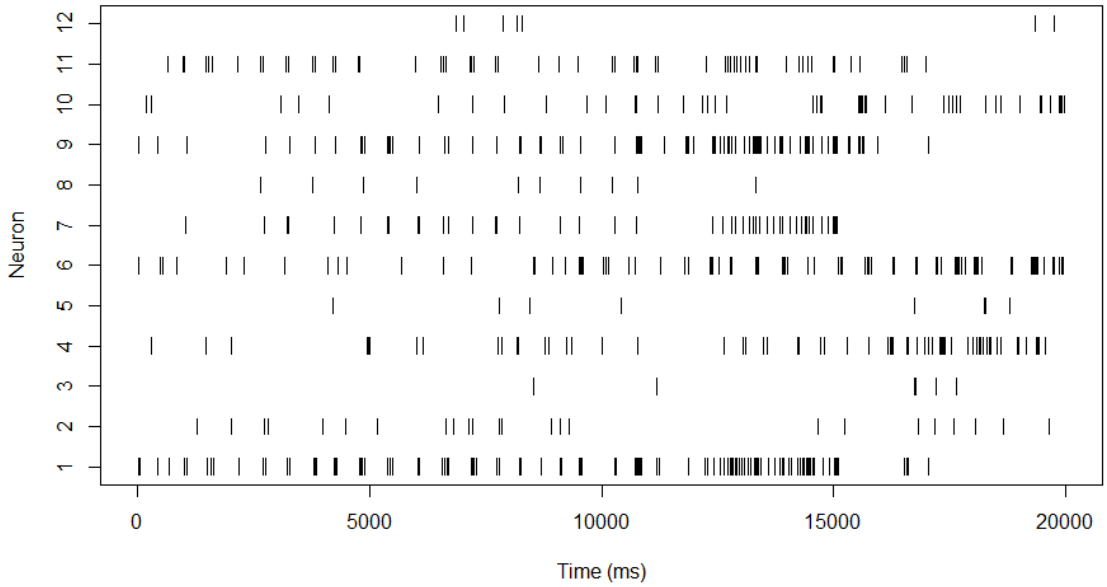


Figure 4.2: 12 neural spike trains detected from 4 membrane potential recordings

## 4.2 PMMH result

For latent states and parameters estimation, we choose an observation interval with length of T = 3000 ms at the start of the third phase of the experiment when a constant puncture stimulus is applied. The time resolution is chosen to be $\Delta$ = 3 ms to ensure that there is at most one spike per time bin. Before running MCMC on parameters, a one-to-one transformation is made on $\phi$, i.e., we introduce a new parameter $\phi = \tanh(\rho)$ to ensure the stationarity of latent process when a random walk proposal is used for both $\mu$ and $\rho$. Proposal density for particles is chosen to be the transition density of the latent process, and 500 particles are used in the SMC algorithm.

The priors used for $\mu$ and $\rho$ are normal priors $N(-5, 2)$ and $N(1, 0.1)$ respectively. The random walk proposal for parameters in PMMH is defined as $N(\mu^{(i-1)}, 0.15)$ for $\mu$ and $N(\rho^{(i-1)}, 0.05)$ for $\rho$ where $i$ corresponds to the iteration number. Andrieu, Doucet and Holenstein (2010) states that a high acceptance rate is desirable in particle filtering, and the acceptance rate of PMMH is 82% under this setting. Sampling result is illustrated in the following figures:

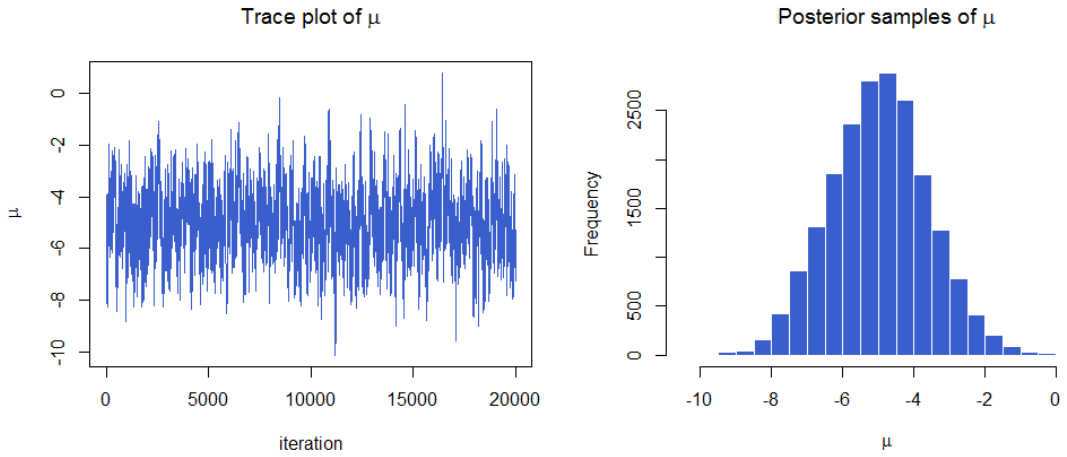


Figure 4.3: PMMH sampling result of paramters $\mu$

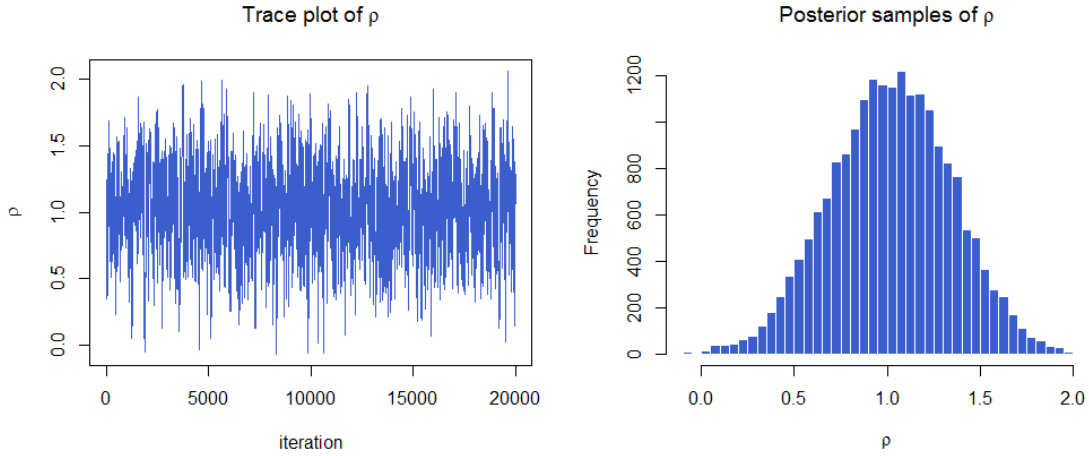


Figure 4.4: PMMH sampling result of paramters $\rho$

Trace plots indicate satisfactory mixing of MCMC chains of both $\mu$ and $\rho$. We see a higher autocorrealtion for MCMC chain of $\mu$ compared to that of $\rho$ as shown in Figure 4.5. Even though mixing of $\rho$ appears to be better than $\mu$, we see better shrinkage of the prior for $\mu$ in Figure 8 and Figure 9. The highest probability density (HPD) region for both parameters are included in Figure 4.6 and 4.7, and summary statistics is illustrated in Table 1.

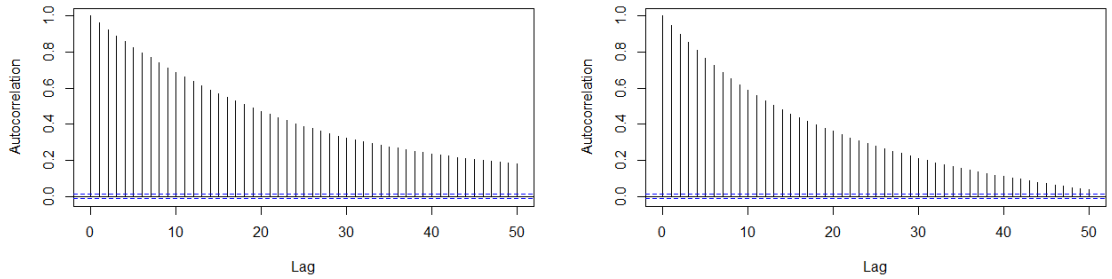


Figure 4.5: Autocorrelation functions of MCMC chains of $\mu$ (left) and $\rho$ (right).

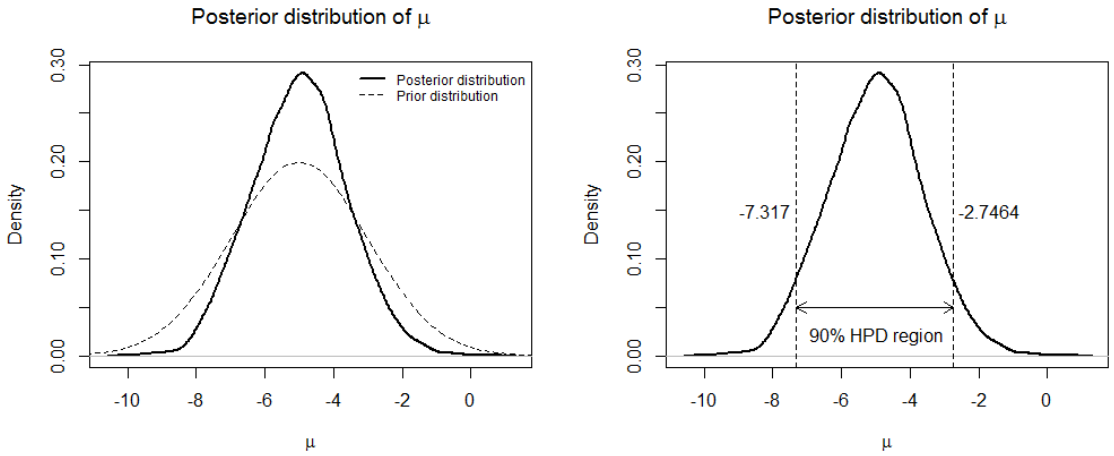


Figure 4.6: Posterior density and 90% high posterior density region of $\mu$

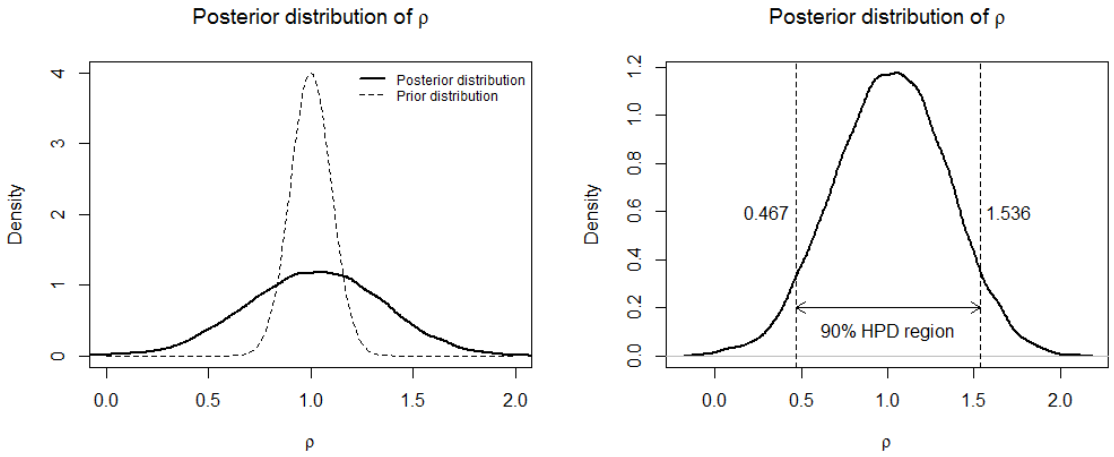


Figure 4.7: Posterior density and 90% high posterior density region of $\rho$

Table 4.1: Statistics of posterior samples of parameters

| | Posterior mean | Posterior variance | Effective sample size |
|---|---|---|---|
| $\mu$ | -4.985 | 1.968 | 367 |
| $\rho$ | 1.019 | 0.106 | 527 |

## 4.3 Smoothing result

Using posterior means of both parameters and latent states, we can obtain a single trajectory estimate of the conditional intensity function as the black line in the following figure.

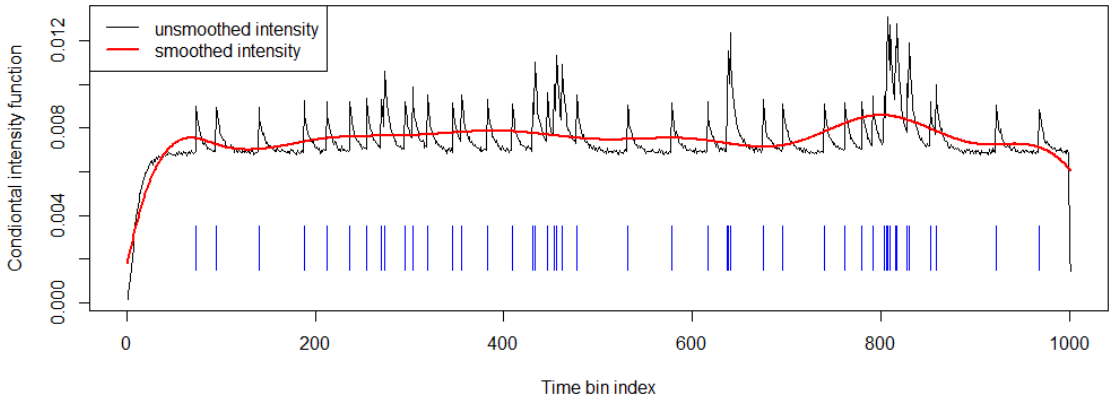


Figure 4.8: Unsmoothed and smoothed conditional intensity function

The unsmoothed estimate of intensity appears to have too many fluctuations, and local changes may overshadow the overall pattern of neural activities under stimulation. Applying smoothing technique allows us to have a stable estimate that focuses on capturing the general behavior. Moreover, using basis expansion method can give immediate estimate of derivative of conditional intensity function as shown in the following figure:

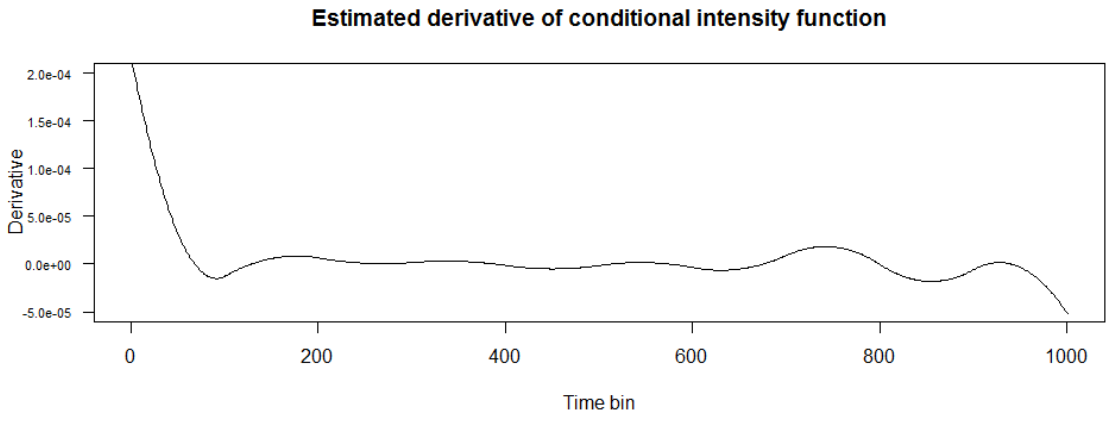


Figure 4.9: Derivative estimate of conditional intensity function

We can observe the adaption behaviour of neuron based on the derivative. The derivative starts high in magnitude at the beginning of stimulation and declines quickly to a normal state. It shows that the neuron has adapted to the stimulation since the derivative remains closely to 0. On the other hand, we cannot observe the adaption behavior from the unsmoothed trajectory since it is constantly surging back and forth throughout the observation period. In addition, the unsmoothed trajectory seems too sensitive to the individual spikes, whereas, the smoothed trajectory is able to capture clusters of spikes.

# Chapter 5

# Simulation study

## 5.1 Simulation method

Simulation study is based on synthetic neural spiketrains and aims to verify that the estimation procedure would produce stable and reliable estimates. Synthetic spike trains are generated by thinning algorithm described in **simThinning** Thinning algorithm generates a non-homogeneous Poisson process without the need of integrating the conditional intensity function and can be used for any rate or conditional intensity function. Let $\lambda^u$ be the maximum for a conditional intensity function $\lambda(t)$. First, we generate a homogeneous Poisson process according to the constant intensity $\lambda^u$. Each spike in the simulated process is then accepted with probability $\frac{\lambda(t)}{\lambda^u}$. The remaining spikes will resemble a non-homogeneous Poisson process with conditional intensity function $\lambda(t)$. The details of thinning algorithm is summarized as follows:

**Algorithm 3:** Thinning algorithm for simulating neural spike trains

**Input:** $\lambda^u$, interval length T

**Output:** Spike times $s_1, ..., s_k$

Set t = 0 and k = 1 ;

**while** $t < T$ **do**

  Generate an exponential variable E with rate parameters $\lambda^u$ and let $t = t + E$ ;

  Generate U from the uniform distribution U(0,1) ;

  **if** $U < \frac{\lambda(t)}{\lambda^u}$ **then**

    $s_k = t$ and $k = k + 1$ ;

  **end**

**end**

## 5.2 Simulation results

For simulation study, 50 spike trains are generate to run both PMMH and smoothing based on the smoothed intensity estimate as shown in Figure 4.8. Then, we run PMMH (same proposals as for the real data) and smoothing (tuned by GCV) for each simulated spike train. Using posterior means from each PMMH result, we can obtain a smoothed trajectory as an estimate of conditional intensity function for that spike train. The following figure includes intensity estimates for 50 simulated spike trains and the true intensity used for simulation. Also, a 95% point-wise confidence interval is calculated based on intensity estimates from simulated spike trains. We can observe that the true intensity is well covered by the confidence interval.

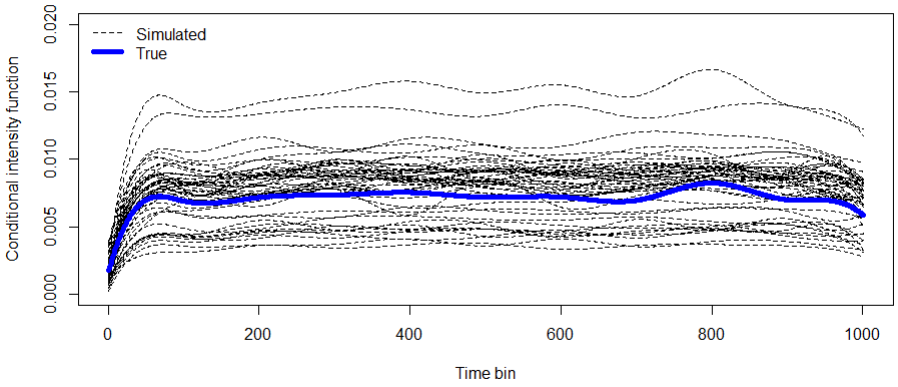


Figure 5.1: Estimated conditional intensity function for 50 simulated spike trains

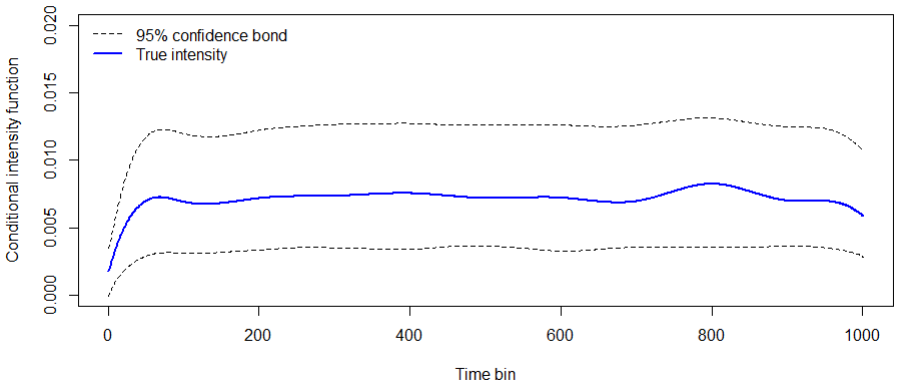


Figure 5.2: 95% point-wise confidence interval for true CIF

# Chapter 6

# Conclusion and Discussion

In this work, PMMH sampler and smoothing technique are studied with neural spike train data. PMMH is able to provide excellent estimate of conditional intensity function for a real neural spike train given implicit stimulus. In addition, PMMH is easy to implement and does not require great effort in tuning. Smoothing technique ensures that any estimated intensity from PMMH can be further refined to become a more stable trajectory. For both real data analysis and simulation study, smoothing is a suitable solution for estimates to comply the assumption that neural responses should be moderate under constant stimulation. Consequently, a reliable estimate of intensity allows us to correctly quantify neuron responses and examine how neurons process stimulation information.

Point process model is perfectly adequate for characterizing a neural spike train. It offers a conditional intensity function serving as a rate code to understand neural activities. Moreover, it simplifies the modelling problem to modelling the conditional intensity, since its likelihood can be uniquely and completely expressed in terms of CIF. If stimulus is explicit (measurable and quantifable), then we can construct a parametric CIF model including stimulus effect directly as shown in Turccolo et al. (2005). However, the brain or sensory neurons are often receiving stimulations that cannot be defined with a precise and consistent unit of measurement. Implicit stimulus gives rise to a latent process determining how stimulus influences the underlying behaviour of neurons. Paralleling the observed neural spike train and the latent process suggests a state-space model framework. The central problem of a state-space model is state inference, and SMC algorithm is ideal for estimating latent states. PMMH sampler adopts SMC algorithm so that it can update parameters of observed point process given estiamtes of latent states and marginal likelihood. In order present any trajectory of CIF from estimation, the trajectory needs to be smoothed for

reducing local fluctuations that overshadow the overall neural behaviour. In chapter 4, we can see that trajectory calculated by posterior mean of parameters and states becomes less locally variable after smoothing. More importantly, smoothing by basis expansion is able to provide the derivative of intensity which can offer another angle to understand neural behaviour. In this case, we can observe that the adaption process of neuron can be reflected in the derivative. As a result, the smoothed trajectory, combined with its derivative, is able to provide a more general and direct picture of neural activities under stimulation.

One disadvantage of PMMH algorithm is its computational complexity $O(TN)$. Given longer observation time $T$, PMMH algorithm requires huge increase in particle size N to maintain an acceptable effective sample size of both parameters and latent states. Even though it is convenient to use transition density of latent process as a proposal density for SMC, the choice can be suboptimal sometimes. Hence, it may require great effort to construct a good proposal prior to running PMMH algorithm. The state-space model is able to include stimulus effect and spiking history of neuron itself, but it ignores interaction effect received from other neurons when neurons are usually group together to form a network structure. Current measuring techniques allow simultaneous recording of multiple neurons, and an extension of the current modelling approach is to investigate the influence from other neurons in the group.