High-Dimensional Ordinary Differential Equation Models for Connectivity Studies

Jingwei Wu
Shaowu, Fujian, China

M.S., University of Virginia, USA, 2012
B.S., Tsinghua University, P. R. China, 2010

A Dissertation Presented to the Graduate Faculty of
University of Virginia in Candidacy for the Degree of
Doctor of Philosophy

Department of Statistics

University of Virginia
May, 2015

Abstract

We introduce a dynamic directional model (DDM) for studying brain effective connectivity based on intracranial electrocorticographic (ECoG) time series. The DDM consists of two parts: a set of differential equations describing neuronal activity of brain components (state equations), and observation equations linking the underlying neuronal states to observed data. The combined high temporal and spatial resolution of ECoG data result in a much simpler DDM, allowing investigation of complex connections between many regions. To identify functionally-segregated subnetworks, a form of biologically economical brain networks, we propose the Potts model for the DDM parameters. The neuronal states of brain components are represented by cubic spline bases and the parameters are estimated by minimizing a log-likelihood criterion that combines the state and observation equations. The Potts model is converted to the Potts penalty in the penalized regression approach to achieve sparsity in parameter estimation, for which a fast iterative algorithm is developed. An L_1 penalty is also considered for comparison. The methods are applied to an auditory ECoG data set and extensive simulation studies.

Acknowledgements

As writing this dissertation I suddenly realized that my PhD study was sadly approaching the end. I want to express my thanks to everyone that shared with me, brought joy to, and helped me through this special five year journey.

I am indebted to my advisor Professor Tingting Zhang, who took me as her first ever PhD student. Tingting has been perseveringly sharing her knowledge of ODE modeling, statistics and scientific thinking with me, and has pointed out many immature ideas of mine. I enjoyed every bit of discussion we had, no matter it was in her office, at Newcomb Hall or at 2 am through a long thread of emails. I am in awe of her enthusiasm for research, work ethic, sharp intuituion, professional way of communicating with collaborators, humble and personable personality.

I am thankful with all my heart to my dissertation committee, Professor Jeff Holt, who is willing to assist with any academic or nonacademic issues, Dan Keenan, whose passion towards research will surely influence me lastingly and who also taught me Applied Time Series and Actuarial Statistics, Yanjun Qi who is able to shed light on my research from a computer scientist point of view, Jianhui Zhou, who has deep understanding in functional data analysis and dimension reduction, and has been a great mentor and friend. I also benefited considerably from my excellent teachers: Professor Ted Chang, Randy Cogill (Systems Engineering), Alfredo Garcia (Systems Engineering), Christian Gromoll (Math), Gianluca Guadagni (Math), Tao Huang, Feifang Hu, Tai Melcher (Math), Gabrial Robins (Computer Science), Dan Spitzner, and Amanda Wang. I am also grateful to the statistics department who provides me many opportunities, including the trust in me to teach 4 undergraduate courses (2120 twice, 3120 twice), from which I enjoyed delivering statistics knowledge to as well as learning from communicating with students with different majors and

backgrounds. I'd also like to extend my gratitude to Mrs. Karen Dalton, who has created a warm and sweet atmosphere at Halsey, and knows everything from when the last day to drop a class is to where the "secret key" is hidden. During the past two years I have become one of the most heavy users of UVA cluster, it's impossible for me to have performed so many computational tasks without the help of Dr. Katherine Holcomb and Dr. Ed Hall from UVACSE, who are the experts of high performance computing and guided me through many technical difficulties.

Most of my research time was spent in Halsey 123 - the windowless graduate office. I still remember there was one time I was working after 1 am and didn't realize my car was almost buried by the snow. And I cannot count how many naps I have taken at the red leather couch in the lobby of Halsey during the weekends and holidays. It's my great honor to have worked with so many intelligent and approachable officemates and classmates, including those who have graduated - Yanqing Hu, Yue Liu, Zhenjun Ma, Jing Guo, Wei Ma, Paul Manser, Xiaoxiao Tang, Oliver Bluemke, Paul Diver, Dazhi Gong, Wenyuan Lin, Feiyang Niu, Robert Pfister, Chris Tait, Xiwei Tang, Yuan Xue, Bailey Zhang, Xiaoming Li, Andy Lin, Chenyi Pan, Caitlin Steiner, Xin Liang, Miao Lv, Taylor Brown, Maria Ferrara, Qiannan Yin, Yin Zhang.

I'd like to reserve the last gratitude to my family. In particular, I'd like to thank my parents Meng Wu and Meiqin Liao. They have given me unwavering and unconditional support and love to every aspect of my life, without which I couldn't have made this far.

Contents

Chapter 1

Introduction

The ordinary differential equation (ODE) model has been used in many scientific disciplines to describe temporal changes in physical, biological or biochemical systems, and provides mechanistic insights and causal interpretation of the relationship among system components. However, when using ODEs to model a high-dimensional dynamic system such as the brain, difficulties with model specification, estimation and validation raise challenges. Consequently, only a few statistical methods are available for modeling and inference of high-dimensional dynamic systems within an ODE framework. These challenges, the demand for novel statistical methods to solve high dimensional ODE models and the high quality of real data are the motivation of this dissertation. We will first review the existing strategies to estimate parameters of ODEs in Section 1.1. Although ODE models can be applied in many areas, in this dissertation we are particularly interested in modeling brain effective connectivity through the Electrocorticography (ECoG) time series data because of its high temporal and spatial resolution. In Section 1.2 we will introduce different brain imaging data especially ECoG time series.

1.1 Overview of Parameter Estimation Strategies for Differential Equations

1.1.1 Problem Setup

A dynamic system is built up by time-dependent neuronal state functions whose evolution processes are characterized by a set of ordinary differential equations. Motivated by the problem of our ECoG data to study brain connectivity, we consider a general ODE model with non-time-varying coefficient:

$$\frac{d\boldsymbol{x}(t)}{dt} = F(\boldsymbol{x}(t), \boldsymbol{u}(t), \Theta),$$

where $t \in [0, T]$ is time (independent variable), $\boldsymbol{x}(t) = (x_1(t), \ldots, x_d(t))$ is a d-dimensional vector of neuronal state functions that constitutes the system, $\boldsymbol{u}(t)$ is a known input function vector, Θ is a multi-dimension parameter set that we aim to estimate, and $F = (F_1, \ldots, F_d)$ is a d-dimensional vector of differentiable functions with known forms but unknown parameter set Θ, the ith function F_i describes the dynamics of the ith neuronal state function characterized by interactions of both neuronal state function vector $\boldsymbol{x}(t)$ and input function vector $\boldsymbol{u}(t)$. The neuronal state function vector $\boldsymbol{x}(t)$ are measured at time $t_1, \ldots, t_n$ with measurement errors, i.e.,

$$\boldsymbol{y}_i = \boldsymbol{x}(t_i) + \boldsymbol{\epsilon}_i, 1 \leq i \leq n,$$

where $\boldsymbol{y}_i$ is the ith observation vector, and $\boldsymbol{\epsilon}_i$ is a d-dimensional vector of error terms with mean zeroes and variance-covariance matrix Σ. The strategies to estimate parameter of ODEs tend to fall into two categories: discretization methods using

numerical approximation and basis function expansion.

1.1.2 Existing Approaches

Ogunnaike and Ray (1994) proposed a nonlinear least square (NLS) method to estimate the parameters of the ODE model. The NLS iterates between two steps to search for the Θ that minimizes the sum of squares of the discrepancies between observed and fitted neuronal states:

$$\text{SSE}(\Theta) = \sum_{j=1}^{n} (\hat{x}(t_j; \Theta) - y_j)^2,$$

where $\hat{x}(t_j, \Theta)$ is the estimated value of x at t_j as a function of parameter set Θ (they assume single state function). For each trial parameter set Θ, the ODEs can be solved numerically by standard numerical schemes such as Runge-Kutta algorithm, obtaining the fitted value $\hat{\boldsymbol{x}}$. Hence by trying different parameter sets we can build the whole SSE surface as a function of Θ, then the parameter estimation boils down to locating the global minimum in the SSE surface. They described an iterative algorithm which was commonly used to estimate ODE parameters. First the ODEs are solved using numerical schemes and trial parameters (and initial values). Then an optimization algorithm will improve the parameter estimation by providing a better search direction. By linearizing the SSE with first order Taylor approximation, we can rewrite SSE as

$$\begin{aligned}
\text{SSE}(\Theta + \delta\Theta) &= ||\Delta(\Theta + \delta\Theta)||^2 \\
&= ||\Delta(\Theta) + \frac{\partial \Delta}{\partial \Theta}||^2 \\
&= ||\Delta(\Theta)||^2 + 2\delta\Theta^T (\frac{\partial \Delta}{\partial \Theta})^T \Delta + \delta p^T (\frac{\partial \Delta}{\partial \Theta})^T (\frac{\partial \Delta}{\partial \Theta}) \delta\Theta,
\end{aligned}$$

where $\Delta = \hat{\boldsymbol{x}}(\Theta) - \boldsymbol{y}$ is the discrepancy between fitted and observed value. Now we can identify the best "direction" by solving a quadratic problem. Sensitivity equations can be solved to find the Jacobian matrix. Iteration between numerically solving ODEs and updating parameters continues until the optimization function cannot be improved by some pre-specified significance level. Although this method is straightforward and many other works are based upon it, it suffers from many drawbacks, such as expensive computational cost with most computational effort spent on solving ordinary differential equations numerically, high sensitivity of numerical scheme of solving ODE, low convergence rate, and the need for initial or boundary conditions to solve ODE numerically.

Varah (1982) proposed a spline least square method which opened another direction for ODE parameter estimation. The biggest advantage of this method is that it avoids solving ODEs numerically hence reduces the computation burden tremendously and overcomes other drawbacks caused by solving ODEs as well. The spline least square method is a two-step approach. In the first step the state functions and their derivatives are represented by cubic splines:

$$\hat{x}_i(t) = \sum_{j=1}^{p} \Gamma_{ij}\phi_j(t),$$

where $\phi_1(t), \ldots, \phi_p(t)$ are the basis functions, p is the number of basis, and Γ_{ij} is the coefficient corresponding to jth basis function when approximating for ith state function. Then the curve fitting is converted to a problem of estimating the coefficient matrix Γ. The criterion of selecting Γ is to minimize the sum of squares

of the fitted errors:

$$\sum_{j=1}^{n} ||\boldsymbol{y}_j - \hat{\boldsymbol{x}}(t_j)||^2 ,$$

where $\boldsymbol{y}_j$ is the jth (observed at t_j) observation vector across all state functions. In the second step the parameters were estimated by minimizing the following objective function:

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}} \sum_{j=1}^{n} \left|\left| \frac{d\hat{\boldsymbol{x}}(t_j)}{dt} - F(\hat{\boldsymbol{x}}(t_j), \boldsymbol{u}(t_j), \Theta) \right|\right|^2 .$$

Note that this method minimizes the "wrong" sum of squares, and this is precisely the reason why it can avoid solving ODEs. This method can only provide reasonable solutions and the results depend heavily on the curve fitting step. When profiling the raw observation into functional data object, the trade-off between function being overly smooth and overly fluctuating plays an important role in the accuracy of fitting. The former (too smooth) fails to capture the actual dynamics of state functions while the latter induces extra dynamics that are mainly caused by the noise in the original data. Varah tuned this trade-off by adaptively adjusting the number and position of the knots of cubic splines, and the user (himself) served as a judge to determine if the interactive graphics achieved a satisfiable smoothing level.

Ramsay and Silverman (2005) introduced the principal differential analysis (PDA) method which was also a two-step algorithm but improved Varah's spline least square method in two ways. Firstly, a more generalized process was designed to address the curve fitting trade-off. In particular, they introduced a second-order derivative

penalty (curvature penalty) besides the SSE:

$$\sum_{j=1}^{n} ||\boldsymbol{y}_j - \hat{\boldsymbol{x}}(t_j)||^2 + \lambda \cdot \sum_{i=1}^{d} \int \left(\frac{d^2 \hat{x}_i(t)}{dt^2} \right)^2 dt,$$

where λ is the smoothness parameter that controls the size of the influence of curvature penalty. The basis coefficient matrix Γ is obtained by minimizing the objective function above. The second order derivative controls the smoothness of the fitted curve hence prevents the profiling being either too smooth or too fluctuating. It's a more data-driven process than a human judge so future methods mostly adopt Ramsay and Silverman's idea. Secondly, instead of minimizing the sum of squares of the discrepancy between derivatives of fitted functional data object and ODE equations, PDA minimizes the integral:

$$\sum_{i=1}^{d} \int \left[\frac{d\hat{x}_i(t)}{dt} - F_i(\hat{\boldsymbol{x}}(t), \boldsymbol{u}(t), \Theta) \right]^2 dt. \tag{1.1}$$

Although this method improved Varah's two-step method, it also suffered from poor accuracy due to the heavy dependence on curve fitting procedure.

Poyton *et al.* (2006) stepped further and proposed the iterative principal differential analysis (iPDA). The iPDA iterates between data profiling and parameter estimation in PDA. It incorporates the object function of parameter estimation Equation (1.1) into the object function of curve fitting, then updates both spline basis coefficients and model parameter set Θ until convergence. We will discuss iPDA in detail in Chapter 3.

Ramsay *et al.* (2007) combined both the idea of NLS and PDA, and proposed the generalized smoothing approach. The iPDA treats the structural parameter Θ and the nuisance parameter Γ equally and selects the best combination of Θ and Γ while

Ramsay's generalized smoothing approach expresses Γ as a function of Θ and aims at Θ directly. This approach smooths the raw data using basis splines with a more generalized penalty, hence avoids solving ODEs numerically. The spline coefficient Γ is estimated by minimizing

$$J(\Gamma|\Theta,\lambda) = -\sum_{j=1}^{n} \ln g(\boldsymbol{e}_j|\Theta,\lambda) + \sum_{i=1}^{d} \lambda_i \int \left[\frac{d\hat{x}_i(t)}{dt} - F_i(\hat{\boldsymbol{x}}(t), \boldsymbol{u}(t), \Theta)\right]^2 dt,$$

where

$$e_{ij} = y_{ij} - \hat{\Gamma}_{\cdot i}(\Theta;\lambda)'\boldsymbol{\phi}(t_j)$$

is the residual of ith state function at t_j, g is the likelihood function, and the penalty term assesses the fidelity to the ordinary differential equations. In the least square and linear ODE case it is possible to express $\hat{\Gamma}$ analytically as a function of Θ. To estimate model parameter Θ Ramsay took a negative log likelihood criterion:

$$H(\Theta|\lambda) = -\sum_{j=1}^{n} \ln g(\boldsymbol{e}_j|\Theta,\lambda).$$

If normal assumption is made for the distribution of errors, this criterion is simply the sum of squares of the discrepancies as in NLS method. Then the gradient of H with respect to Θ is computed to be set up for an optimization scheme. When $\hat{\Gamma}$ cannot be expressed explicitly as a function of Θ, the implicit function theorem can be applied to $\partial J/\partial\Gamma$ to obtain the gradient. In the description above, λ is assumed to be fixed, while in real applications all the parameters can be viewed as a function of λ, and it can be selected by some measure of model complexity such as generalized cross-validation.

All the methods above suffered from a common drawback that they failed to work in high dimensional ODEs. The performance of two-step methods depends heavily on the accuracy of data profiling, because a small turbulence in data fitting would cause a huge bias in the estimation of its derivative. Two-step methods can only provide reasonable-but-not-accurate solution in low-dimension case, it's difficult to obtain even close to appropriate estimates in high dimension. The iterative methods cannot improve much on the accuracy of data profiling and they suffer another problem - the computational burden. In this dissertation we aim to build a tractable model as well as efficient algorithms to tackle high dimensional dynamic systems. No one has proposed any reliable method to deal with high dimensional ODEs because of the difficulty of the problem.

1.2 ECoG Time Series: the Motivation

The second part of the introduction focuses on a real problem of estimating parameters of ODEs. We will briefly introduce the history of connectivity study and mainly discuss the characteristics of ECoG data set. It's the uniqueness and advantage of ECoG time series that drives us to propose a simplified dynamic causal model for which our algorithms are designed.

The human brain does not function like isolated islands. It's quite the opposite, neurons interact with each other through afferent and efferent connections within an elaborately collaborative system to ensure neurons and neural networks process information, perform simple and complicated tasks. A useful nominal taxonomy of brain connectivity relies on three broad categories: anatomical, functional and effective. Anatomical refers to the network architecture, whereas functional and effective connectivity refer to network engagement. By Friston, functional connectivity is

defined as the "temporal correlations between spatially remote neurophysiological events" (Friston *et al.*, 1993a). Integration within a distributed system is better understood through effective connectivity. Effective connectivity is defined as "the influence that one neural system exerts over another either directly or indirectly" (Friston *et al.*, 1993b). Functional connectivity does not explain how neuronal events correlates with each other (for example, Gerstein, 1969), it's simply description of correlations. While it is effective connectivity that examines the interactions and we are trying to study at through ECoG data.

1.2.1 Overview of Neuroimaging Data

In this dissertation we will use the Electrocorticography (ECoG) time series data to study effective connectivity. But before introducing ECoG let's review other data sets first. The data sets (or more precisely, the techniques) neuroscientists have worked the most with are electroencephalography (EEG), magnetoencephalography (MEG), functional magnetic resonance imaging (fMRI) and positron emission tomography (PET). Those data (techniques) all measure the activity of a large group of neurons, rather than the behavior of a single neuron. The difference between those data sets lays in the temporal and spatial resolution they can achieve, as well as what subject they attempt to measure. In general, fMRI and PET measures brain activity by detecting changes in blood flow with corresponding technology, yielding relatively high spatial resolution, but limited temporal resolution due to the slower rate of brain hemodynamics. In contrast, EEG and MEG measure electrical and magnetic activity in the brain, hence they provide high temporal resolution in the order of milliseconds but might not be able to obtain a high spatial resolution. To understand better the difference between different types of data, we briefly intro-

duce how they are collected. By Friston (1994), "the fMRI data were a time-series of 64 gradient-echo EPI coronal slices (5 mm thick, with 64×64 voxels $2.5 \times 2.5 \times 5$ mm) through the calcarine sulcus and extrastriate areas. Images were obtained every 3 seconds from a normal male subject using a 4.0T whole body system, fitted with a small (27 cm diameter) z-gradient coil (TE 25 ms, acquisition time 41 ms). Photic stimulation (at 16 Hz) was provided by goggles fitted with 16 light emitting diodes." Then some correction and interpolation will be applied to generate the final product. Functional MRI is most commonly applied using blood oxygenation level-dependent (BOLD) contrast (Ogawa, *et al.*, 1992). Similarly EEG is obtained by placing electrodes on the scalp with a conductive gel or paste.

1.2.2 The ECoG Time Series

ECoG involves intracranial electrophysiology recordings from subdural electrodes implanted directly on the cortical surface for clinical purposes in neurosurgical patients with medically intractable seizures or tumors. The combined high spatial (diameter 2.3 mm) resolution and temporal resolution (data collected every 1 ms) of ECoG data make it an ideal candidate for building effective connectivity models (Korzeniewska et al., 2011). Of course, it is not without its limitations, notably including the very restricted population available for study, necessarily low subject sample sizes and subject-dependent and varying electrode placement lo- cations, all impacting the generalizability of ECoG results. Nonetheless, for studying effective connectivity, ECoG offers a unique complement to traditional scalp EEG recordings methods (see Bressler and Ding, 2002; Boatman-Reich et al., 2010, for a detailed comparison between EEG and ECoG).

Effective connectivity is usually characterized by a model on the dynamic inter-

actions between brain components (Aertsen and Preissl, 1991; Friston *et al.*, 2004), and the most commonly used models include Structural Equation Models (SEM, McIntosh and Gonzalez-Lima, 1994) and the closely related Dynamic Causal Models (DCM, Friston *et al.*, 2003). Here we use a model that can be thought of as a special case of DCM, as it attempts to describe the biophysical mechanism of the brain system building from the neuronal level. In contrast, most standard applications of SEM evaluate connection strength based on the variance-covariance structure of the observed data.

A DCM requires two parts: (1) neuronal state equations consisting of a set of ordinary differential equations (ODE), which describe how instantaneous changes of the neuronal activities of system components are modulated jointly by the immediate states of the components and experimental inputs; and (2) observation equations linking the underlying neuronal states of brain components to the observed data. A DCM can be viewed as a continuous time state-space model, parameterizing effective connectivity as coupling between the neuronal states of the brain system under the influence of experimental inputs.

Although DCM has been widely used in brain connectivity research, existing implementations, primarily within the setting of fMRI, EEG and MEG data (Friston et al., 2003; David and Friston, 2003; David et al., 2006; Kiebel et al., 2006; Daunizeau et al., 2009), have two major complications. First, parameter estimation of the DCM is computationally difficult, due to the complicated model formulation. Thus the number of brain regions included in the model is usually limited. Second, identifiability issues can arise, even with only a moderate number of brain components. The current practice in addressing this problem is to conduct Bayesian inference, using a highly informative prior, introducing subjective knowledge of the existence and strength of connections, and thus imposing regularization on the coupled dy-

namic system. However, a strong prior increases the risk of bias, raising concerns on the reliability of the results. These drawbacks are alleviated in ECoG, which has high temporal and spatial resolution, and a strong signal-to-noise ratio (SNR), to evaluate the effective connectivity among many brain regions. We propose a new ODE-based model, hinging on the unique properties of ECoG data, and develop efficient methods to estimate the model. We refer to this model as a dynamic directional model (DDM) to delineate from the general DCM and to avoid confusion with the widely used Rubin Causal Model (Rubin, 1974, 1978; Holland, 1986). Though we use ECoG as an application of the proposed methods, we note that they have potential applicability in many other network studies where multivariate time series data measuring temporal changes of system components are collected, and the focus is on investigating directional interactions among them.

Anatomical and functional connections between brain regions are commonly believed to be biologically expensive, as they take up space and consume energy (Foldiak and Young, 1995; Olshausen and Field, 2004; Anderson, 2005). Therefore, it is reasonable to assume that connections between the components of a complex brain system are sparse (Bullmore and Sporns, 2009; Micheloyannis, 2012). Sparsely-connected brain networks can arise in different forms, and we here focus on the one that is decomposable into several functionally-segregated subnetworks/modules, a network structure called modularity and most relevant to brain organization (Tononi et al., 1994; Newman, 2004). A main thrust of this dissertation is to propose a modified DDM by using the Potts model (Potts, 1952; Graner and Glazier, 1992) - called Potts-based DDM (PDDM) - for the ODE parameters in the state equations for characterizing modularity.

To solve for the proposed PDDM, the log-likelihood-based criterion proposed by Varah (1982) and Ramsay *et al.* (2007) is adopted, in which neuronal state and

observation equations are combined into one formula. The time-varying neuronal states of brain components are represented by spline bases. The parameters for the state and observation equations are estimated simultaneously by optimizing the log-likelihood criterion. To achieve sparsity, we employ a popular penalized likelihood approach in regression analysis (for a review, see Shao, 1998; Fan and Lv, 2010). This is intuitive, since the ODEs can be viewed as a set of special regression models, where temporal functions of neuronal states are predictors and their derivatives are the responses. In particular, we convert the Potts model to a Potts penalty - a new penalty in the literature - to penalize large modules, and identify small functionally segregated subnetworks by minimizing the penalized criterion.

The proposed PDDM for the ECoG data is a special ODE model. There is an extensive statistical literature on solving ODEs as discussed in Section 1.1. However, these methods are mostly effective for low-dimensional cases, and are not directly applicable to the ECoG data because either the model is highly case-specific or the associated computation is too expensive. By decomposing high-dimensional differential equations into several independent low-dimensional ones using the Potts penalty, we greatly reduce the computational demand and increase estimation efficiency. As such, besides advancing the scientific research in effective brain connectivity, this dissertation also contributes to statistical methodology for inference of high-dimensional ODEs.

1.3 Outline of Dissertation

The dissertation will address the intractability of high dimensional ODE problems by proposing a new dynamic model specifically designed for the ECoG time series, and also building an efficient algorithm to estimate the model. The rest of the

dissertation is organized as follows.

In Chapter 2 we introduce the Potts-based DDM based on which the following chapters rest upon. In this chapter we will start with discussing general dynamic causal model, its formal definition as well as scientific explanation. Then we define DDM which encounters huge computational unwieldiness. To tackle the computational issue and also deliver neuroscientific explanation for simple brain function, we propose the Potts-based DDM.

Even with a sparse model Potts-based DDM, estimation remains a difficult task because the Potts penalty is essentially an L_0 penalty, which makes optimal solution searching an NP hard problem. In Chapter 3 we demonstrate our strategy to tackle the estimation of Potts-based DDM. Although commonly used method like iterative principal differential analysis (iPDA) fails to produce meaningful estimates for high-dimensional ODEs, it is the cornerstone of our algorithm and we will introduce iPDA in detail in this chapter. Two new algorithms, lasso-based iPDA (L-PDA) and Potts-based iPDA (P-iPDA) improve iPDA by adding penalty terms to original log-likelihood, while different penalty structures favor different network structures. The main algorithm we want to present in this dissertation is the P-iPDA which searches for local optimum within neighborhood of current network structure in each iteration.

Chapter 4 focuses on computational issues of iPDA, L-iPDA, and P-iPDA. The computation procedures are similar for the three algorithms, all of them consist two step iterations: (1) data profiling, and (2) ODE parameter estimation. They share the data profiling step but differ in ODE parameter estimation step. The parameter estimation step boils down to regression analysis. So the ODE parameter estimation for iPDA is simply OLS, and the one for L-iPDA is equivalent to lasso regression. Parameter estimation for P-iPDA is a little more complicated (conceptually, but

not computationally) since it has to search for the best cluster among all possible network structures that are defined as "neighbors" of current network structure.

Chapter 5 applies P-iPDA to ECoG time series. Because of the size ($d = 45$) of ECoG data set, it's extremely time-consuming to apply iPDA or L-iPDA, so only P-iPDA is reported in this dissertation. The P-iPDA is able to detect auditory related channels and cluster them in the same module.

Chapter 6 applies all three algorithms to three simulation examples. The first example is a 4-dimensional example demonstrating how the algorithms work, it's also used to discussed some common issues across all examples. Other examples are: a network with 32 nodes and 2 clusters (Example 2), a network with 20 nodes and 4 clusters (Example 3 and 4).

Chapter 2

Directional Dynamic Model

In this section we will first review dynamic causal models for hemodynamic and electromagnetic time series. The high quality of the ECoG time series enables us to introduce the dynamic directional model which can be viewed as a special case of DCM - a bilinear approximation of neuronal equations and direct simplification on observation equations. The Potts-based DDM will then be proposed by taking into account subnetwork structure, upon which the rest of the dissertation lays.

2.1 General Dynamic Causal Model

The basic idea of DCM is to construct a reasonably realistic neuronal model of coupling cortical regions. A forward model then transforms the underlying neuronal or synaptic activity to observable response. These supplement models may be hemodynamic models of fMRI time series or forward models of electromagnetic measurements. In this section we will mainly use fMRI as an example to illustrate DCM. The essence of DCM is to model brain as a deterministic nonlinear dynamic system which takes both stimulus-free and stimulus-bound inputs and pro-

duces outputs. Because human brain is a continuous-time physicochemical system changing rapidly over time, it is intuitive to model the dynamics by characterizing its instantaneous changes, or, mathematically speaking, first order derivatives. Among all the ODE-based dynamic models, the one with the Markovian property that the instantaneous change only depends on the current neuronal state and experimental/stimulus inputs has the least model complexity without sacrificing much scientific meaning in our problem, because when performing a simple task such as visual, auditory and motor functions in a short period of time, it is reasonable to assume Markovian property. However for a complicated function like memory, it might not be appropriate to use such a simple model.

In DCM effective connectivity is parameterized in terms of coupling/interaction among unobserved neuronal activity in different regions based on the ODE model, which distinguishes from established approaches using multivariate autoregressive processes. Another critical difference is that in conventional methods there is no designed perturbation and inputs are treated as unknown and stochastic, while in DCM the experimentally designed inputs are deterministic and controlled by researchers. In DCM experimental designed inputs can affect the dynamics in two ways: they can produce responses directly in the neuronal state functions, or they can change the effective connectivities. We borrow a simple example (Figure 2.1) from Friston (Friston *et al.*, 2003) to illustrate the mechanism of underlying dynamic causal modeling. We have introduced DCM briefly in Chapter 1, but let's now express DCM more rigorously (Daunizeau *et al.*, 2009). It usually contains two parts:

- a set of ordinary differential equations describing hidden neuronal states (so-

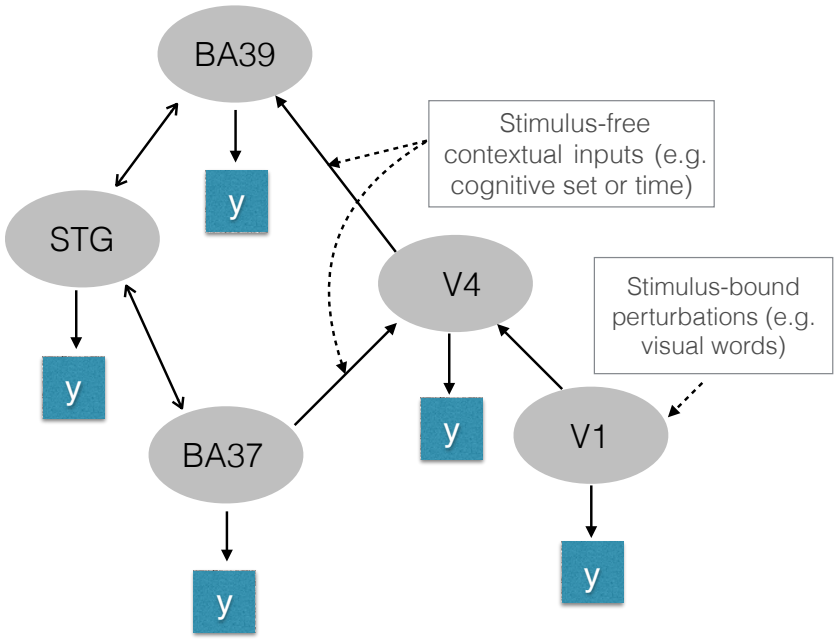


Figure 2.1: This is a schematic borrowed from Friston (2003) to illustrate the scheme of DCM through a simple example. Inputs can impact responses both directly and through changing coupling dynamics. In this example there are five neuronal states, including visual areas **V1** and **V4** in the fusiform gyrus, areas 39 and 37, and the superior temporal gyrus **STG**. Stimulus-bound perturbations act as extrinsic inputs to the primary visual area **V1**. Stimulus-free or contextual inputs mediate their effects by modulating the coupling between **V4** and **BA39** and between **BA37** and **V4**. The blue square boxes represent the components of the DCM that transform the neuronal state functions in each region into measured (hemodynamic) response y_i's.

called *evolution equations* or *neuronal state equations*):

$$\frac{d\boldsymbol{x}(t)}{dt} = F_1(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\theta}_1),$$

where $\boldsymbol{x}(t) = (x_1(t), ..., x_d(t))$ is the unobserved time-dependent neuronal state vector of d brain regions, $d\boldsymbol{x}(t)/dt$ is the instantaneous rate of change of the system's state with respect to time, $\boldsymbol{u}(t)$ is the known experimental input function vector, $\boldsymbol{\theta}_1$ is a set of unknown evolution parameters, and F_1 is a vector of functions that specifies the biophysical mechanism based on which the underlying neuronal states behave.

- an *observation equation* (in fMRI case this is the hemodynamic state equation) describing how experimental measures are determined by underlying hidden neuronal states $\boldsymbol{x}(t)$:

$$\boldsymbol{y}(t) = F_2(\boldsymbol{x}(t), \boldsymbol{\epsilon}(t), \boldsymbol{\theta}_2),$$

where $\boldsymbol{y}(t) = (y_1(t), ...y_l(t))$ is the observed response function vector (usually only available at a fixed number of time points $t_1, \ldots, t_n$), $\boldsymbol{\epsilon}(t)$ is the error term, $\boldsymbol{\theta}_2$ is another set of observation parameters, and F_2 is another vector of functions that link unobserved neuronal states with experimental measurements.

The formulation of F_1 and F_2 depends on the targeted imaging modality. For fMRI data, the neuronal states equations in F_1 are usually approximated by the first and part of the second order Taylor expansions, with a bilinear form that will be specified later (Equation 2.1). Strong restrictions are imposed on the parameter space to ensure that the underlying system is stable over time (hundreds of seconds).

The observation equations F_2 include several differential equations translating neuronal activity into hemodynamic responses. Hemodynamic states are a function of the underlying neuronal states and are irrelevant to other variables. Some specific examples of hemodynamic state equations can be found elsewhere (Friston *et al.*, 2000). The estimation of parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ is done by a Bayesian approach with highly informative priors. This approach only works if the number of brain regions involved is low.

2.2 The DDM for ECoG Time Series

ECoG signals are recorded from electrodes implanted directly over the cortical surface of the brain. Electrodes are 2-3 mm in diameter and evenly spaced at 10 mm, center-to-center, in 6 or 8×8 arrays. The ECoG time series are recorded from all electrodes simultaneously. ECoG recordings are characterized by high signal-to-noise ratios and excellent temporal (1-2 ms) and spatial (10 mm) resolution. ECoG recordings from human auditory cortex have been shown recently to be highly reliable and reproducible (Cervenka *et al.*, 2013). Resting on the unique properties of the ECoG time series, we propose a new dynamic directional model (DDM) with a simple formulation to evaluate the effective connectivity among many brain regions.

The ECoG data has millisecond temporal resolution which allows one to approximate the nonlinear neuronal dynamic F_1 by its first order and part of second order Taylor expansions:

$$\frac{d\boldsymbol{x}(t)}{dt} = A\boldsymbol{x}(t) + \sum_{j=1}^{J} B^j u_j(t)\boldsymbol{x}(t) + C\boldsymbol{u}(t) + D, \tag{2.1}$$

where $A = (A_{ij})_{d\times d}$ and $B^j = (B^j_{i_1 i_2})_{d\times d}$ are d by d matrices whose scientific meaning

will be explained later, note that the superscript j in B is not an exponent, it simply means the matrix interacting with the jth stimulus function, $C = (C_{ij})_{d\times J}$ and $D = (D_1, ..., D_d)'$ are d by 1 matrices; J is the number of experimental inputs. The matrices A, B^j's, C, and D remain constant over time.

The bilinear approximation is also natural and useful in terms of interpretability of effective connectivity. The Jacobian or connectivity matrix A represents the first-order connectivity among the regions in the absence of input $\boldsymbol{u}$. While as the jth input is introduced by the experimenter, the coupling among brain regions is perturbed by the amount of $B^j u_j$. When all the inputs are presented we can represent the connectivity matrix as $A+\sum u_j B^j$. The matrix C gauges the extrinsic influence that is completely induced by inputs not through changing the coupling of neuronal states. Finally the matrix D serves as a constant effect just like intercept terms in regression models. In this dissertation, we will name A the *connectivity matrix*, B^j the *coupling matrix*, and C the *extrinsic matrix*. The parameter set $\Theta = (A, B^1, \ldots, B^J, C, D)$ defines the dynamic architecture of neuronal activity and we wish to identify them. Model (2.1) is referred to as a "directional" or "causal" model, because it specifies two separate parameters A_{ij} and A_{ji} for the effect from variable i to j and the other way around.

We use an example (Friston, 2003) to recapitulate the specific architecture in Figure 2.1 and demonstrate how matrix form of the bilinear model describes the coupling relationship among neuronal states. We label V1, V4, BA37, BA39, and STG as x_1 to x_5, and the stimulus-bound perturbation and stimulus-free input as u_1 and u_2. The following differential equation is one possible scenario of Figure 2.1:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \\ \dot{x_5} \end{bmatrix} = \left\{ \begin{bmatrix} A_{11} & 0 & 0 & 0 & 0 \\ A_{21} & A_{22} & A_{23} & 0 & 0 \\ 0 & 0 & A_{33} & 0 & A_{35} \\ 0 & A_{42} & 0 & A_{44} & A_{45} \\ 0 & 0 & A_{53} & A_{54} & A_{55} \end{bmatrix} + u_2 \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & B^2_{23} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & B^2_{42} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \right\} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} C_{11} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} u_1$$

Here A_{ij} and B^2_{ij} are the intrinsic and induced effect exerted by neuronal state function x_j upon x_i separately. Hence the connectivity matrix A (and the coupling matrix B^2) can be divided into three parts. The lower part below diagonal represents the forward effects - the effect elicited by previous neuronal state functions on later ones, the diagonal part represents the self effects - reflecting how neuronal states influence themselves, the upper part above diagonal represents the backward effects - the effect exerted by later neuronal state functions upon previous ones. B^1 is a zero matrix because the stimulus-bound input does not perturb the interaction.

Let's now turn to the second part of DCM and introduce how DDM simplifies the observation equation of DCM. Because of the high temporal resolution and highly-localized property of the ECoG data, and the fact that electrodes are placed directly over the exposed cortical surface to measure the ECoG signals, the measurements are direct observations of underlying neuronal state with random noises, i.e., F_2 is simply an identity function of neuronal activity $\boldsymbol{x}(t)$ plus error:

$$\boldsymbol{y}(t) = \boldsymbol{x}(t) + \boldsymbol{\epsilon}(t), \tag{2.2}$$

where $\boldsymbol{\epsilon}(t) = (\epsilon_1(t), ..., \epsilon_d(t))'$ is a d-dimensional vector of errors. In summary, DDM approximates the nonlinear dynamic of neuronal state equation by a bilinear form,

and assumes direct observations on underlying neuronal activity with error measurements. We need to highlight that this simplification only makes sense because of the high temporal and spatial resolution of ECoG time series, and the fact that biophysical process only refers to a simple auditory task.

2.3 Potts-based DDM for Functional-Segregated System

Even with a great simplification from DCM to DDM, it is still unreliable with very large mean square error when the number of components, d , is large for two reasons. First, though the observed data $\boldsymbol{y}(t)$ have a large signal-to-noise ratio, the estimated derivative $d\boldsymbol{x}(t)/dt$ still generates considerable amount of estimation error. This is a prevalent problem in ODE parameter estimation, since noise in observation will be amplified in derivatives. Second, to ensure the bilinear form an appropriate approximation, only very short periods of ECoG time series with only a few hundred time points were used to fit the model, which leads to a lack of data problem. The number of model parameters, $(J+1)d^2 + (J+1)d$, is in quadratic order of d, and consequently large d leads to large number of parameters, which makes variance uncontrollable given the limited data. This problem can be addressed by imposing sparsity assumptions on the parameters. Such sparsity-inducing regularization has been widely used in regression problems with a large number of candidate predictors (for a review, see Shao, 1998; Fan and Lv 2010). Since given the observation $\boldsymbol{x}(t)$, the parameter estimation for the DDM model (2.1) is equivalent to solving d linear regression models (this will be explained in detail in Chapter 3), hence the sparsity assumption would likewise help to increase the efficiency of the DDM.

Apart from statistical and computational points of view, more importantly, the sparsity assumption also has a scientific motivation. The connections among brain regions are energy consuming (Foldiak and Young, 1995; Olshausen and Field, 2004; Andersen, 2005), and biological units tend to minimize energy-consuming activities unless necessary in order to survive and propagate (Bullmore and Sporns, 2009; Micheloyannis, 2012). As such, it is scientifically reasonable to believe that connections among many brain components should be sparse, especially when brain is performing a simple function within a short period of time. Among all the possible sparse network structures, we focus on the one that is decomposable into several functionally-segregated subnetworks, a structure known as *modularity*, for two reasons. First, modularity has been widely reported in the literature on brain networks (Girvan and Newman, 2002; Milo *et al.*, 2002; Newman, 2003; Milo *et al.*, 2004; Newman, 2006). The modules form "building blocks" for large network systems and the "the modularity of the brain's architecture" "effectively insulates functionally bound subsystems form spreading perturbations due to small fluctuations in structure or dynamics" (Sporns, 2011). Second, the modularity structure provides intuitive interpretation of independent functions of brain regions in different modules, and support functional segregation and specialization, an important principle of brain functional organization (Friston *et al.*, 2004; Sporns, 2013). Note that in this dissertation, we use subnetworks, modules and clusters interchangeably.

Towards this goal, we propose a new model for neuronal states. We assume that under a perfect functional-segregated system, all neuronal states can be separated into modules. Figure 2.2 shows a very simple 3 module example. Let m_i be the module indicator of the ith system component, which can take integer value from 1 to G. $G < d$ is the number of modules in total. And let $\delta(m_i, m_j)$ be the Kronecker delta which equals 1 when $m_i = m_j$ and 0 otherwise. For system component i we

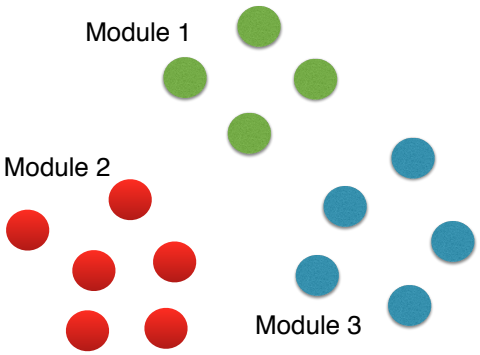


Figure 2.2: A three-module example: In this 15-dimensional network, three modules are independent with each other while neuronal states collaborates closely within each module.

assume the following model, a generalization of Model (2.1):

$$\frac{dx_{i_1}(t)}{dt} = \sum_{i_2=1}^{d} \delta(m_{i_1}, m_{i_2}) \cdot A_{i_1 i_2} \cdot x_{i_2}(t) + \sum_{j=1}^{J} u_j(t) \sum_{i_2=1}^{d} \delta(m_{i_1}, m_{i_2}) \cdot B^j_{i_1 i_2} \cdot x_{i_2}(t) + \sum_{j=1}^{J} C_{i_1 j} \cdot u_j(t) + D_{i_1}. \quad (2.3)$$

We further assume the Potts model on the module label m_i:

$$\mathcal{P}(m_1, ..., m_d) \propto \exp\{-\mu \cdot \sum_{i_1, i_2=1}^{d} \delta(m_{i_1}, m_{i_2})\}, \quad (2.4)$$

where $\mathcal{P}$ denotes a probability measure, and μ is a positive constant. Equation (2.3) and (2.4) are referred to as the Potts-based DDM (PDDM). From Model (2.3) it is clear that the larger a module is, the more parameters are required to characterize the dynamic system, and the increase is quadratic of the size of the cluster. Specifi-

cally, the most energy-intensive system is the one with all the components grouped in the same cluster, as in Model (2.1), and the most parsimonious model is the one with d clusters in each of which there is only one component. This is opposite to the clustering scenario, where elements in the same cluster are assumed to have the same distribution, hence larger cluster contains fewer model parameter, and is considered to be more parsimonious. The Potts model (2.4) assumes large probability for systems with many small functionally-independent systems, and small probability for systems with large cluster. Hence when estimating the system structure, we favor small clusters unless we can improve the performance significantly.

Chapter 3

Algorithms to Estimate DDM

In this chapter we first introduce a widely used log-likelihood based method iterative principal component analysis (iPDA) to estimate the parameters $\Theta = (A, B^1, \ldots, B^J, C, D)$ in the DDM (Equation 2.1). This algorithm will also be the basic component of other algorithms designed for sparsely-connected systems. However iPDA fails to work even for a moderate number of neuronal state functions with or without the sparsity assumption. We then propose two new algorithms designed to learn two different sparse systems - a functional segregated system for Potts model and a sparsely connected but inseparable system for comparison. The strategy of selecting the roughness penalty and sparsity penalty will also be discussed in this section.

To clarify the notations, we list all the symbols that will appear in this Chapter in the following table, and we want to explicitly mention that we use prime ' to denote the transpose of a matrix, d/dt to denote the derivative of a function, and hat $\hat{}$ to denote the estimated value. And when we speak of the underlying neuronal states, we use system component, node, neuronal state functional interchangeably.

Table 3.1: List of symbols

J	Number of exogenous inputs (experimental stimulus)
d	Number of neuronal state functions
p	Number of bases used to smooth the data
n	Number of observations
$\boldsymbol{x}(t)$	Actual and unobserved neuronal states
$\boldsymbol{y}(t)$	Observed neuronal states
$\boldsymbol{u}(t)$	Exogenous inputs
$\boldsymbol{\phi}(t)$	Basis functions
Γ	Basis coefficient matrix
Θ	ODE model parameter (A, B, C, D)
Θ_i	The ith row of Θ
λ	Roughness penalty coefficient
μ	Sparsity penalty coefficient
H,**HP**,**HL**	Optimization criterion of iPDA, P-iPDA and L-iPDA
Fid	ODE model penalty, the fidelity to the ODE equations
$\mathbf{L}, \mathbf{LP}$	Linear operator
$\mathbf{H}_{\mathrm{P}}$	SSE + Fid in P-iPDA
$\mathbf{R}_{\mathrm{P}}$	Fid + Potts penalty
$\mathbf{R}_{\mathrm{L}}$	Fid + L_1 penalty
m_i	Module label of neuronal state function i
M	Module labels of all variables
I_i	Indices of nodes that are in the same module as variable i
G	Number of modules
$\mathcal{N}$	Neighborhood of a network structure

3.1 Log-likelihood Based Criteria

Given system state vector $\boldsymbol{x}(t)$, we will later show that estimating model parameters Θ can be converted to a linear regression problem which can be easily solved. So fitting system states $\boldsymbol{x}(t)$ becomes especially important to parameter estimation. However, fitting $\boldsymbol{x}(t)$ especially $d\boldsymbol{x}(t)/dt$ requires carefully designed methods. There are two popular approaches in the literature for ODE fitting: discretization methods using numerical approximations and basis function expansion as discussed in Chapter 1. Here we adopt the latter and approximate $\boldsymbol{x}(t)$ by a set of cubic B-spline bases:

$$\boldsymbol{x}(t) \approx \Gamma' \boldsymbol{\phi}(t), \tag{3.1}$$

where $\boldsymbol{\phi}(t) = (\phi_1(t), ..., \phi_p(t))'$ represents the p-dimensional spline bases, and Γ is a p by d matrix. The ith column of Γ, denoted by $\Gamma_{\cdot i}$, represents the coefficients of spline basis expansion of $x_i(t)$. The estimation function

$$\hat{x}_i(t) = \hat{\Gamma}'_{\cdot i} \boldsymbol{\phi}(t)$$

is determined by selecting spline-weighting coefficients $\hat{\Gamma}$ to minimize the SSE at the observation times t_j:

$$\hat{\Gamma} = \underset{\Gamma}{\operatorname{argmin}} \sum_{i=1}^{d} \sum_{j=1}^{n} (y_i(t_j) - \Gamma'_{\cdot i} \boldsymbol{\phi}(t_j))^2,$$

which can also be thought as minimizing negative log-likelihood.

In order to prevent the fitting from being too smooth (so that it does not capture the detailed dynamics) or being not smooth enough (overfitting), a roughness

penalty term and a smoothness coefficient used to control the amount of roughness penalty are used to balance this tradeoff. Varah (1982) tuned the number and position of knots by hand until he was satisfied with the smoothing. This method does not work well when dimension of data is high. To let the data tell, another way of controlling the amount of smoothing is to add a penalty on higher-order derivatives of the splines (Ramsay, 2005). A second-order derivative (curvature) penalty is widely used in general situations. Under this scenario $\hat{\Gamma}$ will be determined by:

$$\hat{\Gamma} = \underset{\Gamma}{\operatorname{argmin}} \left[\sum_{i=1}^{d} \sum_{j=1}^{n} (y_i(t_j) - \Gamma'_{\cdot i}\boldsymbol{\phi}(t_j))^2 + \lambda \sum_{i=1}^{d} \int \left(\frac{d^2 x_i(t)}{dt^2} \right)^2 dt \right],$$

where

$$\frac{d^2 x_i(t)}{dt^2} \approx \Gamma'_{\cdot i} \frac{d^2 \boldsymbol{\phi}(t)}{dt^2}$$

and λ is the smoothness coefficient: small λ puts large weight on data fitting so it will tend to fit the error by fluctuating as closely as possible towards the observations $\boldsymbol{y}(t)$, while big λ overly smooths data so actual dynamics may be lost.

Ramsay (1996) proposed principal differential analysis (PDA) to replace curvature penalty by ODE model-based penalty: so that not only the variance of the estimator is controlled but also deviation from underlying ODE model is penalized. This deviation is captured by the fidelity to ODE models

$$\text{Fid}(\Gamma, \Theta) = \sum_{i=1}^{d} \int \left(\Gamma'_{\cdot i} \frac{d\boldsymbol{\phi}(t)}{dt} - \mathbf{L}_i(\Gamma' \boldsymbol{\phi}(t), \boldsymbol{u}(t), \Theta_i) \right)^2 dt, \tag{3.2}$$

where $\mathbf{L}_i$ is a linear operator on $\boldsymbol{x}(t) = \Gamma' \boldsymbol{\phi}(t)$ and $\boldsymbol{u}(t)$ which describes the dynamics

on $x_j(t)$ exerted by $\boldsymbol{x}(t)$:

$$\mathbf{L}_i(\Gamma'\boldsymbol{\phi}(t), \boldsymbol{u}(t), \Theta_i) = A_{i\cdot}\boldsymbol{x}(t) + \sum_j u_j(t)B^j_{i\cdot}\boldsymbol{x}(t) + C_{i\cdot}\boldsymbol{u}(t) + D_i,$$

in which $A_{i\cdot}$, $B^j_{i\cdot}$, $C_{i\cdot}$, and D_i denote the ith row of A, B^j, C and D respectively and $\Theta_i = (A_{i\cdot}, B^1_{i\cdot}, ..., B^J_{i\cdot}, C_{i\cdot}, D_i)'$ is a $(d + Jd + J + 1) \times 1$ vector of unknown model parameters. Formally PDA estimates Γ by minimizing the following criterion:

$$\mathbf{H}(\Gamma, \Theta) = \sum_{i=1}^{d}\sum_{j=1}^{n}(y_i(t_j) - \Gamma'_{\cdot i}\boldsymbol{\phi}(t_j))^2 + \lambda \cdot \mathrm{Fid}(\Gamma, \Theta). \tag{3.3}$$

Note that although we write $\mathbf{H}$ as a function of both Γ and Θ in Equation (3.3), Θ is treated as known in PDA, i.e., PDA assumes a known underlying dynamic and the problem is just to fit $\boldsymbol{x}(t)$ by estimating Γ.

But our problem is different from what PDA targets. Our main concern is to estimate the ODE parameter Θ which is treated known in PDA. By representing $\boldsymbol{x}(t)$ as basis expansions, we create an extra set of unknown parameters Γ (nuisance parameter) besides our target parameter Θ (structural parameter). Estimating Γ and Θ simultaneously in Equation (3.3) will be unrealistic, Poyton *et al.* (2006) proposed an iteratively PDA (iPDA) strategy which refines the PDA by iterating between two steps, estimating basis coefficients Γ (holding Θ fixed) and ODE model parameters Θ (holding Γ fixed). This algorithm is the cornerstone of our algorithm and can be summarized in Figure 3.1.

We make three notes for the algorithm above:

1. The initial value Θ^0 is estimated by a naive method: smoothing $\boldsymbol{y}(t)$ by penalizing the second order derivative of $d\boldsymbol{x}(t)/dt$, then estimating Θ using $d\hat{\boldsymbol{x}}(t)/dt$ and $\hat{\boldsymbol{x}}(t)$. When smoothing $\boldsymbol{y}(t)$ the amount of smoothing is selected by gen-

The iterative principal differential analysis (iPDA) algorithm:

0. Initialize $\Theta = \Theta^0$.

   At the $(s+1)$th step:

1. Data profiling. Fix $\hat{\Theta}^s$ estimated from the sth step, find $\hat{\Gamma}^{s+1}$ that minimizes $\mathbf{H}(\Gamma, \hat{\Theta}^s)$ in Equation (3.3).

2. Model parameter estimation. Fix $\hat{\Gamma}^{s+1}$ estimated from Step 1, find $\hat{\Theta}^{s+1}$ that minimizes $\text{Fid}(\hat{\Gamma}^{s+1}, \Theta)$. Then update the value of $\mathbf{H}(\Gamma, \Theta)$.

3. Iterate (1) and (2) until the relative change of $\mathbf{H}(\Gamma, \Theta)$ is less than a prespecified threshold.

Figure 3.1: The iterative principal differential analysis (iPDA) algorithm.

eralized cross-validation (GCV).

2. In the algorithm above and later, we assume that the parameter that controls the penalty λ (and later μ) are fixed. How to select λ (and μ) in step 1 will be discussed in Section 3.4.

3. We will see in Chapter 4 that $\text{Fid}(\Gamma, \Theta)$ can be written as a quadratic function of Θ hence Θ can be estimated by multiple linear regressions with response $d\hat{x}_i/dt$ and regressor $(\boldsymbol{x}(t), \boldsymbol{u}(t)\boldsymbol{x}(t), \boldsymbol{u}(t), \mathbf{1})$.

The above algorithm works well for a simple dynamic system ($d < 5$), but its performance for a system with even a moderate size (e.g., $d = 10$) can be questionable. As discussed in Tibshirani (1996), variability of parameter estimates of regression models can be reduced by shrinking some parameter, which can be achieved via penalized likelihood approaches. Below we introduce two penalty terms upon which two algorithms are based.

3.2 Estimation with the Potts Penalty

Taking into account the Potts model (2.4), we propose a Potts penalty (PP)

$$PP(M) = \sum_{i,i_2=1}^{d} \delta(m_i, m_{i_2}) \tag{3.4}$$

and include it in the log-likelihood criterion (3.3) we define

$$\begin{aligned}
\mathbf{HP}(\Gamma, \Theta) =& \mathrm{SSE}(\Gamma) + \lambda \cdot \mathrm{Fid}_{\mathrm{P}}(\Gamma, \Theta) + \lambda \cdot \mu \cdot \mathrm{PP}(M) \\
=& \sum_{i=1}^{d} \sum_{j=1}^{n} (y_i(t_j) - \Gamma'_{\cdot i} \boldsymbol{\phi}(t_j))^2 \\
&+ \lambda \cdot \sum_{i=1}^{d} \int \left(\Gamma'_{\cdot i} \frac{d\boldsymbol{\phi}(t)}{dt} - \mathbf{LP}_i(\Gamma' \boldsymbol{\phi}(t), \boldsymbol{u}(t), \Theta_i) \right)^2 dt \\
&+ \lambda \cdot \mu \cdot \sum_{i,i_2=1}^{d} \delta(m_i, m_{i_2}).
\end{aligned} \tag{3.5}$$

We want to emphasize the difference between Equation (3.3) and (3.5) so we add a subscript "P" which stands for Potts model. Define $\mathbf{H}_{\mathrm{P}}(\Gamma, \Theta)$ as a modified $\mathbf{H}(\Gamma, \Theta)$ with the second term changed according to (2.3):

$$\begin{aligned}
\mathbf{H}_{\mathrm{P}}(\Gamma, \Theta) =& \sum_{i=1}^{d} \left[\sum_{j=1}^{n} (y_i(t_j) - \Gamma'_{\cdot i} \boldsymbol{\phi}(t_j))^2 \right. \\
&\left. + \lambda \cdot \int \left(\Gamma'_{\cdot i} \frac{d\boldsymbol{\phi}(t)}{dt} - \mathbf{LP}_i(\Gamma' \boldsymbol{\phi}(t), \boldsymbol{u}(t), \Theta_i) \right)^2 dt \right],
\end{aligned} \tag{3.6}$$

where

$$\mathbf{LP}_i(\Gamma'\boldsymbol{\phi}(t), \boldsymbol{u}(t), \Theta_i) = \sum_{i_2=1}^{d} \delta(m_i, m_{i_2}) \cdot A_{ii_2} \cdot x_{i_2}(t) + \sum_{j=1}^{J} u_j(t) \sum_{i_2=1}^{d} \delta(m_i, m_{i_2}) \cdot B^j_{ii_2} \cdot x_i(t) + \sum_{j=1}^{J} C_{ij} \cdot u_j(t) + D_i.$$

There is a little notation inconsistency here with equation (2.3). We drop the subscription for i_1 for simplicity. We define a few module-related notations to further simplify the Potts model penalty. Let I_i denote the set of variable indices (from smallest to largest) whose corresponding variables are in the same module as $x_i(t)$. For example, if neuronal states x_1, x_3, x_7 and x_8 are in the same module, then $I_1 = I_3 = I_7 = I_8 = (1, 3, 7, 8)$. Let A_{iI_i}, $B^j_{iI_i}$ denote the submatrices of A and B^j with corresponding row index i and column indices I_i, similar as $\boldsymbol{x}_{I_i}(t)$ Adopting this notation, we can rewrite $\mathbf{LP}_i(\Gamma'\boldsymbol{\phi}(t), \boldsymbol{u}(t), \Theta_i)$ in a more compact form:

$$\mathbf{LP}_i(\Gamma'\boldsymbol{\phi}(t), \boldsymbol{u}(t), \Theta_i) = A_{iI_i}\boldsymbol{x}_{I_i}(t) + \sum_j u_j(t) B^j_{iI_i}\boldsymbol{x}_{I_i}(t) + C_{i\cdot}\boldsymbol{u}(t) + D_i.$$

The Potts penalty (3.4) is essentially an L_0 penalty, i.e., it penalizes on the number of predictors.

Our goal is to select spline coefficient Γ, module structure M and model parameter Θ to minimize $\mathbf{HP}(\Gamma, \Theta)$. Our algorithm is based on Poyton's iPDA: we also iterate between data profiling and parameter estimation. The Potts penalty and subnetwork structure of the system enable us to propose a novel iterative scheme to detect the subnetwork hence estimate the model parameters very efficiently. Here is how it works.

In data profiling step at sth iteration, given the subnetwork structure Θ es-

timated from previous parameter estimation step (we will explain later that the parameter estimation step not only estimates Θ but also searches for an optimal network structure M), since modules are completely isolated with each other, we are able to perform data profiling like we did in iPDA within each module instead of minimizing $\mathbf{H}_{\mathrm{P}}(\Gamma, \Theta)$ globally. Let $M^s = (m_1^s, ..., m_d^s)$ be the collection of module indicators of all d system components at step s, and suppose there are G^s modules at current step (without loss of generality, we always index modules from 1 to G^s, hence $G^s = \max(M^s)$). We can rewrite $\mathbf{H}_{\mathrm{P}}$ as:

$$\begin{aligned}\mathbf{H}_{\mathrm{P}}(\Gamma, \Theta) =& \sum_{g=1}^{G^s} \Bigg[\sum_{i \in \{M^s = g\}} \sum_{j=1}^{n} (y_i(t_j) - \Gamma'_{\cdot i} \boldsymbol{\phi}(t_j))^2 \\ &+ \lambda \sum_{i \in \{M^s = g\}} \int \left(\Gamma'_{\cdot i} \frac{d\boldsymbol{\phi}(t)}{dt} - \mathbf{LP}_i(\Gamma' \boldsymbol{\phi}(t), \boldsymbol{u}(t), \Theta_i) \right)^2 dt \Bigg] \\ \triangleq& \sum_{g=1}^{G^s} \mathbf{H}_{\mathrm{P,g}}(\Gamma, \Theta), \end{aligned} \tag{3.7}$$

where $\{M^s = g\} = \{i : m_i = g, 1 \leq i \leq d\}$ is the set of variable indices whose corresponding neuronal state functions are in Module g. The term in the brackets does not contain variables that are not in Module g, hence $\mathbf{H}_{\mathrm{P}}$ can be minimized separately within each module, which tremendously lowers the computational cost especially when d is large and the size of each module is relatively small compare to d. We will discuss computational issues in Chapter 4.

In parameter estimation step, we are given estimated Γ from data profiling step and try to select Θ to minimize

$$\mathbf{R}_{\mathrm{P}}(\Gamma, \Theta) = \mathrm{Fid}(\Gamma, \Theta) + \mu \cdot \sum_{i, i_2 = 1}^{d} \delta(m_i, m_{i_2}),$$

which sums the last two terms in Equation (3.5). This is a combinatorial problem because an exhaustive search for all possible assignments of m_i and m_{i_2} is a combinatorial problem. We borrow the idea from Metropolis-Hastings algorithm, instead of searching for the global minimum, alternatively, we search for a local minimum. To be more specific, at this step we only search for the minimum among the "neighborhood" of current subnetwork structure M^s. Suppose the neighborhood only contains a few possible subnetwork structures, then we are able perform an exhaustive search without triggering too much computation burden. As long as we decrease $\mathbf{HP}(\Gamma, \Theta)$ at each iteration - it might not be the deepest descent direction - we are able to converge. The neighborhood of M^s, $\mathcal{N}(M^s)$, is defined as all the sets of module indicators that can be obtained by flipping the indicator of M^s at at most one position:

$$\mathcal{N}(M^s) = \bigcup_{i=1}^{d} \left\{ (m_1^s, ..., m_{i-1}^s, z, m_{i+1}^s, ..., m_d^s) : z = 1, ..., G^s, G^s + 1 \right\}$$

We use the following toy example to illustrate the neighborhood we just defined. Suppose at the sth step the system network structure is displayed as in Figure 2.2. At the $(s+1)th$ step, to search in the neighborhood of M^s we can flip the module label of any single variable. Suppose m_i^s, the module label of variable i, is flipped. Figure 3.2 shows all possible neighbors by only flipping m_i^s: at the sth step, neuronal state function x_i is in module 2, at the $(i+1)$th step, it could be in module 1, 2, 3, or create a new module by itself.

Similarly as we rewrite $\mathbf{H}_{\mathrm{P}}$ based on subnetwork structure, we can also rewrite

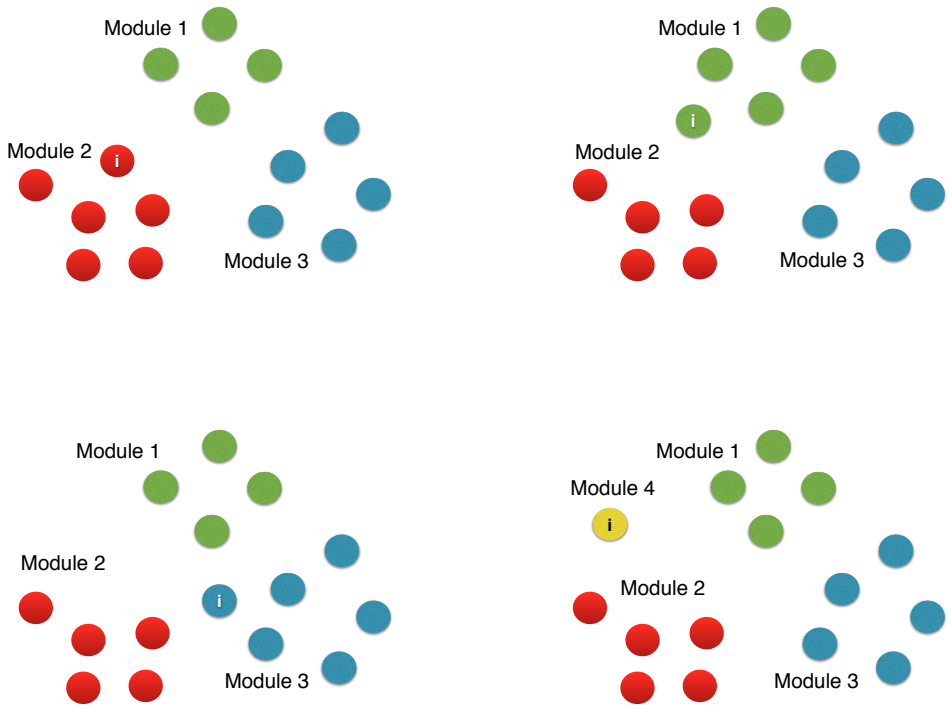


Figure 3.2: The neighbors of current subnetwork structure obtained by flipping model label of x_i. The neighborhood of the current network structure (top left panel) is defined as all structures that can be obtained by flipping the module label of a single variable. In this figure we show all neighbors by flipping variable i. Currently node x_i is in module 2, so it could be allocated to module 1 (top right panel), module 2 (itself, top left panel), module 3 (bottom left panel), or create a new module 4 by itself (bottom right panel). We will need to flip the module labels of all the other nodes to obtain the full neighborhood.

$\mathbf{R}_{\mathrm{P}}$ as

$$\mathbf{R}_{\mathrm{P}}(\Gamma,\Theta)=\sum_{g=1}^{G^s}\left[\sum_{i\in\{M^s=g\}}\int\left(\Gamma'_{\cdot i}\frac{d\boldsymbol{\phi}(t)}{dt}-\mathbf{LP}_i(\Gamma'\boldsymbol{\phi}(t),\boldsymbol{u}(t),\Theta_i)\right)^2dt\right]+\mu\cdot\sum_{i,i_2=1}^{d}\delta(m_i,m_{i_2}). \tag{3.8}$$

So in the parameter estimation step, we loop through all the network structures in the neighborhood of current structure M^s. For each structure, the term in the bracket does not contain neuronal state functions that are not in the module g, hence to minimize $\mathbf{R}_{\mathrm{P}}$ we only need to run regressions within each module. Because the size of $\mathcal{N}$ is $O(d^2)$ and linear regressions can be computed very fast, searching among the neighborhood of M^s is very efficient. We summarize our Potts-based iterative principal differential analysis algorithm in Figure 3.3.

Note that in step 0 we initialize M as d independent modules, just to be conservative and convenient. One can certainly pick any other structure if she has prior information. We will discuss starting point selection in Section 3.5 in detail.

3.3 Estimation with the L_1 Penalty

While P-iPDA utilizes the Potts penalty and the subnetwork structure, there are different types of sparsity structures. In this section we propose another modified iPDA algorithm by adding an L_1 penalty, the most popular sparsity-inducing penalty in the literature, on A and B to the criterion $\mathbf{H}(\Gamma,\Theta)$:

$$\mathbf{HL}(\Gamma,\Theta)=\mathbf{H}(\Gamma,\Theta)+\lambda\cdot\mu\cdot\left(\sum_{i_1,i_2=1}^{d}|A_{i_1i_2}|+\sum_{j=1}^{J}\sum_{i_1,i_2=1}^{d}|B_{j,i_1i_2}|\right), \tag{3.9}$$

The Potts-based iterative principal differential analysis (P-iPDA) algorithm:

0. Initialization. Set $M^0 = (1, ..., d)$ and $\Theta = \mathbf{0}$.

   At the $(s+1)$th step:

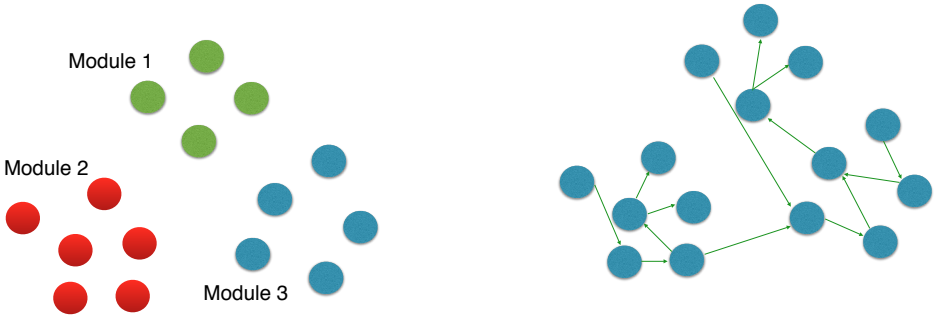
1. Data profiling. Fix $\hat{\Theta}^s$ and current subnetwork structure $\hat{M}^s$.
   For g in $1, ..., G^s$,
   Fit $\hat{x}_i$, i.e. $\hat{\Gamma}_{i\cdot}^{s+1}$, by minimizing $\mathbf{H}_{\mathrm{P,g}}$ (Equation 3.7) for all $i \in \{M^s = g\}$ at the same time within cluster g.

2. Model parameter estimation. Fix $\hat{\Gamma}^{s+1}$.
   For M^{s+1} in $\mathcal{N}(M^s)$,
   Estimate Θ by minimizing $\mathbf{R}_{\mathrm{P}}(\hat{\Gamma}^{s+1}, \Theta)$ (Equation 3.8) under subnetwork structure M^{s+1}.
   Select the $\hat{M}^{s+1}$ and $\hat{\Theta}^{s+1}$ that minimize $\mathbf{R}_{\mathrm{P}}$.

3. Iterate (1) and (2) until the subnetwork structure M stabilizes and the relative change of log-likelihood $\mathbf{HP}(\Gamma, \Theta)$ is less than a pre-specified threshold.

Figure 3.3: The Potts-based iterative principal differential analysis (P-iPDA) algorithm.

where the "L" stands for Lasso. The L_1 penalty shrinks many model parameters to be zero. In fact, this is equivalent to assuming a sparsely connected brain system in the sense that the effective connectivity between many regions are zero, distinct from a functionally segregated system in which effective connectivities are distributed densely within each module. The former might not be able to identify independent clusters, the latter can separate independent clusters but within each cluster, it fails to achieve further sparsely, i.e., all components within the same module are connected with each other. Figure 3.4 shows the difference between the two networks structures.



(a) A modularizable network

(b) A sparsely connected but inseparable network

Figure 3.4: Comparison of two sparse network structures: subnetwork structure vs sparsely connected network. The left panel shows a typical subnetwork structure in which there are 3 modules, within each module all nodes are completely connected (so we do not show the arrows) and between modules no one influence the other; the right panel shows sparsely connected network in which you cannot separate nodes into isolated sub-systems, but the connections are sparse. Arrows indicate directional connectivities.

Similarly we define

$$\mathbf{R}_{\mathrm{L}}(\Gamma,\Theta)=\sum_{i=1}^{d}\int\left((\Gamma_{\cdot i})'\frac{d\boldsymbol{\phi}(t)}{dt}-\mathbf{L}_i(\Gamma'\boldsymbol{\phi}(t),\boldsymbol{u}(t),\Theta_i)\right)^2dt + \mu\cdot\left(\sum_{i_1,i_2=1}^{d}|A_{i_1i_2}|+\sum_{j=1}^{J}\sum_{i_1,i_2=1}^{d}|B_{j,i_1i_2}|\right). \tag{3.10}$$

The scheme we propose to minimize $\mathbf{HL}(\Gamma,\Theta)$ is similar as iPDA and is summarized in Figure 3.5.

The first two steps (step 0 and step 1) are identical to the first two steps in iPDA. Step 2 is essentially an L_1-penalized regression (Lasso) problem which has been well developed and all the quadratic programming techniques can be applied. Among them two methods specifically designed for lasso regression - least angle regression (LARS, Efron *et al.*, 2004) and pathwise coordinate optimization (Friedman

The Lasso-based iterative principal differential analysis (L-iPDA) algorithm:

0. Initialize $\Theta = \Theta^0$;

   At the $(s+1)$th step:

1. Data profiling. Fix $\hat{\Theta}^s$ estimated from the sth step, find $\hat{\Gamma}^{s+1}$ that minimizes $\mathbf{H}(\Gamma, \hat{\Theta}^s)$ in Equation (3.3).
2. Model parameter estimation. Fix $\hat{\Gamma}^{s+1}$ estimated from Step 1, find $\hat{\Theta}^{s+1}$ that minimizes $\mathbf{R}_{\mathrm{L}}(\hat{\Gamma}^{s+1}, \Theta)$
3. Iterate (1) and (2) until the relative change of $\mathbf{HL}(\Gamma, \Theta)$ (Equation 3.9) is less than a pre-specified threshold.

Figure 3.5: The Lasso-based iterative principal differential analysis (L-iPDA) algorithm

et al, 2007) - are especially fast when dealing with our problem. In this dissertation we use the pathwise coordinate optimization algorithm to solve it. This algorithm breaks down a d-dimensional lasso regression to d one-dimensional lasso regressions which has analytical solution hence can be solved easily and quickly. Not to be confused here, we have to run d d-dimensional lasso regressions in parameter estimation step, and pathwise coordinate optimization algorithm can break each of the d-dimensional lasso regressions into d one-dimensional lasso problems. Besides the fact that pathwise coordinate optimization algorithm is among the fastest algorithms to solve lasso problem in the literature, another reason why we choose it is that the form of the optimization problem it handles is consistent with the form our problem is formulated. We know that one can represent a optimization problem in two forms equivalently (dual problems), i.e., either list the L_1 penalty as a constraint or incorporate it in the optimization function like in Equation (3.10). In this situation because we have d lasso regressions, and they share the same penalty

coefficient μ, it would be unrealistic to use the dual form which lists L_1 penalty as a constraint, hence other efficient algorithms like LARS is not applicable here.

For the same data, generally L-iPDA is much more computationally intensive than P-iPDA, mainly for two reasons. First, the calculation of Γ, a matrix of much larger dimension than that of the model parameters, involves expensive matrix inversion and dominates the computation. Through decomposing the whole dynamic network into smaller subnetworks and computing Γ for each subnetwork separately, P-iPDA replaces the computation of the large matrix Γ by the computation of several smaller matrices, and thus significantly reduce the computation load. Second, even though efficient algorithms like LARS or pathwise coordinate optimization algorithm might be able to achieve the same time complexity as OLS, computation of Lasso estimates generally takes longer time than that of the regular OLS estimates.

3.4 Selection of Penalty Parameters

3.4.1 A Modified Cross-validation Approach

The criteria $\mathbf{HP}(\Gamma, \Theta)$ depends on two penalty parameter, λ and μ, where the former is used to reduce variability of the estimated $\hat{\boldsymbol{x}}(t)$, and the latter is to regularize the ODE parameter estimates. Algorithms in previous sections all assume that λ and μ are held constant, but when dealing with real problems, how to select penalty coefficient is always a tricky question. Existing approaches for penalty selection include ordinary cross-validation (OCV), generalized cross-validation (GCV, Wahba, 1990), GCV for functional data (Reiss & Ogden, 2007, 2009), restricted maximum likelihood (Wood, 2011), etc. Because of the complex formulation of the optimizing functions in our setting, we propose a modified twenty-fold cross validation (TFCV)

to select the penalty parameters. For time series data, one cannot leave out all the data points in one continuous time period, because the resulting discontinuity will cause difficulty in predicting the left-out data. Therefore we use a different strategy to divide data into folds. For our ECoG real data analysis and simulations examples n is 251, so we select the number of folds to be 20 - neither too large to make estimation unstable, nor too small to make prediction unpersuasive.

We first divide the time series at T time points into $\lfloor (n-1)/20 \rfloor$ time intervals, each containing 20 consecutive points, where $\lfloor t \rfloor$ denotes the maximum integer no larger than t. Let each fold be a collection of one time point in each interval. For example, the first fold consists of data at time points $F_1 = (2, 2+20, ..., 2+20\lfloor (n-1)/20 - 1 \rfloor)$, and the twentieth fold consists of data at time points $F_{20} = (21, 21+20, ..., 21+20\lfloor (n-1)/20 - 1 \rfloor)$. Then given a candidate combination of (λ, μ), we leave out one fold of data each time, and use the rest of data to fit $\boldsymbol{x}(t)$ at the rest $n - \lfloor (n-1)/20 \rfloor$ time points, and estimate Θ using the proposed iterative procedures. Prediction of the left-out data is conducted as follows. Suppose the data at time t_{v+1} are left out, then based on the estimated $\hat{\boldsymbol{x}}(t_v)$ and $\hat{\Theta}$, the predicted $\boldsymbol{x}$ at t_{v+1} is given by

$$\tilde{\boldsymbol{x}}(t_{v+1}) = \hat{\boldsymbol{x}}(t_v) + \left(\hat{A}\hat{\boldsymbol{x}}(t_v) + \sum_{j=1}^{J} u_j(t_v)\hat{B}_j\hat{\boldsymbol{x}}(t_v) + \hat{C}\boldsymbol{u}(t_v) + \hat{D} \right) \cdot \hbar.$$

where $\hbar = t_{v+1} - t_v$. Then we use the sum of prediction error (SPE)

$$\text{SPE} = \sum_{i=1}^{20} \sum_{t \in F_h} \sum_{i=1}^{d} \left(y_i(t) - \tilde{x}_i(t) \right)^2 \tag{3.11}$$

as the criterion to select the best combination of (λ, μ).

One can also use a modified leave-one-out cross-validation (LOOCV) to select

the penalty parameters. The regular LOOCV leaves one point out each time, fits the model, then records the difference between predicted value and the actual value that was left out. But the regular LOOCV is very time consuming, because we have to repeat our estimation process 251 times. Given the continuity of the dynamic system under study, one may conduct the LOOCV only on a subset of the original time observations. For example, we can select 50 points randomly, and when applying LOOCV, we only leave those 50 points out such that computational time is reduced to 1/5 of original computational time. We call this algorithm the random leave-one-out cross-validation (RLOOCV). In real data and simulation examples, we find that TFCV and RLOOCV with different number of left-out points (as long as it's not too small) will produce similar results, so it's completely up to the researcher which cross-validation method to use.

Note that the discussion above mainly serves for parameter selection of P-iPDA. For iPDA only λ is needed and we can simply select the same λ as P-iPDA, because the same level of roughness is preferred. Same λ is also applied to L-iPDA. However we tune μ in L-iPDA by adjusting the percentage of zeros we want to achieve.

We also need to select other parameters like the number of bases p in data profiling step. We use equally-spaced cubic spline bases to represent $\boldsymbol{x}(t)$ since the time series data under study are equally spaced. Through simulations we found that choice of the number of spline bases did not affect the results much, as long as p was not too far from the number of observations n.

3.4.2 A Prescreen Process

Even the TFCV/RLOOCV is extremely time-consuming if a scan of a large pool of candidates of λ and μ is desired. Because at the selection step, in order not to

miss possible candidates, it's inevitable to select very small μ's, which will incur expensive computational cost in two ways. First, small μ clusters many components into the same module, resulting P-iPDA many iterations to converge. Second, when many components fall into the same module, the computation for the associated Γ, the coefficients of B-spline bases representing $x_i(t)$'s in the same cluster, involves inverting a large matrix, which is computationally extensive. Actually small μ will degenerate the P-iPDA to iPDA in data profiling step after enough iterations which holds the larger proportion of computation in P-iPDA.

So we propose a method that can prescreen hence reduce the large candidate pool purely based on the curve fitting error SSE and the fidelity to ODE model Fid. For each combination of λ and μ in the original large candidate pool, we apply P-iPDA to the entire data, and keep records of $\text{SSE}(\lambda, \mu)$ and $\text{Fid}(\lambda, \mu)$ outputted from the final step. We know that λ controls the fidelity and also the roughness of the curves. So for the same μ (we will fix μ at a very small value at the beginning), the larger the λ is, the larger the SSE is, and the smaller the Fid is (the smoother the curves are). So we abandon those λ's that cannot balance SSE and Fid. And we define this balance as the ratio of the relative percentage changes of them. For example, suppose when we increase λ, the SSE increases 10 times but Fid only decreases by half, then we will abandon this λ and all the larger ones. Since we absolutely do not want to discard any λ that might possibly be a good selection, so we will choose a very conservative ratio cutoff. But even a conservative ratio cutoff can lessen the candidate pool size tremendously. For the same λ, we gradually increase the value of μ so that the cluster size decreases. There will be d clusters when μ is large enough. If a larger μ and smaller cluster size jeopardize neither SSE nor Fid, then we discard the larger μ and only keep the smaller one because we prefer a simple model than a complex one. Then among the rest of a few parameters with both small $\text{SSE}(\lambda, \mu)$

and Fid(λ, μ), we use RLOOCV/TFCV to select the (λ, μ) for final data analysis. We demonstrate how this process works in Section 6.2 through simulation examples.

3.5 Possible Improvements of P-iPDA

Previously in this chapter we have presented the scheme of P-iPDA, especially how to search for the local optimum in the neighborhood of current network structure. While laying out the steps of P-iPDA, we were focusing on delivering the main idea of P-iPDA so that the algorithm was stated as succinct as possible. As readers can sense that this iterating scheme has a strong Metroplis-Hastings flavor, hence some improvements designed for Metroplis-Hastings can also be applied for P-iPDA. Its uniqueness also allows other directions of improvements. In this section we point out some possible directions of improvements.

Starting point

In our P-iPDA we always start from $\Theta = \mathbf{0}$ which specifies the network structure as the most parsimonious one, the one every node forms a module by itself. We recommend this starting point not only because it's parsimonious, but also because it's most computationally economic. With every module only having one node, it's equivalent to fitting d one-dimensional ODEs at the first iteration. And it will keep the computation fast at least for the first a few iterations. As the size of the module increases the network structure will stabilize. This selection of starting point makes sense especially when we know nothing about the system. But we also want to try different starting points for two reasons: (1) Prior knowledge on the network structure. For example, under some circumstances researchers might have known roughly how the neuronal states are segregated so they can incorporate this idea into the

model by tuning the starting point. (2) Preventing P-iPDA from getting stuck at local minimum. The updating scheme of P-iPDA guarantees that it will converge but does not guarantee it will converge to the global optimum. The P-iPDA faces the same problem as Metroplis-Hastings algorithm. The estimator is approximately optimal, especially when the optimization function has high dimension and the surface is hilly, the estimator may never get anywhere near the global optimum. Hence when we deal with high dimensional problem, trying different starting point is an intuitive way to prevent us from misleading by a single path. For example, one can select five different starting points with different network structures ranging from d modules to a single big module, then choose the estimator that minimizes the optimization function.

Network updating scheme

Besides starting point, another reason that our Potts-based strategy might get stuck in local minimum is that the neighborhood we search in every iteration is too small to explore a significant amount of the whole sample space. So if we can improve how we update the module structure such that more area can be investigated, we are able to increase the probability of "jumping out" from a local minimum. We propose the following ideas:

1. Currently we define neighborhood of a network structure as structures that can be obtained by flipping module label of at most one node. This produces only $O(dG)$ elements in the neighborhood including the current structure itself, where G is the number of modules in current network structure. Compared to the number of total possible module labels $O(G^d)$ the neighborhood is too small. Hence we can enlarge the neighborhood by allowing two or more labels

to flip their labels. By allowing two nodes to change their module labels we increase the number of elements in the neighborhood to $O(d^2G^2)$, hence increase the chance to achieve global optimum.

2. When searching in the neighborhood of current network structure, we require that the estimate of Θ that minimizes the function $\mathbf{R}_{\mathrm{P}}$ is selected for next iteration. This criterion ensures us in the steepest descent direction within the neighborhood of current network structure. But this also binds us in that direction and sometimes leads us to a local minimum after a few iterations. One possible improvement is to give the estimation of Θ and the corresponding module labels M more freedom. For example, we can keep the best two estimates for the next iteration, and run the next iteration using each of the two estimates respectively, then combine the results and keep the best two estimates, and so on. Of course one can select the best three, four or more estimates to keep for future iterations.

Essentially either starting from more than one points or updating the network shakier tries to expand the space each iteration can explore. By expanding the search space, both methods sacrifice computational speed. So there is a trade-off between accuracy and complexity (both time and space). For example, if you try 10 staring points, then the computation time is approximately 10 times of the original time.

Although both methods targets the same goal, that is, to expand the search space, they stand in different point of view. This enables us to combine both methods to increase the accuracy even further. For example, one can try multiple starting points, and for each starting points, one can allow multiple nodes to flip their labels. All those improvements can be made and adjusted based on real problems.

Channel-specific penalization parameter

The first two methods both try to expand the space that each iteration can explore. Here we start from a different standing point. When penalizing the ODE model penalty and Potts penalty (or L_1 penalty for L-iPDA) in Equation (3.5), we control the amount by parameter λ and μ respectively. Here we focus on λ. We know that the ODE model penalty

$$\int \left(\frac{dx_i(t)}{dt} - \mathbf{LP}_i(\Gamma' \boldsymbol{\phi}(t), \boldsymbol{u}(t), \Theta_i) \right)^2 dt$$

controls the variance of the fitted time series $\hat{x}_i(t)$. The ideal value of λ is determined by the signal-to-noise ratio. For example, if the time series are observed without error, the perfect λ will be zero, i.e., we want to fit the data completely by minimizing SSE. On the other hand, if the signal-to-noise ratio is small, we will prefer a large λ to control the variance. However, in real problems we don't know the signal-to-noise ratios of $x_i(t)$'s so in our algorithms we simply set a single λ. It also has computational concerns because if we use channel-specific λ's at data profiling step to minimize

$$\sum_{i=1}^{d} \left[\sum_{j=1}^{n} (y_i(t_j) - \Gamma' \boldsymbol{\phi}(t_j))^2 + \lambda_i \int \left(\Gamma'_{\cdot i} \frac{\boldsymbol{\phi}(t)}{dt} - \mathbf{L}_i(\Gamma' \boldsymbol{\phi}(t), \boldsymbol{u}(t), \Theta_i) \right)^2 dt \right],$$

we have to select all the λ_i's by cross-validation. Even the selection of a single λ and μ is already computational heavy, and we have to run them in a parallel way. So it's impossible for us to select 50 λ_i's by cross-validation. Here we propose a method to estimate the ratios of ideal λ_i's, then λ_i can be represented as a common factor λ

multiplied by this ratio:

$$\lambda_i = \lambda \cdot c_i,$$

where c_i is the estimated ratio, and λ is the only parameter that needs to be selected by cross-validation. So this is equivalent as the previous penalty term computationally. And c_i's are estimated by the following non-iterative method. First we fit a naive nonparametric on $\boldsymbol{y}(t)$ (use second order derivative as penalty), then we regress $d\hat{\boldsymbol{x}}(t)/dt$ on $\hat{\boldsymbol{x}}$. The MSE's in the regression analysis are considered as good candidates of c_i's.

3.6 Proof of Convergence

In iPDA, P-iPDA and L-iPDA there is a common stop criterion - the optimization function $\mathbf{H}$ to converge. In this section we prove that this will always happen hence the algorithms will always stop within a certain amount of time. Here we prove this result for iPDA but everything can be extended similarly to P-iPDA and L-iPDA.

Let Γ^i and Θ^i be parameters estimated from the ith iteration, we have

$$\begin{aligned}
\mathbf{H}^i &= \mathbf{H}(\Gamma^i|\Theta^i) \\
&= \sum_{i=1}^{p} \|Y_{\cdot i} - \Phi\Gamma^i_{\cdot i}\|^2 + \mathbf{J}(\Theta^i|\Gamma^i) \\
&\leq \sum_{i=1}^{p} \|Y_{\cdot i} - \Phi\Gamma^i_{\cdot i}\|^2 + \mathbf{J}(\Theta^{i-1}|\Gamma^i) \\
&\leq \sum_{i=1}^{p} \|Y_{\cdot i} - \Phi\Gamma^{i-1}_{\cdot i}\|^2 + \mathbf{J}(\Theta^{i-1}|\Gamma^{i-1}) \\
&= \mathbf{H}(\Gamma^{i-1}|\Theta^{i-1}),
\end{aligned}$$

where $\mathbf{H}(\Gamma^i|\Theta^i)$ indicates that we are in the data profiling step, because we are given the model parameters and are minimizing $\mathbf{H}$ by selecting Γ^i, while

$$\mathbf{J}(\Theta|\Gamma) = \lambda \sum_{i=1}^{d} \int \left(\frac{d\hat{x}_i(t)}{dt} - \mathbf{L}_i(\Gamma' \boldsymbol{\phi}(t), \boldsymbol{u}(t), \Theta_i) \right)^2 dt$$

indicates that we are in the model parameter estimation step, because we are given the fitted $\hat{\boldsymbol{x}}(t)$ and are minimizing $\mathbf{J}$ by selecting Θ. The first inequality is because Θ^i minimizes $\mathbf{H}^i$, and the second inequality is because Γ^i minimizes $\mathbf{H}^i$. We have shown $\{\mathbf{H}^i\}$ is a decreasing sequence and it's greater than 0, hence the sequence $\{\mathbf{H}^i\}$ must converge.

Chapter 4

Computational Issues

The previous chapter lays out a big picture of how we approach the high dimensional dynamic system, but there are still too many computation details remains to be resolved. In this chapter we focus on tackling computation issues that suffice us to package the algorithms to usable software.

Essentially the computation of iPDA, P-iPDA and L-iPDA are similar. P-iPDA is basically running iPDA in subnetworks (plus some network structure updating scheme), and L-iPDA differs from iPDA only at the parameter estimation step, in which L-iPDA fits d lasso regressions while regular iPDA fits d OLS regressions. So in this chapter, we mainly work on how to run iPDA in a computationally economic way. We consider the case when the system only inputs a single stimulus, i.e., $J = 1$, because of three reasons: (1) both our real data analysis and simulation examples only contain one exogenous input source; (2) Notation compactedness; (3) All the computation derived in the case when $J = 1$ can be easily extended to more general case when $J > 1$. So we simply use B to denote B_1, C_i to denote $C_{i\cdot}$ and $u(t)$ to denote $\boldsymbol{u}(t)$. One thing worth mentioning is that in our real data analysis and simulation examples, we use a single λ to account for all the neuronal state

functions, while in the future we might want to improve our model by considering channel specific penalty coefficients. So we generalize our model to allow different model penalty coefficient for different system variable:

$$\begin{aligned}\mathbf{H}(\Gamma,\Theta) =& \sum_{i=1}^{d}\sum_{j=1}^{n}(y_i(t_j)-\Gamma'_{\cdot i}\boldsymbol{\phi}(t_j))^2 \\ &+\sum_{i=1}^{d}\lambda_i\cdot\int\left(\Gamma'_{\cdot i}\frac{d\boldsymbol{\phi}(t)}{dt}-\mathbf{L}_i(\Gamma'\boldsymbol{\phi}(t),\boldsymbol{u}(t),\Theta_i)\right)^2 dt.\end{aligned}$$

This chapter is organized as follows. We first address how to fit neuronal state functions, which is the same across all three algorithms and is usually much more computationally intensive than model parameter estimation. Then we will introduce how to estimate model parameter algorithm by algorithm.

4.1 Data Profiling

Our main concern is the first step, that is, given model parameter Θ, to estimate Γ. In Equation (3.3) we denote the first term SSE and the second term $\mathbf{J}$:

$$\mathbf{H}(\Gamma,\Theta) = \mathrm{SSE} + \mathbf{J}(\Gamma,\Theta),$$

where $\mathbf{J}$ is Fid multiplied by the smoothness coefficient λ. Let

$$\Phi = (\Phi_{ij})_{n\times p} = (\phi_j(t_i)), 1\le i\le n, 1\le j\le p,$$

where n is the number of time observations and p is the number of basis splines. The SSE can be written as

$$\begin{aligned}
\text{SSE} &= \sum_{i=1}^{d} \|Y_{\cdot i} - \Phi\Gamma_{\cdot i}\|^2 \\
&= \sum_{i=1}^{d} (Y_{\cdot i} - \Phi\Gamma_{\cdot i})'(Y_{\cdot i} - \Phi\Gamma_{\cdot i}) \\
&= \sum_{i=1}^{d} (\Gamma'_{\cdot i}\Phi'\Phi\Gamma_{\cdot i} - 2\Gamma'_{\cdot i}\Phi' Y_{\cdot i} + Y'_{\cdot i}Y_{\cdot i}) \\
&= \boldsymbol{\gamma}' P \boldsymbol{\gamma} - 2\boldsymbol{\gamma}' Q' \mathbf{y} + \mathbf{y}'\mathbf{y},
\end{aligned}$$

where

$$\boldsymbol{\gamma}_{pd\times 1} = \begin{bmatrix} \Gamma_{\cdot 1} \\ \vdots \\ \Gamma_{\cdot d} \end{bmatrix}, \quad \mathbf{y}_{nd\times 1} = \begin{bmatrix} Y_{\cdot 1} \\ \vdots \\ Y_{\cdot d} \end{bmatrix},$$

$$P_{pd\times pd} = \begin{bmatrix} \Phi'\Phi & & \\ & \ddots & \\ & & \Phi'\Phi \end{bmatrix}, \quad Q_{nd\times pd} = \begin{bmatrix} \Phi & & \\ & \ddots & \\ & & \Phi \end{bmatrix}.$$

When computing $\mathbf{J}(\Gamma, \Theta)$, we use Riemann sum to approximate the integral. The points we used for Riemann sum are $(t_1 = 0, t_2, \ldots, t_N = T) \in [0, T]$. Note that these points are not necessarily the same as the points where the original data are measured at, we use the same symbol t just for convenience and it's not difficult to tell when we are using which. For computational convenience we choose them to be equally spaced, the length of each interval is $l = \frac{T}{N-1}$. Formally $\mathbf{J}(\Gamma, \Theta)$ can be

represented as

$$
\begin{aligned}
\mathbf{J} &= \sum_{i=1}^{d} \lambda_i \int_0^T \left[\dot{x}_i(t) - A_{i\cdot}\boldsymbol{x}(t) - u(t)B_{i\cdot}\boldsymbol{x}(t) - u(t)C_i - D_i\right]^2 dt \\
&= \sum_{i=1}^{d} \lambda_i \int_0^T \Big[\Big(- A_{i1}\boldsymbol{\phi}(t)' - u(t)B_{i1}\boldsymbol{\phi}(t)', \ldots, \dot{\boldsymbol{\phi}}(t)' - A_{ii}\boldsymbol{\phi}(t)' - u(t)B_{ii}\boldsymbol{\phi}(t)', \\
&\qquad\qquad \ldots, -A_{id}\boldsymbol{\phi}(t)' - u(t)B_{id}\boldsymbol{\phi}(t)'\Big)\boldsymbol{\gamma} - u(t)C_i - D_i\Big]^2 dt \\
&\approx \sum_{i=1}^{d} \lambda_i l \cdot (R_i\boldsymbol{\gamma} - S_i)'(R_i\boldsymbol{\gamma} - S_i) \\
&= \left[\boldsymbol{\gamma}'\left(\sum_{i=1}^{d}\lambda_i R_i'R_i\right)\boldsymbol{\gamma} - 2\boldsymbol{\gamma}'\left(\sum_{i=1}^{d}\lambda_i R_i'S_i\right) + \sum_{i=1}^{d}\lambda_i S_i'S_i\right]\cdot l, \qquad (4.1)
\end{aligned}
$$

where in the third line we approximate the integral by Riemann sum, and R_i and S_i are defined as

$$
(R_i)_{N\times dp} = \begin{bmatrix} -f_{i1}(t_1) & \ldots & \dot{\boldsymbol{\phi}}(t_1)' - f_{ii}(t_1) & \ldots & -f_{id}(t_1) \\ \vdots & & \vdots & & \vdots \\ -f_{i1}(t_N) & \ldots & \dot{\boldsymbol{\phi}}(t_N)' - f_{ii}(t_N) & \ldots & -f_{id}(t_N) \end{bmatrix},
$$

$$
(S_i)_{N\times 1} = \begin{bmatrix} C_i u(t_1) + D_i \\ \vdots \\ C_i u(t_N) + D_i \end{bmatrix},
$$

where

$$
f_{ij}(t) = A_{ij}\boldsymbol{\phi}(t)' + u(t)B_{ij}\boldsymbol{\phi}(t)'.
$$

Let

$$(A_i)_{N\times dp} = \begin{bmatrix} A_{i1} & \cdots & A_{i1} & \cdots & A_{id} & \cdots & A_{id} \\ \vdots & & & \ddots & & & \vdots \\ A_{i1} & \cdots & A_{i1} & \cdots & A_{id} & \cdots & A_{id} \end{bmatrix},$$

$$(B_i)_{N\times dp} = \begin{bmatrix} B_{i1} & \cdots & B_{i1} & \cdots & B_{id} & \cdots & B_{id} \\ \vdots & & & \ddots & & & \vdots \\ B_{i1} & \cdots & B_{i1} & \cdots & B_{id} & \cdots & B_{id} \end{bmatrix},$$

$$(a_m)_{N\times dp} = \begin{bmatrix} u(t_1) & \cdots & u(t_1) \\ \vdots & & \vdots \\ u(t_N) & \cdots & u(t_N) \end{bmatrix}, \quad (a_v)_{N\times 1} = \begin{bmatrix} u(t_1) \\ \vdots \\ u(t_N) \end{bmatrix},$$

and

$$\Psi_{N\times dp} = \begin{bmatrix} \boldsymbol{\phi}(t_1)' & \cdots & \boldsymbol{\phi}(t_1)' \\ \vdots & \ddots & \vdots \\ \boldsymbol{\phi}(t_N)' & \cdots & \boldsymbol{\phi}(t_N)' \end{bmatrix}, \quad (\dot{\Psi}_i)_{N\times pd} = \begin{bmatrix} 0 & \dot{\boldsymbol{\phi}}(t_1)' & 0 \\ & \vdots & \\ 0 & \dot{\boldsymbol{\phi}}(t_N)' & 0 \end{bmatrix},$$

then we can write R_i as

$$R_i = -(A_i + a_m \odot B_i) \odot \Psi + \dot{\Psi}_i,$$

where $\odot$ means element-wise multiplication which is much faster than regular matrix multiplication. Further if we let

$$R = \sum_{i=1}^{d} \lambda_i R_i' R_i,$$
$$S = \sum_{i=1}^{d} \lambda_i R_i' S_i,$$

we can write $\mathbf{J}$ as

$$\mathbf{J} = \left(\boldsymbol{\gamma}' R \boldsymbol{\gamma} - 2\boldsymbol{\gamma}' S + \sum_{i=1}^{d} \lambda_i S_i' S_i \right) \cdot l.$$

Hence we have the data fitting criterion

$$\begin{aligned} \mathbf{H} &= \mathrm{SSE} + \mathbf{J} \\ &= \boldsymbol{\gamma}'(P + Rl)\boldsymbol{\gamma} - 2\boldsymbol{\gamma}'(Q'\mathbf{y} + Sl) + \sum_{i=1}^{d} (Y_{\cdot i}' Y_{\cdot i} + \lambda_i S_i' S_i l). \end{aligned}$$

We now represent $\mathbf{H}$ in a typical quadratic form, the optimal solution can be obtained as

$$\hat{\boldsymbol{\gamma}} = (P + Rl)^{-1}(Q'\mathbf{y} + Sl). \tag{4.2}$$

When d and N are large, it will be really inefficient to compute those matrix multiplications and inversions repeatedly. If we can save some common intermediate results in the memory and read them in each iteration instead of computing them every time, we are able to save tremendous computation time. In Equation (4.2), P and Q are fixed, most of the computation comes from R and S, more specifically,

from $\sum_{i=1}^{d} \lambda_i R_i' R_i$ and $\sum_{i=1}^{d} \lambda_i R_i' S_i$. In detail,

$$
\begin{aligned}
R_i' R_i &= \left[-(A_i + a_m \odot B_i) \odot \Psi + \dot{\Psi}_i\right]' \left[-(A_i + a_m \odot B_i) \odot \Psi + \dot{\Psi}_i\right] \\
&= A_{i,R} \odot A_{i,R}' \odot (\Psi'\Psi) + A_{i,R} \odot B_{i,R}' \odot (a_m \odot \Psi)'\Psi \\
&\quad + B_{i,R} \odot A_{i,R}' \odot \Psi'(a_m \odot \Psi) + B_{i,R} \odot B_{i,R}' \odot (a_m \odot \Psi)'(a_m \odot \Psi) \\
&\quad - A_{i,R} \odot \Psi_i'^T \Psi - B_{i,R} \odot \Psi_i'^T (a_m \odot \Psi) - A_{i,R}' \odot \Psi' \dot{\Psi}_i \\
&\quad - B_{i,R}' \odot (a_m \odot \Psi)' \dot{\Psi}_i + \Psi_i'^T \dot{\Psi}_i \qquad (4.3) \\
R_i' S_i &= \left[-(A_i + a_m \odot B_i) \odot \Psi + \dot{\Psi}_i\right]' [C_i a_v + D_i \mathbf{1}] \\
&= -C_i A_{i,R}' \odot \Psi' a_v - C_i B_{i,R}' \odot (a_m \odot \Psi)' a_v + C_i \Psi_i'^T a_v \\
&\quad - D_i A_{i,R}' \odot \Psi' \mathbf{1} - D_i B_{i,R}' \odot (a_m \odot \Psi)' \mathbf{1} + D_i \Psi_i'^T \mathbf{1},
\end{aligned}
$$

where $B_{i,R}$ is the same as B_i but resized to make the dimension appropriate for multiplication, its a dp by dp matrix while B_i is a N by dp matrix,

$$
(\Psi' \dot{\Psi}_i)_{pd \times pd} = \begin{bmatrix} 0 & \sum_{i=1}^{N} \boldsymbol{\phi}(t_i)' \dot{\boldsymbol{\phi}}(t_i) & 0 \\ \vdots & \vdots & \vdots \\ 0 & \sum_{i=1}^{N} \boldsymbol{\phi}(t_i)' \dot{\boldsymbol{\phi}}(t_i) & 0 \end{bmatrix}
$$

and

$$
(\Psi_i'^T \dot{\Psi}_i)_{pd \times pd} = \begin{bmatrix} 0 & \dots & 0 \\ \vdots & \sum_{i=1}^{N} \dot{\boldsymbol{\phi}}(t_i)' \boldsymbol{\phi}(t_i) & \vdots \\ 0 & \dots & 0 \end{bmatrix}.
$$

Three notes for the previous computation:

- We use the result which can be easily shown that if each column of B_i is

constant, then $A \cdot (B \odot C) = B \odot (A \cdot C)$, and, on the other hand, if each row of B_i is constant then $(B \odot A) \cdot C = B \odot (A \cdot C)$, but B may be resized to match the dimension.

- In Equation (4.3), only A_i, B_i, C_i, D_i need to be updated during iteration, all the other terms are fixed, and the only operation is matrix addition and element-wise multiplication which can be computed in $O(p^2d^2)$ time in each iteration. This time complexity doesn't take into account computing Equation (4.2). We don't take it into account because matrix inversion and multiplication have time complexity approximately $O(p^3d^3)$ and cannot be improved, and we only compute it once in each iteration, while computing R and S can be improved and they need to be updated d times which is time-consuming if d is large.

- This section only introduces computational issues for iPDA. For iPDA-lasso and iPDA-cluster, we need to estimate Θ within each cluster and combine the estimates. So we need to extract corresponding submatrices from Equation (4.3) for each cluster. If we only have n_c clusters and the biggest cluster contains d_c variables, then the time complexity for this iteration is $O(n_c p^2 d_c^2)$. When d is large, we improve the complexity dramatically as long as we have large number of clusters. An extreme example is, if we have a complete cluster, that is, all the variables are in different clusters, then $n_c = p$ and $d_c = 1$, we reduce the complexity from $O(p^2d^2)$ to $O(p^2d)$ - linear time in d.

4.2 ODE Parameter Estimation

The second step of iPDA, P-iPDA and L-iPDA is ODE parameter estimation, which has been assumed that can be solved by linear regression. In this section we will show details of transforming ODE parameter estimation to a regression problem and other computational details.

4.2.1 Parameter Estimation of iPDA and P-iPDA

In parameter estimation step of iPDA our goal is to find Θ to minimize $\mathbf{J}$ in Equation (4.1) given Γ. Since the ith term in the summation only contains $\Theta_{i\cdot}$, the ith row of Θ, and $\Theta_{i\cdot}$ only appears in the ith term, hence minimizing $\mathbf{H}$ is equivalent to minimizing

$$\int_0^T \left[\dot{x}_i(t) - A_{i\cdot}\boldsymbol{x}(t) - u(t)B_{i\cdot}\boldsymbol{x}(t) - u(t)C_i - D_i\right]^2 dt$$

for each i separately. We can approximate the integral above again by Riemann Sum which can be represented by a quadratic form:

$$\begin{aligned}
\mathbf{J}_i &= \int_0^T \left[\dot{x}_i(t) - A_{i\cdot}\boldsymbol{x}(t) - u(t)B_{i\cdot}\boldsymbol{x}(t) - u(t)C_i - D_i\right]^2 dt \\
&\approx l \cdot \left[\hat{\boldsymbol{x}}(\boldsymbol{\nu})A'_{i\cdot} + a_m \cdot \hat{\boldsymbol{x}}(\boldsymbol{\nu})B'_{i\cdot} + C_i u(\boldsymbol{\nu}) + D_i\mathbf{1} - \hat{\dot{x}}_i(\boldsymbol{\nu})\right]' \\
&\quad \cdot \left[\hat{\boldsymbol{x}}(\boldsymbol{\nu})A'_{i\cdot} + a_m \cdot \hat{\boldsymbol{x}}(\boldsymbol{\nu})B'_{i\cdot} + C_i u(\boldsymbol{\nu}) + D_i\mathbf{1} - \hat{\dot{x}}_i(\boldsymbol{\nu})\right] \\
&= l \cdot \left[(\hat{\boldsymbol{x}}(\boldsymbol{\nu}), a_m \cdot \hat{\boldsymbol{x}}(\boldsymbol{\nu}), u(\boldsymbol{\nu}), \mathbf{1})\,\Theta'_{i\cdot} - \hat{\dot{x}}_i(\boldsymbol{\nu})\right]' \\
&\quad \cdot \left[(\hat{\boldsymbol{x}}(\boldsymbol{\nu}), a_m \cdot \hat{\boldsymbol{x}}(\boldsymbol{\nu}), u(\boldsymbol{\nu}), \mathbf{1})\,\Theta'_{i\cdot} - \hat{\dot{x}}_i(\boldsymbol{\nu})\right],
\end{aligned}$$

where $\boldsymbol{\nu} = (t_1, \ldots, t_N)$,

$$\hat{\boldsymbol{x}}(\boldsymbol{\nu})_{N\times p} = \begin{bmatrix} \hat{x}_1(t_1) & \cdots & \hat{x}_p(t_1) \\ \vdots & & \vdots \\ \hat{x}_1(t_N) & \cdots & \hat{x}_p(t_N) \end{bmatrix}, \quad \hat{\dot{x}}_i(\boldsymbol{\nu})_{N\times 1} = \begin{bmatrix} \hat{\dot{x}}_i(t_1) \\ \vdots \\ \hat{\dot{x}}_i(t_N) \end{bmatrix}.$$

and don't be confused with a_m in previous section, here a_m is slightly different from the previous a_m, here it's a N by d matrix:

$$(a_m)_{N\times d} = \begin{bmatrix} u(t_1) & \cdots & u(t_1) \\ \vdots & & \vdots \\ u(t_N) & \cdots & u(t_N) \end{bmatrix}.$$

This quadratic optimization problem is equivalent to a least square regression, in which $\hat{\dot{x}}(\boldsymbol{\nu})$ serves as the responses and $[\hat{\boldsymbol{x}}(\boldsymbol{\nu}), a_m \cdot \hat{\boldsymbol{x}}(\boldsymbol{\nu}), u(\boldsymbol{\nu}), \mathbf{1}]$ serves as the design matrix.

For P-iPDA, it's similar that we also minimize $\mathbf{J}_i$ to obtain the estimate $\hat{\Theta}$. The difference lies in the fact that in iPDA every node $x_i(t)$ (precisely it's the derivative of x_i) is regressed by all other nodes while in P-iPDA it is only regressed by the nodes that are in the same module as x_i.

4.2.2 Parameter Estimation of L-iPDA

In L-iPDA algorithm we try to minimize the model-based roughness penalty $\mathbf{J}$ plus an L_1 penalty, use numerical approximation for the integral we have

$$\mathbf{L} = \sum_{i=1}^{d} \lambda_i \int_0^T \left[\dot{x}_i(t) - A_{i\cdot}\boldsymbol{x}(t) - u(t)B_{i\cdot}\boldsymbol{x}(t) - u(t)C_i - D_i\right]^2 dt + \mu|\Theta|,$$

where $|\Theta|$ means the summation of the absolute value of all the elements in Θ. Similarly it's equivalent to minimize each

$$\mathbf{L}_i = \left[(\hat{\boldsymbol{x}}(\boldsymbol{\nu}), a_m \cdot \hat{\boldsymbol{x}}(\boldsymbol{\nu}), u(\boldsymbol{\nu}), \mathbf{1})\,\Theta_{i\cdot}' - \hat{\dot{x}}_i(\boldsymbol{\nu})\right]' \left[(\hat{\boldsymbol{x}}(\boldsymbol{\nu}), a_m \cdot \hat{\boldsymbol{x}}(\boldsymbol{\nu}), u(\boldsymbol{\nu}), \mathbf{1})\,\Theta_{i\cdot}' - \hat{\dot{x}}_i(\boldsymbol{\nu})\right] + \frac{\mu}{l}|\Theta_{i\cdot}|.$$

This resembles doing lasso regression, as discussed in Chapter 3 we will use pathwise coordinate optimization algorithm to solve it.

4.3 Speeding up the Computation

So far this chapter has addressed the details we need for converting algorithms into codes. And our software package is indeed completely based on the topics we discussed in this chapter. However, there are still rooms for improvement. This section points out two possible strategies that we can make our program run faster, one tries to speed up the data profiling process by segmenting time interval, and the other relies on parallel computing techniques.

Speeding up basis smoothing

At the data profiling step, which is the same across all three algorithms, we select the Γ that minimize Equation (3.3). In most situations without requirement for high order differential continuity people tend to use the cubic B-spline which is shown in Figure 4.1. As you can see one basis only spans four subintervals, which is a relatively small number compared to total number of subintervals if we place hundreds of knots in the interval and can "take care" of boundary bases. Our idea is to segment the interval into a few subintervals and minimize Equation (3.3)

separately.

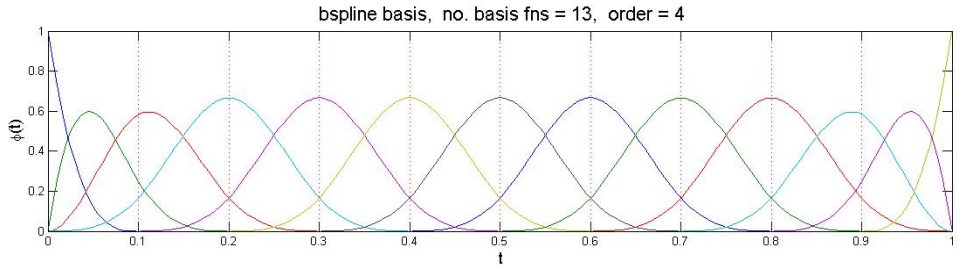


Figure 4.1: Cubic B-spline bases at [0,1]. Knots are placed uniformly between 0 and 1 with length 0.1.

Assume the original time span $[t_1, t_n]$ is divided into two parts with equal length. So $t_1, \ldots, t_{n/2}$ belong to the first subinterval and $t_{n/2+1}, \ldots, t_n$ belong to the second subinterval. For simplicity we assume $n/2$ is an integer, it's not difficult to deal with the case when it's not an integer. Let's further assume that the knots of B-spline are placed at observation points (this is indeed what Ramsay and Silverman, 2005 suggests). Putting aside the boundary situation (i.e., $t_{n/2}$, $t_{n/2+1}$ and their connecting dots) for now, we can rewrite $\mathbf{H}$ in Equation (3.3) as

$$\sum_{i=1}^{d}\sum_{j=1}^{n/2}(y_i(t_j) - \Gamma'_{\cdot i}\boldsymbol{\phi}(t_j))^2 + \lambda\sum_{i=1}^{d}\int_{t_1}^{\frac{t_1+t_n}{2}}\left(\Gamma'_{\cdot i}\frac{d\boldsymbol{\phi}(t)}{dt} - \mathbf{L}_i(\Gamma'\boldsymbol{\phi}(t), \boldsymbol{u}(t), \Theta_i)\right)^2 dt \tag{4.4}$$

$$+\sum_{i=1}^{d}\sum_{j=\frac{n}{2}+1}^{n}(y_i(t_j) - \Gamma'_{\cdot i}\boldsymbol{\phi}(t_j))^2 + \lambda\sum_{i=1}^{d}\int_{\frac{t_1+t_n}{2}}^{t_n}\left(\Gamma'_{\cdot i}\frac{d\boldsymbol{\phi}(t)}{dt} - \mathbf{L}_i(\Gamma'\boldsymbol{\phi}(t), \boldsymbol{u}(t), \Theta_i)\right)^2 dt. \tag{4.5}$$

For simplicity we use Γ in Equation (4.4) and Equation (4.5) but they are not the same Γ in Equation (3.3). In (4.4) Γ stands for the first $n/2$ rows of original Γ, and in (4.5) it stands for the last $n/2$ rows of original Γ. By segmenting the interval, we split the original problem of size p into two problems of size $p/2$. Remember

that the time complexity of computing Equation (4.3) is $O(p^2)$ in terms of p. So the segmentation can make significant improvement computationally if p is large.

Now the only problem left is how to handle boundary issue. When we segment the intervals, we have to truncate the bases that span two connected subintervals. For example, in Figure 4.1 suppose we segment the interval at 0.5, there are 3 bases spanning both $[0, 0.5]$ and $[0.5, 1]$ so we cannot obtain exact estimate of coefficients of those 3 bases. Since normally there are hundreds of bases in total so 3 or 6 (3 segmentations) inaccurate but not ridiculous estimates of basis coefficients will not make a noticeable impact on the whole estimation. One reasonable approximation can be achieved by truncating the bases and averaging estimates on shared bases.

Parallelizing the algorithms

Another way to expedite computation task is to leverage parallel computing techniques. If we apply the segmentation method in data profiling step, we can further speed up the process by parallelizing estimation on each subinterval. The parameter estimation step is also parallelizable since this step can be decomposed to d independent linear regressions. The main application of parallel computing is in the cross-validation which consumes intensive computation burden and can be easily parallelized.

Chapter 5

Application to a Real ECoG Study

In the Chapter 2 we built the Potts-based DDM model to characterize the functionally-segregated brain system. This modularizable effective connectivity structure has particular neuroscientific meaning for the ECoG data set because of its unique property. In this chapter we apply our model and algorithm to analyze the ECoG time series.

5.1 Data Acquisition

The ECoG data are collected from a right-handed adult female epilepsy patient, who had subdural electrodes implanted for clinical purposes of seizure localization and functional mapping prior to surgery for treatment of medically intractable seizures. The data were recorded from a 6×8 electrode array implanted over the left hemisphere, including the posterior temporal lobe (auditory cortex), of a right-handed adult female epilepsy patient (see Figure 5.1[1]). The patient had subdural electrodes implanted for clinical purposes of seizure localization and functional mapping prior

[1]The author thanks Dr. Deepti Ramadoss for assistance with this figure.

to surgery for treatment of medically intractable seizures. Electrodes were 2.3 mm diameter and spaced 9 mm center-to-center. Recordings were performed four days after electrode implantation while the patient was awake and fully responsive. The patient participated in several research studies and provided informed written consent for all research testing in compliance with the Johns Hopkins Institutional Review Board.

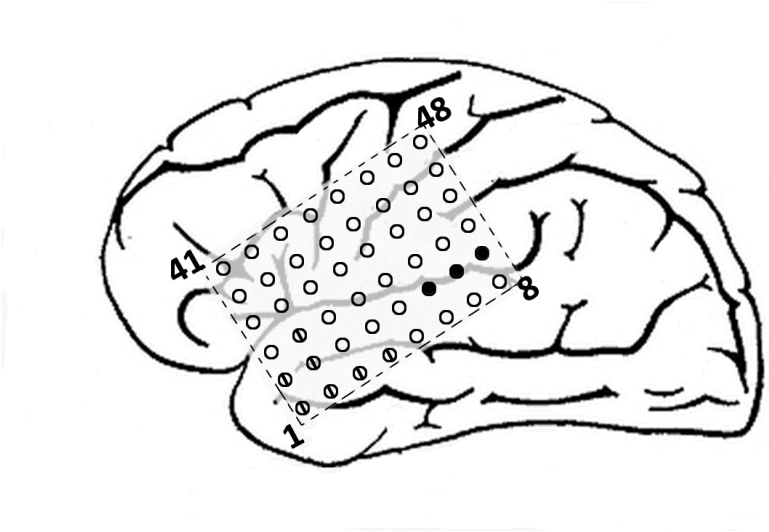


Figure 5.1: Two dimension schematic of left hemisphere with 8×6 electrode array superimposed. Electrode locations are estimated from intra-operative photographs and post-implantation CT scans. Filled electrodes (nodes 14, 15, and 16) represent sites where auditory responses were elicited. Electrodes 1-4, 9-10, and 18 corresponding to circles with vertical lines inside are epileptic areas.

Event-related ECoG recordings were acquired simultaneously from all electrodes using an established 300-trial passive auditory oddball paradigm (Sinai *et al.*, 2009; Cervenka *et al.*, 2013). Two 50 ms duration single-frequency tones were presented: a frequently repeated 1000 Hz tone ($N = 246$ trials) and an infrequently and pseudo-randomly presented (no consecutive repetitions) 1200 Hz tone ($N = 54$ trials). Tone stimuli were presented binaurally at a comfortable listening level through earphones at inter-stimulus intervals of 1450 ms. To reduce attention effects, the patient

watched an animated movie with no sound. The continuous ECoG signal was amplified (5×1000) and recorded digitally using a referential montage, 1000 Hz A/D sampling and a bandwidth of 0.1-300 Hz, as previously described (Cervenka *et al.*, 2013). Two electrodes (channels 47 and 48) in the top corner of the array, outside perisylvian cortex, were assigned as the reference and ground electrodes. Stimulus onset markers were recorded simultaneously to separate EEG channels.

ECoG recordings from the 46 active electrode channels were reviewed visually to identify any with excessive artifact for exclusion from analysis. One channel was identified as noisy and excluded (channel 32). The remaining d = 45 channels of ECoG time series data were analyzed. For each channel, the ECoG signal was segmented into 300 trials of 250 ms duration: 100 ms pre-stimulus ($0 - 100$ ms), 50 ms in-stimulus ($100 - 150$ ms), and 100 ms post-stimulus ($150 - 250$ ms). We use relatively short segments because recent studies have postulated non-stationary dynamics in brain activity, i.e. connectivity relationships vary rapidly over time. Since 1000 Hz is far more frequently presented than 1200 Hz, we focus on the analysis of 246 trials under 1000 Hz. For each 250-ms trial, let $u(t) = 1$ for $100 \leq t \leq 150$ and 0 otherwise. We omit the subscript for matrix B, as only one stimulus is considered in the model.

The presence of cortical auditory evoked and spectral power responses was used to identify electrode sites responsive to auditory stimulation. Evoked responses were derived by trial averaging of the phase-locked signal components in the time domain, where the phase lock refers to neuron firing at or near a particular phase angle of the sinusoidal stimulus sound wave; event- related changes in spectral power were determined by using time-frequency analysis. We focused on the evoked N1 response that occurs around 100 ms post-stimulus and is a large, obligatory cortical response to sound stimulation that is prominent in ECoG recordings from auditory

cortex (Edwards et al., 2005; Sinai et al., 2009). For the spectral power analysis, we used a time-frequency matching pursuit algorithm (Mallat and Zhang, 1993; Franaszczuk and Bergey, 1998; Durka *et al.*, 2001; Boatman-Reich *et al.*, 2010). A total of 3 electrode sites were identified as auditory responsive based on the presence of measurable auditory evoked N1 responses and in- creased spectral power (> 30 Hz). The three electrode sites were located in the posterior superior temporal gyrus, consistent with the location of auditory cortex (Figure 5.1). Based on clinical intracranial EEG recordings, seven electrode channels located in the inferior-anterior portion of the temporal lobe were associated with epileptiform activity and identified as the primary seizure focus (Figure 5.1).

5.2 Data Analysis

5.2.1 Problem Formulation and Parameter Selection

We use the same parameter set to analyze 246 trials of time series. For each trial, the parameters are summarized in Table 5.1. A snapshot of three channels in one trial is shown in Figure 5.2.

Table 5.1: Parameter setting for real data analysis

# of observations n	251
# of neuronal state functions d	45
# of knots $p-2$	101
# of points used for Riemann Sum	251
Initial guess Θ_0	$\mathbf{0}$
Stimulus function $u(t)$	$\mathbf{1}_{100\leq t\leq 150}$
Standardize	Yes

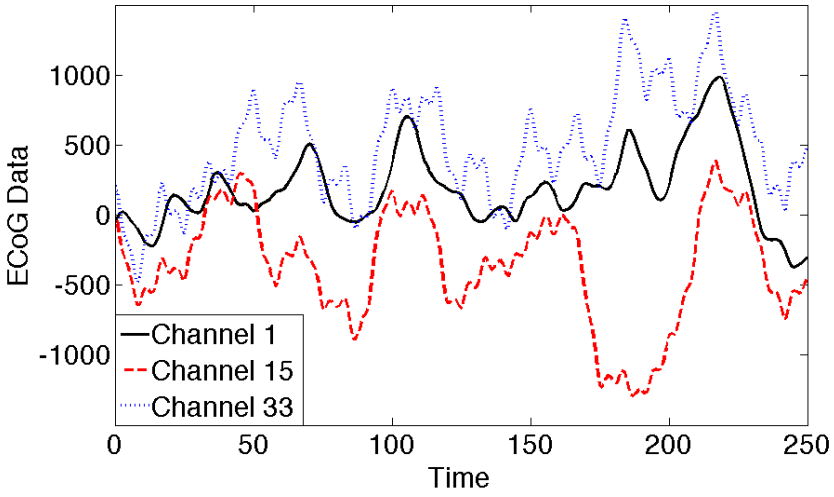


Figure 5.2: ECoG time series collected at Channel 1, 15, 33 in a single trial.

Given one trial of data, we first standardized each time series to unit variance, applied P-iPDA to the standardized data and evaluated $\text{SSE}(\lambda, \mu)$ and $\text{Fid}(\lambda, \mu)$ for all combinations of $\lambda = (0.1, 0.25, 0.5, 1, 2.5, 5, 10, 25, 50, 100, 250, 500, 1000)$ and $\lambda \cdot \mu = (0.0001, 0.001, 0.01, 0.1, 1, 10, 50, 100)$, which cover a wide range of values. Based on the outputted $\text{SSE}(\lambda, \mu)$ and $\text{Fid}(\lambda, \mu)$ for each combination, we screened out parameters with either too large $\text{SSE}(\lambda, \mu)$ or $\text{Fid}(\lambda, \mu)$ and those resulting in either too many, say d, clusters or only 1 cluster. Then we conducted RLOOCV on the rest pairs of parameters to select final $(\lambda, \lambda \cdot \mu)$. We applied this selection procedure to five randomly chosen trials, collected at five different periods, respectively, of the ECoG recording process, ranging from the beginning to the end. We found that the same parameters $\lambda = 0.25$ and $\lambda \cdot \mu = 0.01$ were selected. As such, we used the same pair of penalty parameters for analyzing data across different trials. We want to stress that though brain activities may vary across trials, this does not necessarily mean that the corresponding penalty parameters selected would vary significantly. In fact, the selection of penalty parameters does not directly depend

on the temporal values of the state functions, but rather the SNR of the data and the most significant causal interactions among different components, which may be stable across trials. In this application, there are two potential reasons for identical penalty parameters being selected by RLOOCV. First, the parameter λ - used to control the roughness of the fitted curves - depends most on the SNR of the data. Since the SNR of ECoG data is consistently high, smaller values of λ are consistently chosen, inducing a weak regularization effect. Second, the Potts penalty parameter is used to balance the ODE model size and fitting errors, and its value depends on the significance of the directional effects between components and/or the strength of the association between the instantaneous changes of components state functions and the functions themselves. Even if the state functional curves vary across trials, the most significant connections between components can still be stable. An analogy is a Markov chain with a constant transition probability but temporally varying states. Since here we study connectivity within a small brain area involved in a basic brain auditory function, it is very likely that the most significant interactions among brain components are stable (Flinker et al., 2010). Consequently, very similar (or identical) values of penalty parameters are selected.

5.2.2 Application of P-iPDA

We applied P-iPDA to each trial independently with $(\lambda, \lambda \cdot \mu) = (0.25, 0.01)$, allowing the cluster structure and model parameters to vary across trials. The colored matrix in Figure 5.3 summarizes the percentage of pairs of channels being clustered together across 246 clustering results, each obtained with one trial of data. The color scale is arbitrary, with dark red indicating high percentage and dark blue indicating low percentage. Based on this matrix, we constructed networks of closely-connected

regions, and presented them in Figure 5.4 with different thresholds on clustering frequencies. In the figure each node represents one recording channel and each edge between two channels respectively indicates that they were clustered into the same module by P-iPDA more than 90% and between 70% and 90% of trials. We found that channels 33-46 and 17 are most closely connected, with clustering percentage higher than 90% (corresponding to the darkest red areas in Figure 5.3). This is possibly because these channels all reside in the same brain local area, inferior frontal lobe. Then the connections among them are "local" and thus strongest. As shown in right panel of Figure 5.4, the auditory responsive regions, especially channels 15 and 16, which are located at adjacent sites along the posterior superior temporal gyrus and inferior parietal lobe, proximal to auditory cortex, are closely connected to the inferior frontal lobe. This result is in keeping with the findings that the inferior frontal lobe is involved in auditory processing, specifically phonological and syntactic processing (Burton, 2001), and music perception (Steven et al., 2006). In addition, the small clustering frequencies between channels 1-6, 9-10 and 18 in the epileptic area and channels 14-16 in auditory cortex (the dark blue areas in Figure 5.3) indicate that there is no or very weak interaction between them. This is possibly because the brain sub-network involved in the auditory function was unaffected by the activity in brain epileptic areas during the data collection.

Figure 5.5(a) and 5.5(b) show the average of $\hat{A}$ and $\hat{B}$ estimated by P-iPDA across all trials. Row i ($1 \leq i \leq d$) of $\hat{B}$ and $\hat{A}$ respectively represents interactive effects exerted by other channels on channel i with and without tone stimuli, and column i of matrices $\hat{B}$ and $\hat{A}$ respectively represents interactive effects exerted by channel i on other channels with and without tone stimuli. The columns corresponding to channels 14-16, the auditory-responsive regions, had values close to zero in the averaged $\hat{A}$, indicating that no notable effects of the three channels

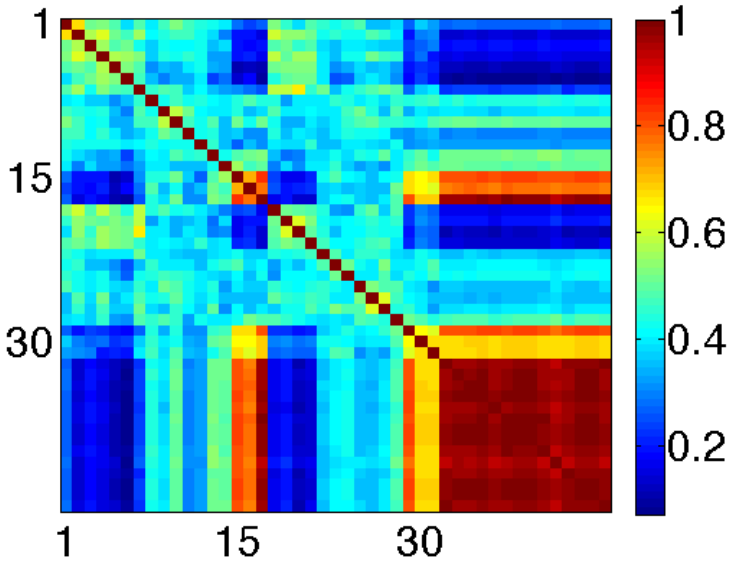


Figure 5.3: The cluster matrix for all channel pairs by 1000 Hz tone stimulus. Each element in the symmetric matrix is the percentage of two regions clustered together by P-iPDA across 246 1000 Hz trials.

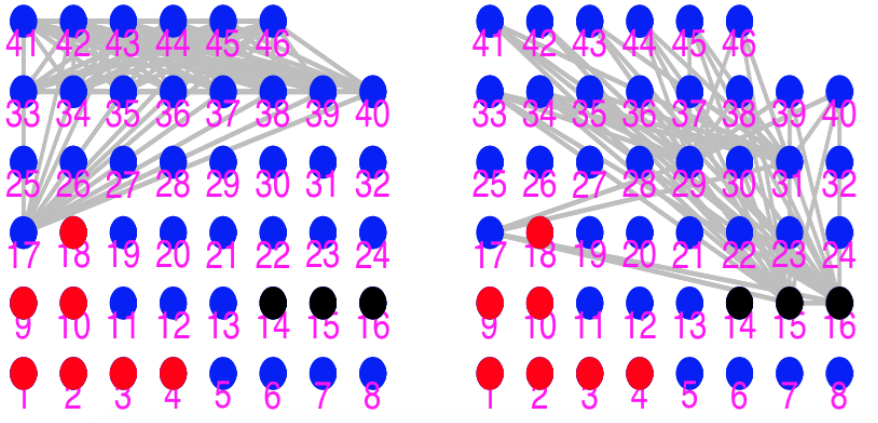
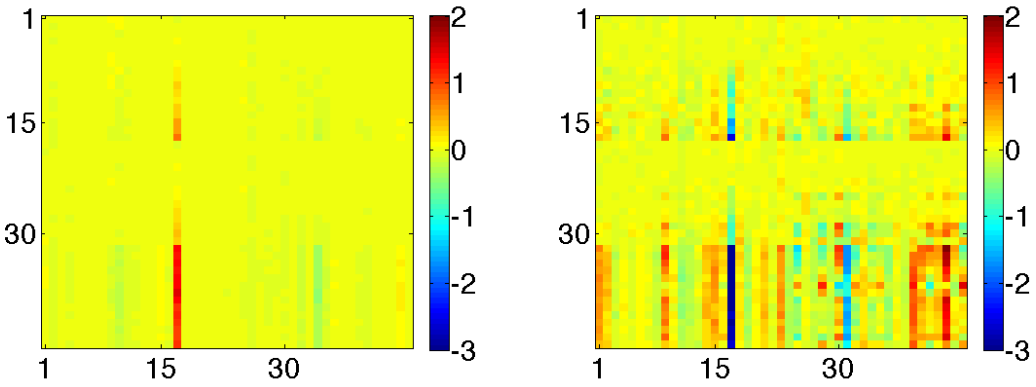


Figure 5.4: Networks constructed based on the clustering matrix in Figure 5.3. Each node represents one recording channel, the red ones are epileptic areas, and the black are auditory responsive areas. Each edge between two channels in left and right panel indicates that they are clustered into the same module by P-iPDA in more than 90%, and between 70% and 90% of trials, respectively.



(a) Average $\hat{A}$ estimated by P-iPDA (b) Average $\hat{B}$ estimated by P-iPDA

Figure 5.5: The averaged estimates of A and B across 246 trials by P-iPDA in the real data analysis.

were observed over other channels when tone stimulus was not evoked. The effect of channel 17 over other channels stood out in $\hat{A}$, revealing that channel 17, located in the inferior frontal lobe, strongly affected all three auditory-responsive electrodes in the first module. Moreover, estimates of A from each of the 246 trials show that the effect of channel 17 was stable over time. This suggests that although channel 17 showed no identifiable auditory response itself, it may serve to monitor activity in those auditory responsive sites located more posteriorly. The top-down monitoring role of the frontal lobe has previously been reported by Stuss and Levine (2002).

Chapter 6

Simulation Studies

In real data analysis we only applied P-iPDA to the ECoG data because both iPDA and L-iPDA were too slow and inaccurate. In this chapter we will present all three algorithms and compare their strength and weakness in different scenarios. The first example shows how each algorithm works in detail and answers many computation detail questions. Each of the other examples focuses on one special module structure or one particular issue.

6.1 The First Example

The first example is a toy example which consists of only four neuronal state functions. Since the dimension is computationally manageable for each algorithm, so each algorithm is able to produce good estimates with low MSE. We demonstrate such a toy example at the beginning for two reasons: (1) It's more readable and understandable for readers to show in detail how each algorithm works and how we select parameters in a low dimensional case, and future examples will only provide results without detail; (2) For some issues a low dimensional example is already

fully capable to show the disadvantages and advantages of each algorithm. We also did corresponding experiments for high dimensional cases but obtain similar conclusions, so we present the discussions here in a low dimensional case, not because they only work for low dimensional case, but because it's easier to present. So many discussions in this example are actually a synthesis of many other simulation examples.

This example is set up as follows. The observations $\boldsymbol{y}(t_i)$'s are measured from 0 to 250. The stimulus function $u(t)$ is set to be the same as in real data analysis, i.e.,

$$u(t) = \begin{cases} 0, \text{ if } 0 \le t < 100 \\ 1, \text{ if } 100 \le t \le 150 \\ 0, \text{ if } 150 < t \le 250 \end{cases}$$

The actual Θ and initial point are

$$A = \begin{bmatrix} -0.478 & -0.601 & 0 & 0 \\ 0.428 & 0.478 & 0 & 0 \\ 0 & 0 & -0.119 & -0.021 \\ 0 & 0 & 0.986 & 0.119 \end{bmatrix}, \quad B = 2 \cdot A,$$

$$C = \begin{bmatrix} 0.06 \\ 0 \\ -0.06 \\ 0.03 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \\ 0.08 \\ 0 \\ 0.03 \end{bmatrix}, \quad x_0 = \begin{bmatrix} -0.4 \\ 0.7 \\ -0.4 \\ 0.2 \end{bmatrix}.$$

As we can see from the definition of A and B, there are two clusters each of which has two neuronal state functions. You can also tell the modules from the actual

observations that are shown in Figure 6.1, $y_1(t)$ and $y_2(t)$ have a relatively low frequency while $y_3(t)$ and $y_4(t)$ have a relatively high frequency. The signal-to-noise ratio is chosen as 10. More specifically, the error $\epsilon_i(t)$ is generated by $N(0, \sigma^2_{\epsilon_i})$ where $\sigma^2_{\epsilon_i}$ is $\text{Var}(x_i)/10$. We set the signal-to-noise ratio to 10 because the estimated signal-to-noise ratio of real data, $\text{Var}(\hat{x}_i(t))/\text{Var}(\hat{\epsilon}_i(t))$, is above 10 across all trials and channels.

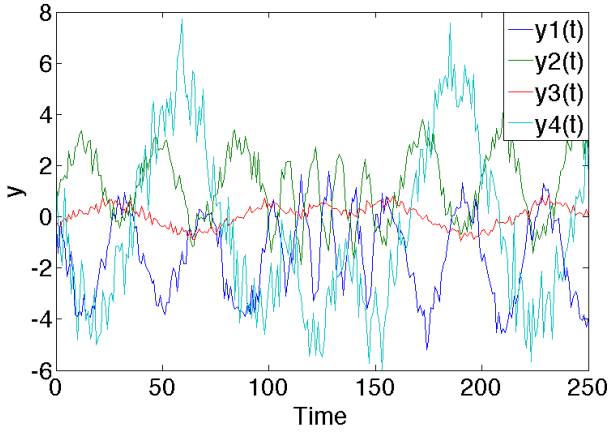
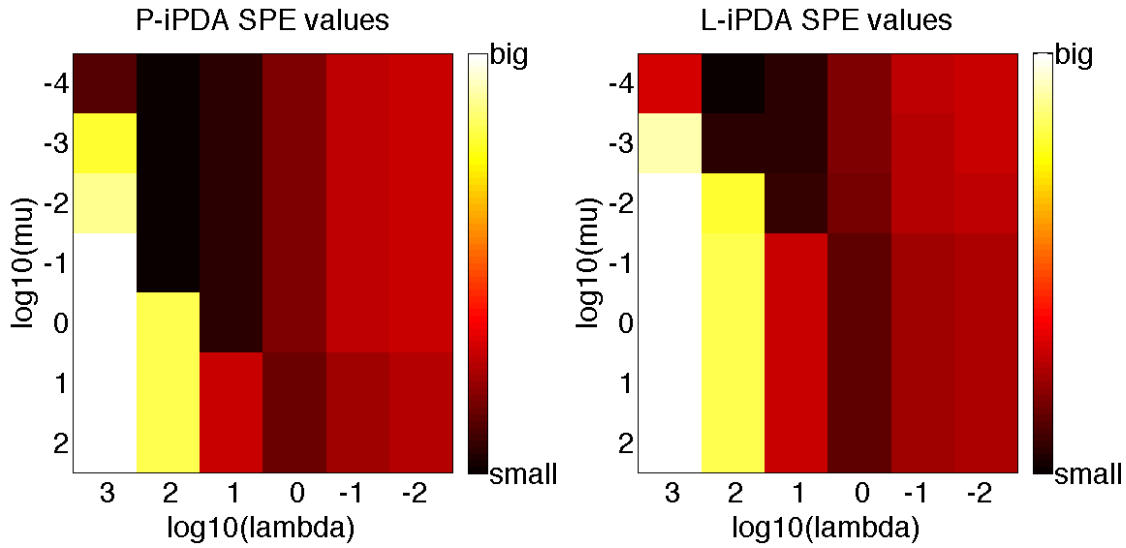


Figure 6.1: Observations $y(t)$ of Example 1. Data are observed at each millisecond from 0 to 250. In this toy example $x_1(t)$ and $x_2(t)$ are in one module and the other two are in the other module.

Because of the low dimensional we are able to perform cross-validation across a large pool of candidates without the prescreening process. Figure 6.2 (a) and (b) show the cross-validation SPE (defined in Equation 3.11) for P-iPDA and L-iPDA respectively. Dark color represents small value and light color represents large value. Since λ controls the roughness of curves, it should only be influenced by the signal-to-noise ratio but not the extra penalty like Potts or lasso, so we can use the λ selected by cross-validation of P-iPDA and do not need to run cross-validation for iPDA. The cross-validation results also verify that both P-iPDA and L-iPDA select the same λ. For P-iPDA, the optimal μ could be any value between 10^{-4} and 0.1,

since these μ's produce the same module structure. For L-iPDA, the optimal μ is selected to be the smallest μ.



(a) Cross-validation SPE by P-iPDA (b) Cross-validation SPE by L-iPDA

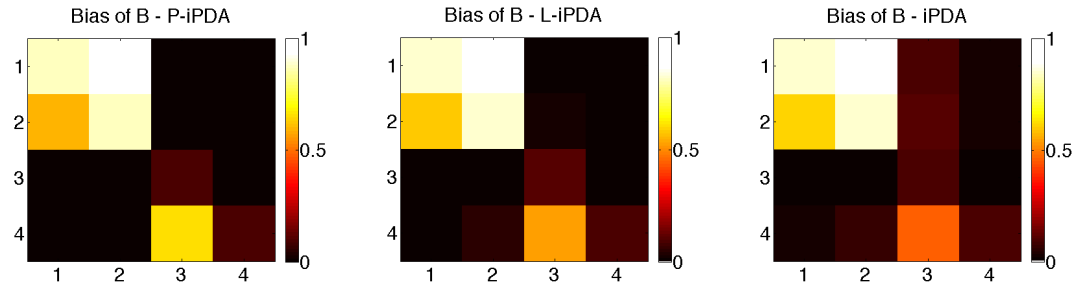
Figure 6.2: Cross-validation results for P-iPDA and L-iPDA. In this example λ ranges from 10^{-2} to 10^{3}, and μ ranges from 10^{-4} to 10^{2}. Each mosaic represents the SPE value by using the corresponding λ and μ. Darker color means small SPE and light color means large SPE. We find that when $\lambda = 100$ and $\mu \leq 0.1$ SPE of P-iPDA achieves minimum, and when $\lambda = 100$ and $\mu = 10^{-4}$ SPE of L-iPDA achieves minimum.

Comparison of sparsity, bias and variance

Now we set $(\lambda, \mu) = (100, 0.1)$, $(100, 10^{-4})$ and $\lambda = 100$ for P-iPDA, L-iPDA, and iPDA respectively, and generate $\boldsymbol{y}(t)$ with signal-to-noise ratio 10 100 times. In this example, P-iPDA is the only algorithm that can produce sparse estimate, and it reaches the module structure $M = (1, 1, 2, 2)$ only after two iterations. So if we only cares about the module structure not the detailed estimates of Θ P-iPDA is extremely fast. By its nature iPDA fails to sparsify the network. The L-iPDA also fails to sparsify the network in this example because of the tradeoff between sparsity

and accuracy. We face this similar dilemma in other examples for L-iPDA. In this example if $\lambda = 100$ and $\mu = 0.1$ L-iPDA can obtain the same module structure as P-iPDA. By shrinking off-block diagonal elements to zero, L-iPDA also penalizes block diagonal elements hence introduces bias, which makes estimates inaccurate. That's why cross-validation favors small μ's. So if our ultimate goal is clustering instead of prediction, μ cannot be selected from cross-validation for L-iPDA. One way is to select the optimal λ (from P-iPDA), and tune the μ to achieve a certain level of sparsity (for example, on par with P-iPDA).

The mean and variance for B (results for A, C and D are similar) are shown in Figure 6.3 and Figure 6.4. From Figure 6.3 the bias of P-iPDA is slightly less than L-iPDA and iPDA. But by imposing a module structure P-PDA controls its variance better than L-iPDA and iPDA as shown in Figure 6.4.

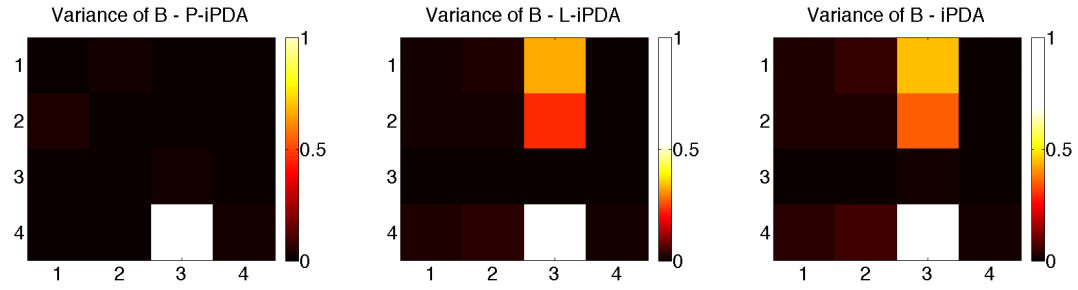


(a) P-iPDA bias (b) L-iPDA bias (c) iPDA bias

Figure 6.3: Absolute value of bias of B using three algorithms.

Smoothness parameters: number of basis p and λ

It's straightforward that the number of spline bases controls the variance of fitted curve. The parameter λ, as well as number of spline bases, also controls the smoothness of fitted data. Figure 6.5 demonstrates how λ could affect the fitting. In the figure, the black dashed line is the true $x_1(t)$. The red line is the fitted $x_1(t)$ under $\lambda = 10^{-3}$. As we can see unnecessary turbulence is incurred. The blue line is



(a) P-iPDA variance (b) L-iPDA variance (c) iPDA variance

Figure 6.4: Variance of B using three algorithms.

the fitted $x_1(t)$ under $\lambda = 10^3$. Here λ is too large to capture the dynamics during stimulus period. Another two extreme cases would be when λ goes to infinity, no nonzero derivative can be tolerated so the fitted value forms a horizontal line, and when λ goes to zero, SSE dominates the minimizing function so the fitted data will be an interpolation of observed points.

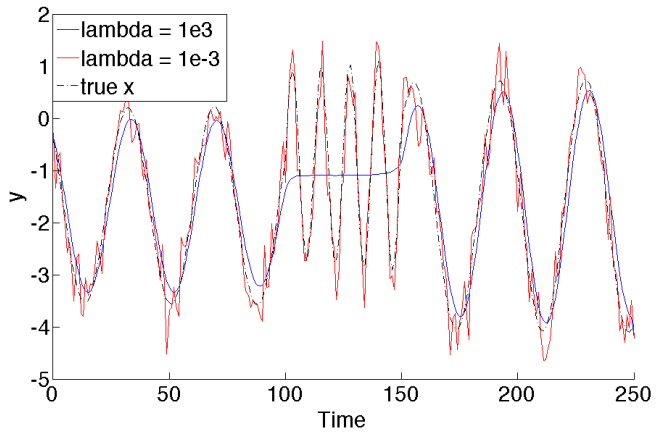


Figure 6.5: Compare fitted $x_1(t)$ by different λ's. Large λ overly smoothes the data, for example, in this example the dynamics during stimulus period are "punished" off by large λ. Small λ overly fitted fitted the data.

The inevitable problem: the trap of local minimum

The P-iPDA algorithm is a greedy algorithm, it makes the local optimal choice at every stage. Although we can guarantee the algorithm will converge after enough

iterations, we cannot guarantee the solution is the global optimal solution. It is similar as doing stepwise selection in linear regression setting, in which we cannot escape from the local minimum trap. We have proposed a few methods to deal with the issue in Section 3.5. Here we use this 4 dimension example to show how this problem arises. If we set $\mu = 0$, which put no penalty for clustering multiple nodes. So theoretically it's equivalent as iPDA since a module structure that merges two smaller clusters to one cluster will always have a less **H** value. But let's see how P-iPDA updates the module structure M. At the beginning, every system component is in its whole module with just itself in it, so $M^0 = (1, 2, 3, 4)$. At the first iteration, it will cluster either node $(1, 2)$ or $(3, 4)$, depending on whichever gives a better **H**. Suppose $M^1 = (1, 1, 2, 3)$ has been selected. At the second iteration clustering node 2 and node 3 together will improve the current dynamics significantly so $M^2 = (1, 1, 2, 2)$. This is a local optimum because all the other neighbors cannot perform better than M^2. Specifically, $\mathcal{N}(M^2) = (3, 1, 2, 2), (2, 1, 2, 2), (1, 2, 2, 2), (1, 1, 3, 2), (1, 1, 1, 2), (1, 1, 2, 1)$. The only module structure under $\mu = 0$ that is better than $M^2 = (1, 1, 2, 2)$ is $M^{opt} = (1, 1, 1, 1)$. However M^{opt} is not in the neighborhood of M^2 hence is not reachable by the model updating scheme. Hence P-iPDA will stop updating module labels after step 2. The converged module label M^2 is a local minimum but not the global optimal solution.

6.2 Example 2: Two Clusters with Medium Size

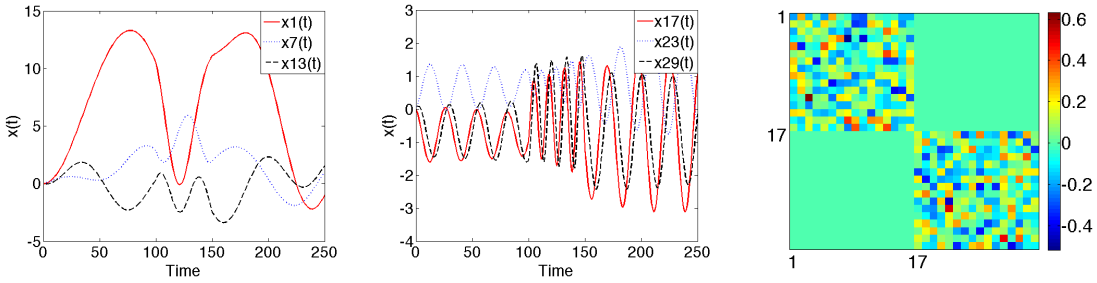
In the second example we extend our algorithm to high dimension. Example 2 uses data generated from a dynamic system of 32 channels. To mimic the real data, the stimulus function $u(t)$ is identical to the one in the real data and Example 1, and

there are two functionally-segregated sub-networks, each containing 16 channels. For simplicity, we use identical A and B. The parameters are chosen such that $x(t)$ have periodic temporal variations, and do not monotone increase or decrease over time, as shown in Figure 6.6. Also, channels in the two modules have different frequencies of temporal variation, such that $x(t)$ in two clusters are not linearly dependent. All the channels in each module are pair-wisely connected, that is, none of the elements of A and B within each module is zero. The values of A/B are shown in Figure 6.6(c). The signal-to-noise ratio is also set to be 10. The parameter setting is summarized in Table 6.1.

Table 6.1: Parameter setting for simulation example 2

# of observations n	251
# of neuronal state functions d	32
# of knots $p-2$	101
# of points used for Riemann Sum	251
Network structure	Two clusters with equal size
Initial guess Θ_0	$\mathbf{0}$
Stimulus function $u(t)$	$\mathbf{1}_{100\leq t\leq 150}$
Standardize	Yes

We will use this example to show how we prescreen λ and μ from a large pool of candidates. The prescreen process was introduced in Section 3.4.2, and the result table was presented in Appendix Table A.4. The pool of (λ, μ) is summarized in the first two columns, where λ ranges from 0.1 to 500 and μ ranges from 0.001 to 100. We compare the SSE and Fid (Equation 3.2) for each combination of λ and μ. The criterion is as follows. If one combination of λ and μ can improve both SSE and Fid, or can improve either the SSE or Fid tremendously without sacrificing the other



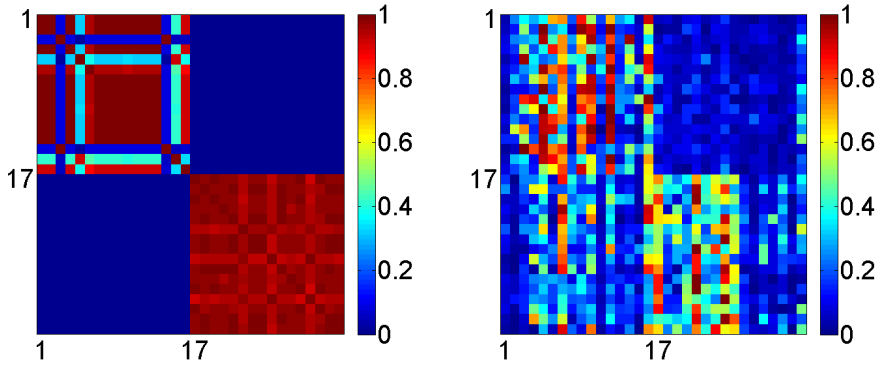
(a) First module (b) Second module (c) The matrix A/B

Figure 6.6: Panel (a) and (b) show temporal changes of the simulated state x(t) at three channels in the first and second module separately for Example 2. Panel (c) shows the value of A/B.

too much, then we adopt the combination. If two μ's produce comparable SSE and Fid for the same λ, then we choose the larger μ because we prefer a simpler model. Then after the prescreen process we apply RLOOCV to select the best combination based on SPE. From Table A.4, we first detect that large λ is more appropriate because large λ sacrifices SSE a little but reduce Fid considerably. For example, if you look at $(0.1, 0.001)$ versus $(50, 0.25)$, the latter only increase SSE less than twice, but reduce Fid approximately 100 times. In this case we don't even need to run cross-validation, because use the same argument, $\lambda = 50$ betters all the smaller λ's. $\lambda = 100$ dominates $\lambda = 50$ since it improves both SSE and Fid, $\lambda = 250$ outperforms $\lambda = 100$ again because it increase SSE only 20% but reduce Fid by 1/3. For $\lambda = 250$, $\mu = 0.0625, 0.125$ and 0.25 don't make a difference so we pick the largest one. Hence we select the combination $(250, 0.25)$.

After we select the optimal combination of parameters, we generate 100 trials and analyze the cluster accuracy and stability. Figure 6.7(a) and (b) summarize the percentage of each pair of channels clustered into the same module by P-iPDA and L-iPDA respectively. Figure 6.8(b) and (c) respectively present the corresponding networks using different thresholds on frequencies: higher than 90% and between

70% and 90%. The true positive rate of P-iPDA (the frequency of correctly detecting nonzero values of A and B) is 81.5%, and false positive rate (the frequency of estimating zero values of A and B incorrectly nonzero) is 0. Overall, P-iPDA successfully detected two clusters, though it occasionally missed clustering one or two channels, which is possibly due to the multicollinearity among $\boldsymbol{x}(t)$ in the same cluster. We also plot the clustering result by L-iPDA, as we can imagine for this type of network, L-iPDA completely fails to detect network structures.



(a) Clustering matrix by P-iPDA (b) Clustering matrix by L-iPDA

Figure 6.7: The (i, j)th (same as the (j, i)th) element of the 32×32 symmetric matrix in Panel (a) represents the percentage of channels i and j in Example 2 clustered into the same module by P-iPDA and L-iPDA among 100 simulations.

We evaluated and compared the biases and variances of $\hat{A}$ and $\hat{B}$ estimated by P-iPDA and iPDA, which are summarized in Table 6.2. For easy comparison, we used the same λ in the two methods. In Example 2, P-iPDA produced estimates with slightly smaller biases and much smaller variances than those by iPDA. The reasons are two folds. First, if $x(t)$ is known and P-iPDA correctly identifies all interactive channels, the regression models outputted from P-iPDA, where $dx_i(t)/dt$ of each channel i is the response, and $x(t)$ in the same cluster as i are the predictors, are equivalent to those in iPDA. In addition, since estimated $\hat{\boldsymbol{x}}(t)$ by P-iPDA and iPDA based on identical λ take similar values, the estimates of the above mentioned

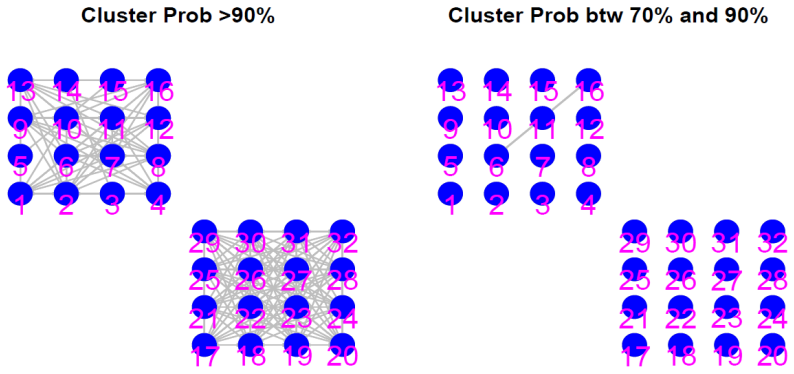


Figure 6.8: Networks constructed based on the clustering matrix in Figure 6.7(a) with different thresholds: Each node represents one recording channel and each edge respectively indicates that the effect, exerted by one region on another, is estimated non-zero by P-iPDA in more than 90% and between 70% and 90% of the 100 simulations.

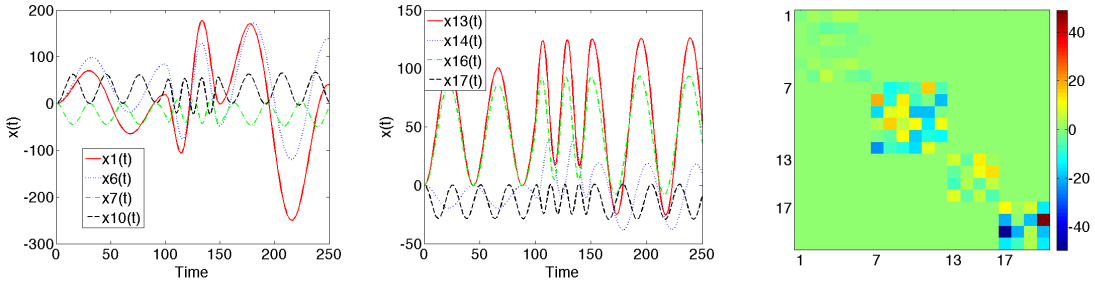
regression coefficients, i.e., A and B, by the two methods have similar means and biases. Second, with the sparsity constraint, P-iPDA uses much fewer predictors in the regression models than iPDA, and thus the ensuing estimates have much smaller variances. Other than achieving better estimation efficiency than iPDA, P-iPDA, by partitioning a large network into several independent smaller ones, also takes much less computational time.

Table 6.2: The bias and standard deviation of estimated A and B by P-iPDA and iPDA in Example 2 and Example 3

Example	Parameter	Average P-iPDA	Bias iPDA	Average P-iPDA	Std iPDA
2	A	0.08	0.10	0.26	0.36
	B	0.14	0.15	1.18	1.02
3	A	2.00	2.03	0.20	0.61
	B	2.03	2.33	0.44	4.71

6.3 Example 3: Multiple Small Clusters

In this simulation study we set up a network with 20 nodes and 4 clusters, of size 6, 6, 4, 4 respectively. The settings are the same as Example 2 but the model parameter Θ. The parameter A/B and a few observed curves are shown in Figure 6.9. All the other parameters used the same values as in Table 6.1.

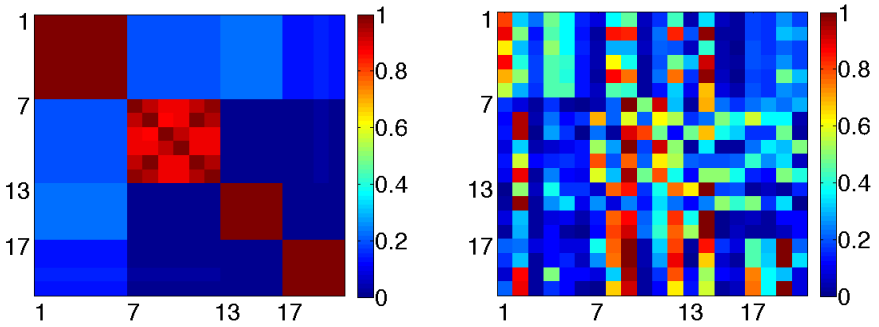


(a) First module (b) Second module (c) The matrix A/B

Figure 6.9: Panel (a) and (b) show temporal changes of the simulated state x(t) at three channels in the first and second module separately for Example 3. Panel (c) shows the value of A/B.

The parameters λ and μ were selected by a prescreen process and cross-validation. By the prescreen process only 12 candidates are selected for a further cross-validation process. Those combinations of λ and μ and the corresponding SPE are summarized in Table A.2. Without the prescreen process it will take us forever to run cross-validation across the original pool of candidates. After selecting the parameter we ran 100 trials like in Example 2. Figure 6.10(a) and (b) summarize the frequencies of each pair of components being clustered together by P-iPDA and L-iPDA across 100 simulations respectively. Figure 6.11 presents the associated networks constructed by using different thresholds on the clustering frequencies. Overall, the true positive rate of the P-iPDA in Example 2 is 97.5% and the false positive rate is 10.9%.

Comparing estimated $\hat{A}$ and $\hat{B}$ outputted from P-iPDA and iPDA, the former again achieved slightly smaller biases and much smaller variances than the latter,



(a) Clustering matrix by P-iPDA (b) Clustering matrix by L-iPDA

Figure 6.10: The (i, j)th (same as the (j, i)th) element of the 20×20 symmetric matrix in Panel (a) represents the percentage of channels i and j in Example 3 clustered into the same module by P-iPDA and L-iPDA among 100 simulations.

as shown in Table 6.2. We note that since Example 3 has a higher percentage of zero values in matrices A and B than that in Example 1, the reduction of estimation variability by P-iPDA, in comparison to iPDA, is more pronounced.

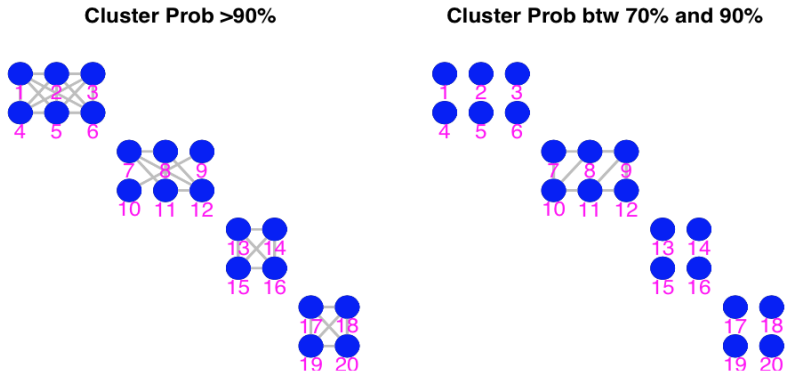
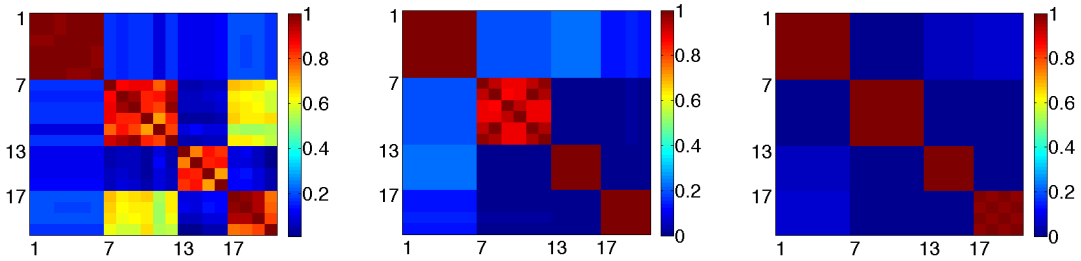


Figure 6.11: Networks constructed based on the clustering matrix in Figure 6.10(a) with different thresholds: Each node represents one recording channel and each edge respectively indicates that the effect, exerted by one region on another, is estimated non-zero by P-iPDA in more than 90% and between 70% and 90% of the 100 simulations.

6.4 Example 4: Time Series with Different Lengths

We have also investigated how the clustering results by P-iPDA vary with different lengths of time series. Using the same A, B, C, D in Example 3, two additional sets of $x(t)$ were generated, one with $T = 100$ and the other with $T = 500$. We let $u(t) = 1$ for $25 \leq t \leq 75$ in the former, and $u(t) = 1$ for $100 \leq t \leq 150$ and $400 \leq t \leq 450$ in the latter. For each set of $x(t)$, we conducted 100 simulations in the same manner as in Example 2, summarized the clustering frequencies by P-iPDA and L-iPDA for each of two sets of $x(t)$ respectively in Figures 6.12. When the length of time series is reduced by half, the true positive rate of P-iPDA is decreased to 91.8% and false positive rate is increased to 20.8%. On the other hand, when the length is doubled, the true positive rate is increased to 99.7% and the false positive rate is significantly reduced to 2.4%. Overall, the length of the data affects the false positive rate more than the true positive rate. This is possibly due to the fact that the model fitting errors are more affected by missing truly interactive channels, but less affected by including non-interactive channels, and thus P-iPDA tends to cluster more channels together when the data information is limited.

In summary, P-iPDA achieved a higher true positive rate in Examples 3 and 4 than that in Example 2. In general, P-iPDA is most effective for cluster structures consisting of small clusters: the smaller the clusters, the fewer iterations that P-iPDA takes to identify the clusters, and less computation needed for estimating the spline-basis coefficients and model parameters.



(a) P-iPDA: 100 ms (b) P-iPDA: 250 ms (c) P-iPDA: 500 ms

(d) L-iPDA: 100 ms (e) L-iPDA: 250 ms (f) L-iPDA: 500 ms

Figure 6.12: Clustering frequency by P-iPDA and L-iPDA among 100 simulations.

Chapter 7

Conclusion and Discussions

We propose a differential-equation-based dynamic model for ECoG data to study directional effects between brain regions. A new Potts model for the state equations in the DDM is introduced to automatically identify functionally segregated brain networks. The high spatial and temporal resolution of the ECoG data allows the dynamic model to have a simple structure that can accommodate a large number of brain components, unlike related DCMs in other modalities. We represent the neuronal states of brain components by B-spline bases and estimate the model by minimizing a log-likelihood-based criterion, for which we have developed an iterative optimizing algorithm. The Potts model is converted to the new Potts penalty in the penalized likelihood approach. An L_1 penalty is also considered for comparison.

Penalty parameter selection is a data-dependent process. As the ECoG recordings analyzed in this dissertation cover only a small brain area under one experimental paradigm with a simple auditory stimulus, brain auditory responses measured by ECoG tend to be stable across trials (Flinker *et al.*, 2010), and consequently very similar penalty parameters were selected. However, in more common modalities such as fMRI and EEG, significant heterogeneitydue to the longer length or larger brain

area coveredin the underlying effective connectivity across trials is widely reported (e.g. Duann *et al.*, 2002; Truccolo *et al.*, 2002; Turetsky *et al.*, 1988). In such cases, penalty parameter selection should be conducted separately for each trial, which will likely lead to different values being selected for different trials. Since parameter screening and cross validation are performed independently for each combination of parameter and each data point, this process can be parallelized.

While the proposed method is motivated from and applied to the ECoG data, the statistical methodology, particularly the PDDM, can be applied to a broad range of applications using multivariate time series. First, the Potts penalty, in fact, does not rely on the linearity of the ODE model assumed in this analysis, and can be used in any dynamic models. Second, the Potts penalty is also applicable to settings where the observation time n is smaller than $(J+1) \times d^2$, the number of parameters characterizing pairwise interactions between components, analogous to the "small n large p" paradigm. Indeed, when the number of parameters in each module is believed to be much smaller than n , one can start with the most economic PDDM in which each component forms an independent module, and thus requires the least number of parameters. Through similar optimizing procedures as that in Section 3.2, at each step one node is selected to be clustered with one existing module according to the ensuing criterion. Then the size of modules will increase and the ensuing number of modules will decrease until the criterion cannot be optimized anymore. Such a procedure is comparable to a stepwise linear regression that adds one variable at a time, and can be used in the "small n large p" paradigm as long as the number of selected variables is much smaller than n. Third, the Potts penalty can be used for time series that are observed in segments. Even when the parameters are allowed to differ between segments, the penalty can still be used to identify modules as long as they are assumed to remain the same over the time.

We fit our PDDM to the ECoG time series observed over a very short (less than 1s) period of time, different from the common practice of fitting DCMs to long time series (often in hundreds of seconds) in fMRI. Thanks to its high temporal resolution, such ECoG time series, with a large number of total observations still, offers a unique opportunity for studying effective connectivity, for the following reasons. First, a brain system may change dramatically over a short period of time, inducing non-ignorable temporal changes in parameter values, or even modules. Second, even if there is no dramatic change in the brain system over a long period of time, the underlying brain activity is likely to deviate significantly from the assumed linear system. As such, model assumptions based on first or second order Taylor approximation are relevant in the context of dynamic systems evolving over a short time. Third, analyzing short time series allows us to avoid making strong assumptions on the parameters of the PDDM, such as a negative definite parameter matrixcommonly assumed in fMRI-based connectivity studiesin order to ensure a stable system over an extended period. Nevertheless, it is still feasible to investigate brain effective connectivity using data measured over a long time. One possible approach is to first divide the data into several much shorter periods, within each of which a separate linear model is assumed. Then, identify functionally independent modules using the PDDM. Finally, within each identified module, apply nonparametric regression methods to approximate the nonlinear and unknown relationship between instantaneous changes of neuronal states with themselves and the experimental input.

The PDDM specifies two separate parameters for the connection in each of two directions between any two components within the same cluster, and the associated estimation algorithm P-iPDA clusters two components together if the connection in any one direction is strong. It is possible that the connection between some components within the same cluster is void in one direction, but strong in the other di-

rection. Our current method, however, does not evaluate the statistical significance of the directional effects and thus cannot distinguish which underlying directional effect within a cluster is void or nonzero. One potential approach to address this issue is to conduct hypothesis testing on the estimates of the directional effects from P-iPDA. This procedure must take into account of the uncertainty in identifying the clusters by P-iPDA, which is non-trivial in practice. Another potential approach is to impose both Potts penalty and L_1 penalty (Tibshirani, 1996) on the parameters within clusters in the log-likelihood criterion. Though achieving simultaneous clustering and sparsity within clusters, this approach is computationally demanding with three penalty parameters to be selected, and thus may require more iterations to converge. These will be the focus of our future directions in high-dimensional ODE model estimation.

We also discussed how to improve P-iPDA in Section 3.5 and 4.3. Apart from that, there are several other directions for improving the PDDM and the P-iPDA algorithm. First, the spatial information of brain regions can be incorporated into the Potts model, so that spatially-close regions are more favored to be clustered into one module. Second, our current practice of using identical penalty parameters for all the regions may not be suitable for brain networks comprising modules with distinct interactive patterns. One potential solution is to use adjustable and region-dependent penalty parameters as mentioned in Section 3.5. Another possibility is to modify P-iPDA such that the already-identified clusters can be removed from the optimizing function, and thus do not affect the estimation of other clusters. Third, the PDDM estimation is formulated as an optimization problem in the dissertation; statistical inference such as confidence interval construction and hypothesis testing on the model parameters is not straightforward. As elucidated before, the Potts model defines a prior distribution for the DDM parameters, and thus inference of

the PDDM can be naturally carried out within a Bayesian framework. Finally, the PDDM can be modified to allow for very few channels that have interactive activity with several clusters and act as the“hub” of the brain network.

Appendix A

Tables of Parameter Prescreen

Table A.1: Real data: cross-validation

λ	μ	SPE
0.1	0.01	31.794
0.25	0.01	24.225
0.5	0.1	42.354
1	0.1	39.288
2.5	0.1	41.158
5	0.1	47.941

Table A.2: Example 3: cross-validation

λ	μ	SPE
50	0.001	116.1
50	0.01	115.85
50	0.1	115.75
50	1	116.21
100	0.001	122.6
100	0.01	122.54
100	0.1	122.24
100	1	122.32
250	0.001	149.11
250	0.01	149.3
250	0.1	148.67
250	1	151.5

Table A.3: Real Data: parameter prescreen process

λ	μ	SSE	Fid	Potts	λ	μ	SSE	Fid	Potts
0.1	0.001	3.6383	9.9764	396	10	0.001	270.85	92.169	361.6
0.1	0.01	22.154	104.88	613.6	10	0.01	270.85	92.169	361.6
0.1	0.1	22.072	323.92	1.6	10	0.1	190.12	34.108	820.4
0.1	1	22.071	325.99	0	10	1	296.81	18.721	211.2
0.1	10	22.071	325.99	0	10	10	564.55	79.803	0
0.25	0.001	22.397	271.1	396	25	0.001	487.98	63.07	361.6
0.25	0.01	24.046	141.45	782.4	25	0.01	487.98	63.07	361.6
0.25	0.1	29.871	236.05	71.2	25	0.1	396.04	39.318	628.8
0.25	1	28.232	289.65	0	25	1	408.13	5.792	262.4
0.25	10	28.232	289.65	0	25	10	1095.5	40.102	5.6
0.25	50	28.232	289.65	0	25	50	1123.4	44.528	0
0.5	0.001	28.387	238.99	396	50	0.001	952.29	35.266	328.8
0.5	0.01	29.261	166.53	814.4	50	0.01	952.29	35.266	328.8
0.5	0.1	38.405	97.818	315.6	50	0.1	728.87	21.009	582.4
0.5	1	41.846	252.26	0	50	1	616.01	3.3617	234
0.5	10	41.846	252.26	0	50	10	1553.8	18.414	18.4
0.5	50	41.846	252.26	0	50	50	1746.9	26.854	0
1	0.01	41.535	206	396	100	0.01	1327.7	17.18	452.8
1	0.1	51.422	53.349	482.4	100	0.1	805.37	7.1569	668.8
1	1	72.628	204.3	3.2	100	1	858.79	1.8512	225.6
1	10	71.514	211.2	0	100	10	1967.8	7.7657	28.4
1	50	71.514	211.2	0	100	50	2588.2	14.907	0
2.5	0.0001	69.855	170.16	396	250	0.0001	1993.2	9.2735	463.2
2.5	0.001	69.855	170.16	396	250	0.001	1993.2	9.2735	463.2
2.5	0.01	69.855	170.16	396	250	0.01	1988.1	9.2771	439.2
2.5	0.1	78.194	75.698	549.2	250	0.1	770.76	0.31011	736.4
2.5	1	145.99	103.72	59.6	250	1	1244.3	0.76338	214.4
2.5	10	161.1	156.46	0	250	10	2940.9	2.961	25.6
2.5	50	161.1	156.46	0	250	50	4033.1	5.5781	0
2.5	100	161.1	156.46	0	250	100	4033.1	5.5781	0
5	0.0001	146.48	124.86	396	500	0.0001	3008.8	3.339	516.8
5	0.001	146.48	124.86	396	500	0.001	2353.3	2.4517	731.6
5	0.01	146.48	124.86	396	500	0.01	2351.1	2.4526	715.6
5	0.1	122.34	49.656	748.4	500	0.1	1915	0.12418	716.8
5	1	215.16	44.053	140.4	500	1	1772	0.38699	182.4
5	10	306.88	115.92	0	500	10	3891.9	1.3199	19.6
5	50	306.88	115.92	0	500	50	5193.6	2.1959	0
5	100	306.88	115.92	0	500	100	5193.6	2.1959	0

Table A.4: Example 2: Parameter screen process.

λ	μ	SSE	Fid	Potts	λ	μ	SSE	Fid	Potts
0.1	2.5E-05	430.41	166.24	219.2	5	0.00125	584.68	15.622	74.8
0.1	5.0E-05	430.41	166.24	219.2	5	0.0025	584.68	15.622	74.8
0.1	0.0001	430.41	166.24	219.2	5	0.005	584.68	15.626	74.8
0.1	2.5E-4	430.41	166.24	219.2	5	0.0125	584.68	15.626	74.8
0.1	0.0005	430.41	166.24	219.2	5	0.025	585.25	16.127	65.6
0.1	0.001	430.41	166.24	219.2	5	0.05	585.54	16.297	61.6
0.1	0.0025	430.44	168.66	181.2	5	0.125	585.56	16.318	60.4
0.1	0.005	430.49	175.92	84.4	5	0.25	585.99	16.604	56
0.25	6.25E-05	437.38	131.56	180.4	10	0.0025	639.19	7.3994	84
0.25	1.25E-4	437.38	131.56	180.4	10	0.005	639.19	7.3994	84
0.25	2.5E-4	437.38	131.56	180.4	10	0.01	639.19	7.3994	84
0.25	6.25E-4	437.38	131.56	180.4	10	0.025	639.19	7.3994	84
0.25	0.00125	437.38	131.56	180.4	10	0.05	639.16	7.4221	81.2
0.25	0.0025	437.37	131.17	193.2	10	0.1	641.28	7.7012	69.6
0.25	0.00625	437.58	137.92	84	10	0.25	642.81	8.0363	60.4
0.25	0.0125	437.6	139.54	66.8	10	0.5	642.72	8.3042	54
0.5	1.25E-4	450.89	95.007	165.6	50	0.0125	772.13	1.4354	124.4
0.5	2.5E-4	450.89	95.007	165.6	50	0.025	774.31	1.4094	122.8
0.5	0.0005	450.89	95.007	165.6	50	0.05	774.32	1.4094	122.8
0.5	0.00125	450.99	96.176	151.2	50	0.125	775.92	1.4343	108.4
0.5	0.0025	450.99	96.176	151.2	50	0.25	775.58	1.4337	105.6
0.5	0.005	451.24	99.302	94.8	50	0.5	790.38	1.5963	88.4
0.5	0.0125	451.35	100.75	76.8	50	1.25	809.4	1.8776	58
0.5	0.025	451.46	101.62	64	50	2.5	829.2	2.0091	51.2
1	2.5E-4	477.03	63.029	98.4	100	0.025	729.5	0.34615	259.6
1	0.0005	477.03	63.029	98.4	100	0.05	728.07	0.34763	259.6
1	0.001	477.28	63.764	90	100	0.1	728.04	0.36838	252
1	0.0025	477.29	63.896	87.6	100	0.25	724.69	0.35623	226
1	0.005	477.36	64.224	82	100	0.5	722.34	0.39222	180
1	0.01	477.38	64.311	78.8	100	1	781.13	0.56157	105.6
1	0.025	477.44	65.185	64.8	100	2.5	904.41	0.97978	68
1	0.05	477.48	65.318	62.4	100	5	937.81	1.2105	48.4
2.5	6.25E-4	531.89	30.274	78.4	250	0.0625	895.86	0.11359	395.5
2.5	0.00125	531.89	30.274	78.4	250	0.125	857.81	0.12649	304
2.5	0.0025	532.09	30.513	74	250	0.25	866.25	0.1139	330.4
2.5	0.00625	532.02	30.515	73.6	250	0.625	859.16	0.14013	228.8
2.5	0.0125	532.02	30.515	73.6	250	1.25	1061.6	0.20077	128.4
2.5	0.025	532.19	30.825	66.8	250	2.5	1600.4	0.40505	73.2
2.5	0.0625	532.32	31.067	62.4	250	6.25	3672.4	1.0623	28.4
2.5	0.125	532.35	31.383	58.4	250	12.5	4322.8	2.1402	8.4

Table A.5: Example 3: parameter prescreen process when $n = 250$.

λ	μ	SSE	Fid	Potts	λ	μ	SSE	Fid	Potts
0.1	2.50E-05	269.81	134.41	94.4	5	0.00125	372.87	8.5943	85.2
0.1	5.00E-05	269.81	134.41	94.4	5	0.0025	372.87	8.5943	85.2
0.1	0.0001	269.81	134.41	94.4	5	0.005	372.87	8.5943	85.2
0.1	2.5E-4	269.81	134.41	94.4	5	0.0125	372.87	8.5943	85.2
0.1	0.0005	269.81	134.41	94.4	5	0.025	372.87	8.5943	85.2
0.1	0.001	269.81	134.41	94.4	5	0.05	373.03	9.2058	80.4
0.1	0.0025	269.81	134.41	94.4	5	0.125	374.31	9.8929	70.4
0.1	0.005	269.8	136.12	82	5	0.25	374.47	10.506	65.6
0.25	6.25E-05	275.36	103.89	88	10	0.0025	394.46	3.5042	110.8
0.25	1.25E-4	275.36	103.89	88	10	0.005	396.16	3.8001	110.4
0.25	2.5E-4	275.36	103.89	88	10	0.01	396.16	3.8001	110.4
0.25	6.25E-4	275.36	103.89	88	10	0.025	396.16	3.8001	110.4
0.25	0.00125	275.36	103.89	88	10	0.05	396.16	3.8001	110.4
0.25	0.0025	275.36	103.89	88	10	0.1	397.66	4.0272	100.4
0.25	0.00625	275.36	103.89	88	10	0.25	399.47	4.0332	81.2
0.25	0.0125	275.28	104.95	80.4	10	0.5	400.2	4.0416	71.6
0.5	1.25E-4	286.93	70.587	97.6	50	0.0125	446.06	0.22601	108
0.5	2.5E-4	286.93	70.587	97.6	50	0.025	446.06	0.22601	108
0.5	0.0005	286.93	70.587	97.6	50	0.05	446.78	0.22737	93.6
0.5	0.00125	286.93	70.587	97.6	50	0.125	445.11	0.23416	84
0.5	0.0025	286.93	70.587	97.6	50	0.25	446.47	0.23232	84
0.5	0.005	286.92	70.608	97.6	50	0.5	445.41	0.231	84
0.5	0.0125	287.05	71.219	88	50	1.25	491.62	0.51839	68.8
0.5	0.025	287.03	73.103	76.8	50	2.5	579.6	2.0589	52.8
1	2.5E-4	306.78	44.994	84.8	100	0.025	529.99	0.13102	158
1	0.0005	306.78	44.994	84.8	100	0.05	529.99	0.13102	158
1	0.001	306.8	45.331	80	100	0.1	529.99	0.13102	158
1	0.0025	306.8	45.331	80	100	0.25	531.71	0.13969	122
1	0.005	306.8	45.331	80	100	0.5	527.81	0.13839	96.4
1	0.01	306.8	45.331	80	100	1	529.08	0.13828	86.4
1	0.025	306.71	45.203	80	100	2.5	827.42	1.4466	55.2
1	0.05	307.38	46.81	56	100	5	1191.3	2.8669	27.6
2.5	6.25E-4	343.19	18.946	83.2	250	0.0625	732.93	0.047557	268.4
2.5	0.00125	343.19	18.946	83.2	250	0.125	737.87	0.047999	251.6
2.5	0.0025	343.19	18.946	83.2	250	0.25	723.71	0.055587	184.4
2.5	0.00625	343.19	18.946	83.2	250	0.625	731.06	0.077015	109.6
2.5	0.0125	343.19	18.946	83.2	250	1.25	914.69	0.15745	79.6
2.5	0.025	342.86	19.689	78.4	250	2.5	1523.6	0.37636	54
2.5	0.0625	342.86	19.689	78.4	250	6.25	2378.4	1.1664	16.8
2.5	0.125	343.39	19.817	68.8	250	12.5	3048.2	1.7862	1.2

Table A.6: Example 3: parameter prescreen process when $n = 100$.

λ	μ	SSE	Fid	Potts	λ	μ	SSE	Fid	Potts
0.1	0.01	4.6704	138.44	203.4	10	0.01	104.35	1.4306	115.6
0.1	0.1	3.8163	219.89	14.6	10	0.1	105.25	1.4763	111.6
0.1	1	3.6456	263.62	0	10	1	118.38	2.3632	56.2
0.1	10	3.6456	263.62	0	10	10	319.01	24.349	1.8
0.1	50	3.6456	263.62	0	10	50	340.55	26.857	0
0.1	100	3.6456	263.62	0	10	100	340.55	26.857	0
0.25	0.0001	15.674	70.374	209.8	25	0.0001	133.95	0.33812	160.4
0.25	0.001	15.738	71.462	204.6	25	0.001	133.95	0.33812	160.4
0.25	0.01	16.004	74.079	187.2	25	0.01	134.81	0.33707	155
0.25	0.1	16.412	119.49	42.8	25	0.1	136.44	0.34488	128.2
0.25	1	14.05	179.6	3.6	25	1	194.6	0.76961	55.8
0.25	10	14.137	200.56	0	25	10	590.08	9.5181	0.6
0.25	50	14.137	200.56	0	25	50	603.69	9.8998	0
0.25	100	14.137	200.56	0	25	100	603.69	9.8998	0
0.5	0.0001	29.612	36.999	184.6	50	0.0001	179.09	0.10603	234.6
0.5	0.001	29.713	37.456	182.4	50	0.001	179.09	0.10603	234.6
0.5	0.01	30.113	38.049	175.4	50	0.01	179.14	0.10464	232.8
0.5	0.1	35.633	52.99	64.6	50	0.1	183.63	0.12017	152.2
0.5	1	31.693	123.96	5.6	50	1	266.8	0.36372	52.4
0.5	10	32.02	150.69	0	50	10	813.78	3.7626	0
0.5	50	32.02	150.69	0	50	50	813.78	3.7626	0
1	0.0001	46.342	17.561	159.4	100	0.0001	243.14	0.042796	199.6
1	0.001	46.342	17.561	159.4	100	0.001	243.14	0.042796	199.6
1	0.01	48.956	18.547	128.6	100	0.01	242.55	0.043045	199.6
1	0.1	54.352	19.786	80.6	100	0.1	250.03	0.054466	109.2
1	1	59.524	64.838	12	100	1	434.86	0.20395	39.8
1	10	61.493	109.22	0	100	10	988.17	1.2027	0
1	50	61.493	109.22	0	100	50	988.17	1.2027	0
2.5	0.0001	68.991	6.4967	127.2	250	0.0001	366	0.013949	179.4
2.5	0.001	68.991	6.4967	127.2	250	0.001	366	0.013949	179.4
2.5	0.01	72.08	6.6597	100.2	250	0.01	349.7	0.013818	159.8
2.5	0.1	73.859	6.5182	92	250	0.1	356.88	0.022146	94
2.5	1	88.922	17.426	32.2	250	1	611.44	0.06657	32.4
2.5	10	126.08	68.767	0	250	10	1142.8	0.20832	0.2
2.5	50	126.08	68.767	0	250	50	1149	0.20925	0
2.5	100	126.08	68.767	0	250	100	1149	0.20925	0
5	0.0001	86.735	3.1616	113	500	0.0001	484.97	0.004986	177
5	0.001	86.735	3.1616	113	500	0.001	484.97	0.004986	177
5	0.01	86.735	3.1616	113	500	0.01	491.47	0.005361	169
5	0.1	86.725	3.0029	103.8	500	0.1	490.84	0.012077	80.4
5	1	100.35	6.603	46.4	500	1	791.1	0.029797	22.8
5	10	201.2	42.761	1	500	10	1256.7	0.062708	0
5	50	209.51	45.335	0	500	50	1256.7	0.062708	0

Bibliography

[1] Aertsen, A. and Preissl, H. (1991). Dynamics of activity and connectivity in physiological neuronal networks. In H. Schuster, editor, *Nonlinear Dynamics and Neuronal Networks*, pages 281-302. VCH publishers Inc, New York.

[2] Anderson, J. (2005). Learning in sparsely connected and sparsely coded system. In *Ersatz Brain Project working note.*

[3] Boatman-Reich, D., Franaszczuk, P., Korzeniewska, A., Caffo, B., Ritzl, E., Colwell, S., and Crone, N. (2010). Quantifying auditory event-related responses in multichannel human intracranial recordings. *Frontiers in Computational Neuroscience*, **4**(4).

[4] Bressler, S. and Ding, M. (2002). Event-related potentials. In *The handbook of brain theory and neural networks*, pages 412-415. John Wiley & Sons, Inc.

[5] Bullmore, E. and Sporns, O. (2009). Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature Reviews Neuroscience*, **10**(3), 186-198.

[6] Burton, M. (2001). The role of inferior frontal cortex in phonological processing. *Cognitive Science*, **25**, 695-709.

[7] Cervenka, M., Franaszczuk, P., Crone, N., Hong, B., Caffo, B., Bhatt, P., Lenz, F., and Boatman-Reich, D. (2013). Reliability of early cortical auditory gamma-band responses. *Clinical Neurophysiology*, **124**(1), 70-82.

[8] Daunizeau, J., David, O., and Stephan, K. E. (2009). Dynamic causal modeling: A critical review of the biophysical and statistical foundations. *NeuroImage*, **58**, 312-322.

[9] David, O. and Friston, K. (2003). A neural mass model for meg/eeg: coupling and neuronal dynamics. *NeuroImage*, **20**, 1743-1755.

[10] David, O., Kiebel, S., Harrison, L., Mattout, J., Kilner, J., and Friston, K. (2006). Dynamic causal modelling of evoked responses in eeg and meg. *NeuroImage*, **30**, 1255-1272.

[11] Duann, J., Jung, T., Kuo, W., Yeh, T., Makeig, S., Hsieh, J., and TJ, S. (2002). Single-trial variability in event-related bold signals. *Neuroimage*, **15**(4), 823-35.

[12] Durka, P., Ircha, D., Neuper, C., and Pfurtscheller, G. (2001). Time-frequency microstructure of event-related electro-encephalogram desynchronisation and synchronisation. *Medical & Biological Engineering & Computing*, **39**(3), 315-21.

[13] Edwards, E., Soltani, M., Deouell, L., Berger, M., and Knight, R. (2005). High gamma activity in response to deviant auditory stimuli recorded directly from human cortex. *Journal of Neurophysiology*, **94**, 4269-4280.

[14] Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least Angle Regression. *Annals of Statistics*, **32**(2), 407-499.

[15] Fan, J. and Lv, J. (2010). A selective overview of variable selection in high dimensional feature space. *Statistica Sinica*, **20**, 101-148.

[16] Flinker, A., Chang, E., Kirsch, H., Barbaro, N., Crone, N., and Knight, R. (2010). Single-trial speech suppression of auditory cortex activity in humans. *The Journal of Neuroscience*, **30**(49), 16643-16650.

[17] Foldiak, P. and Young, M. P. (1995). Sparse coding in the primate cortex. In 895-898, editor, *The Handbook of Brain Theory and Neural Networks*. The MIT Press.

[18] Franaszczuk, P. and Bergey, G. (1998). Application of the directed transfer function method to mesial and lateral onset temporal lobe seizures. *Brain Topogra*, **11**, 13-21.

[19] Friedman, J., Hastie, T., Hofling, H., and Tibshirani, R. (2007). Pathwise coordinate optimization. *The Annals of Applied Statistics*, **1**, 302-332.

[20] Friston, K. J., Frith, C. D., and Frackowiak, R. S. J. (1993a). Time-dependent changes in effective connectivity measured with PET. *Hum. Brain Mapp.*, **1**, 69-80.

[21] Friston, K. J., Frith, C. D., Liddle, P. F., and Frackowiak, R. S. J. (1993b). Functional connectivity: the principal component analysis of large (PET) data sets. *J. Cereb. Blood Flow Metab.*, **13**, 5-14.

[22] Friston, K. J., Mechelli, A., Turner, R., and Price, C.J. (2000). Nonlinear responses in fMRI: the Balloon model, Volterra kernels and other hemodynamics. *NeuroImage*, **12**, 466-477.

[23] Friston, K. J., Harrison, L., and Penny, W. D. (2003). Dynamic causal modeling. *NeuroImage*, **19**, 1273-1302.

[24] Friston, K., Frith, C., Dolan, R., Price, C., Zeki, S., Ashburner, J., and Penny, W. (2004). *Human Brain Function*, Section 4. Academic Press, 2 edition.

[25] Gerstein, G. L. and Perkel, D. H. (1969). Simultaneously recorded trains of action potentials: analysis and functional interpretation. *Science*, **164**, 828-830.

[26] Girvan, M. and Newman, M. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, **99**(12), 7821-7826.

[27] Graner, F. and Glazier, J. (1992). Simulation of biological cell sorting using a two-dimensional extended potts model. *Physical Review Letters*, **69**, 2013-2016.

[28] Holland, P. W. (1986). Statistics and causal inference (with discussion). *Journal of the American Statistical Association*, **81**, 945-970.

[29] Kiebel, S. J., David, O., and Friston, K. J. (2006). Dynamic causal modelling of evoked responses in EEG/MEG with lead-field parameterization. *NeuroImage*, **30**, 1273-1284.

[30] Korzeniewska, A., Franaszczuk, P., Crainiceanu, C., Kus, R., and NE, C. (2011). Dynamic of large-scale cortical interactions at high gamma frequencies during word production: Event related causality (erc) analysis of human electrocorticography (ecog). *NeuroImage*, **56**(4), 2218-2237.

[31] Mallat, S. and Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. *IEEE Transactions Signal Processing*, **41**, 3397-3415.

[32] McIntosh, A. R., and Gonzalez-Lima, F. (1994). Structural equation modeling and its application to network analysis in functional brain imaging. *Humman Brain Mapping*, **2**, 2-22.

[33] Micheloyannis, S. (2012). Graph-based network analysis in schizophrenia. *World Journal of Psychiatry*, **2**(1), 1-12.

[34] Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., and Alon, U. (2002). Network motifs: Simple building blocks of complex networks. *Science*, **298**(5594), 824-827.

[35] Milo, R., Itzkovitz, S., Kashtan, N., Levitt, R., Shen-Orr, S., Ayzenshtat, I., M, S., and Alon, U. (2004). Superfamilies of evolved and designed networks. *Science*, **303**(5663), 1538-1542.

[36] Newman, M. (2003). The structure and function of complex networks. *SIAM Rev*, **45**, 167-256.

[37] Newman, M. (2004). Fast algorithm for detecting community structure in networks. *Physical Review E.*, **69**, 066133.

[38] Newman, M. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, **103**(23), 8577-8696.

[39] Ogawa, S., Tank, D., Menon, R., Ellerman, J., Kim, S., Merkle, H. and Ugurbil, K. (1992). Intrinsic signal changes accompanying sensory simulation: Functional brain mapping and magnetic resonance imaging. *Proc. Nat. Acad. Sci.*, **89**, 5951-5955.

[40] Olshausen, B. and Field, D. (2004). Sparse coding of sensor inputs. *Current Opinions in Neurobiology*, **14**, 481-487.

[41] Ogunnaike, B. A., and Ray, W. H. (1994). Process dynamics, modeling, and control. New York, NY: Oxford University Press, Inc [Chapter 12].

[42] Potts, R. (1952). Some generalized order-disorder transformations. *Mathematical Proceedings*, **48**, 106-109.

[43] Poyton, A. A., Varziri, M. S., McAuley, K. B., McLellan, P. J. and Ramsay, J. O. (2006). Parameter estimation in continuous dynamic models using principal differential analysis. *Computational Chemical Engineering*, **30**, 698-708

[44] Ramsay, J. O., and Silverman, B. W. (2005). Functional data analysis. New York: Springer.

[45] Ramsay, J. O., Hooker, G., Campbell, D., and Cao, J. (2007). Parameter estimation for differential equations: a generalized smoothing approach (with discussion). *Journal of the Royal Statistical Society, Ser. B*, **69**, 741-796.

[46] Reiss, P. T., and Ogden, R. T. (2007). Functional principal component regression and functional partial least square. *Journal of the American Statistical Association*, **102**, 984-996.

[47] Reiss, P. T., and Ogden, R. T. (2009). Smoothing parameter selection for a class of semi-parametric linear models. *Journal of the Royal Statistical Society, Ser. B*, **71**, 505-523

[48] Rubin, D. B. (1974). Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology*, **66**(1), 688-701.

[49] Rubin, D. B. (1978). Bayesian inference for causal effects: The role of randomization. *The Annals of Statistics*, **6**(1), 34-58.

[50] Shao, J. (1998). Convergence rates of the generalized information criterion. *Nonparametric Statistics*, **9**, 217-225.

[51] Sinai, A., Crone, N., Wied, H., Franaszczuk, P., Miglioretti, D., and Boatman-Reich, D. (2009). Intracranial mapping of auditory perception: Event-related responses and electrocortical stimulation. *Clinical Neurophysiology*, **120**, 140-149.

[52] Steven, B., Martinez, M., , and Parsons, L. (2006). Music and language side by side in the brain: a pet study of the generation of melodies and sentences. *European Journal of Neuroscience*, **23**, 2791-2803.

[53] Stuss, D. and Levine, B. (2002). Adult clinical neuropsychology: lessons from studies of the frontal lobes. *Annual Review of Psychology*, **53**, 401-33.

[54] Tibshirani, R. (1996) Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Ser. B.*, **58**, 267-288.

[55] Tononi, G., Sporns, O., and Edelman, G. M. (1994). A measure for brain complexity: Relating functional segregation and integration in the nervous system. *Proc. Natl. Acad. Sci. USA*, **91** 5033-5037.

[56] Truccolo, W., Ding, M., Knuth, K., Nakamura, R., and SL, B. (2002). Trial-to-trial variability of cortical evoked responses: implications for the analysis of functional connectivity. *Clinical Neurophysiology*, **113**(2), 206-226.

[57] Turetsky, B., Raz, J., and Fein, G. (1988). Noise and signal power and their effects on evoked potential estimation. *Electroencephalogr Clin Neurophysiol*, **71**(4), 310-8.

[58] Varah, J. M. (1982). A spline least squares method numerical parameter estimation in differential equations. *SIAM Journal on Scientific Computing*, **3**, 28-46

[59] Wahba, G. (1990). Spline models for observational data. Philadelphia: SIAM.

[60] Wood, S. N. (2011) Fast stable restricted maximum likelihood and marginal likelihood estimation of semi-parametric generalized linear models. *Journal of the Royal Statistical Society, Series B*, **73**(1), 3-36.