NEW PROGRESS IN HOT-SPOTS DETECTION, PARTIAL-DIFFERENTIAL-EQUATION-BASED MODEL IDENTIFICATION AND STATISTICAL COMPUTATION

A Dissertation
Presented to
The Academic Faculty

By

Yujie Zhao

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
H. Milton Stewart School of Industrial and Systems Engineering
Department of Industrial Engineering

Georgia Institute of Technology

May 2021

NEW PROGRESS IN HOT-SPOTS DETECTION, PARTIAL-DIFFERENTIAL-EQUATION-BASED MODEL IDENTIFICATION AND STATISTICAL COMPUTATION

Thesis committee:

Dr. Yajun Mei, Advisor
The H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Xiaoming Huo, Advisor
The H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Jianjun Shi
The H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Haomin Zhou
School of Mathematics
Georgia Institute of Technology

Dr. Sarah E. Holte
Department of Global Health
University of Washington
and
Fred Hutchinson Cancer Research Center

Date approved: April 9, 2021

There are only the pursued, the pursuing, the busy and the tired.

F. Scott Fitzgerald, The Great Gatsby

Dedicated to my parents Jianping Shen, Jiankang Zhao, my grandparents Meiping Zhao, Xuegen Zhao, Liujin Shen and Qingzhen Wu, for their boundless love and support.

ACKNOWLEDGMENTS

The Ph.D. study in Georgia Institute of Technology is like a journal where I traverse the beautify of statistics, the pay and pain in research, as well as the magic of research.

During the four-year study in Georgia Tech, I would like to express my sincere gratitude to my advisor Prof. Yajun Mei. I can still remember the first time when I met Dr. Mei at the 10th ICSA international conference in Shanghai in December 2016. This is my first time attending the academic seminar, and I totally got lost during most of the presentations due to limited research experience. When Dr. Mei gave the presentation, he mentioned that "I know we have undergraduates in the audience, so I will try my best to give the presentation in the most straightforward way for them to understand and motivate them to do research in the future." This is the first presentation I mostly understand, and it raised my interest in pursuing a Ph.D. degree for more research opportunities. After I entered Georgia Tech, Dr. Mei gave me lots of invaluable advice both on the research project and the career path plan. I really appreciate that Dr. Mei kept encouraging me to explore new areas, where I practice lots of my self-learning skills.

My heartfelt gratitude also goes to my advisor, Prof. Xiaoming Huo, for his endless patience, continued motivation, and immense knowledge. I started to work with Prof. Huo at the beginning of the second-year Ph.D. study. I got to know Prof. Huo during the course *Computational Statistics*, where I enjoyed Prof. Huo's teaching style, with the gist of each method clearly conveyed and various of extensions discussed. Besides, Prof. Huo also gave me numerous suggestions in research. Every time I was stuck in the technical proof, I can always get inspired by Prof. Huo's mathematical intuition. I am extremely grateful that thanks to Prof. Huo's foresight, I started researching in the operations research, which is an promising academic research area given the big data era. The first year to pick up operations research is horrible to me, I benefited lots from it and it is fun.

I would also like to thank Prof. Jianjun Shi, Prof. Haomin Zhou, and Prof. Sarah

E. Holte for serving on my thesis committee. Special thanks go to Prof. Jianjun Shi for serving my proposal committee, and being my lecturer on course *Statistical Method in Manufactory Design*. I am grateful to Prof. Haomin Zhou for his generous help and insightful suggestions on my thesis. Many thanks go to Prof. Sarah E. Holte for co-authoring two papers with me and writing recommendation letters for my job seeking.

During the pursuit of my Ph.D. at Georgia Tech, I owe my thanks to many people. I want to extend my gratitude to Prof. Yan Li for introducing the Georgia Tech Ph.D. program to me. I also appreciate May Li for her IT support, Amanda Ford for her administrative support, and Jonathan Etress for repairing my leaking roofs in my office. I am also thankful to other friends, they include but are not limited to Shanshan Cao, Ruizhi Zhang, Mingqiu Zhu, Ziqi Yang, Haomin Jiang, Chuanpin Yu, Chen Feng and Di Wu.

I am very fortunate to work as an intern in Wells Fargo in the summer of 2020 and Roche in the summer of 2019. I owe a special debt of gratitude to Shuguang Sun for providing guidance in working in pharmaceutical companies, which helps me find the career I truly want to pursue in the near future. I would also like to express my gratitude to my friend in Roche, Bingying Xie for her company during lunchtime. My greatness also goes to Lian Shen for being my internship manager and Huan Yan for introducing the quantitative analytic program of Wells Fargo to me.

My sincere gratitude also goes to my boyfriend, Ji Chen, for his great faith in me for the past three years. Though we are mostly in a long-distance relationship, whenever I am stuck in difficulties, he is always there by my side with all his calmness and generosity, motivating me to overcome my fear and face the world with boldness. Though it takes three hours or thirteen hours to visit him on-site, the tiredness of travel fades away when I see the twinkle in his eyes.

Finally, I am greatly indebted to my beloved parents and grandparents for their continued support and encouragement. The weekly WeChat video conversations really comfort me for my homesickness.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This thesis discusses the new progress in (1) hot-spots detection in spatial-temporal data, (2) partial-differential-equation-based (PDE-based) model identification, and (3) optimization in the Least Absolute Shrinkage and Selection Operator (Lasso) type problem.

In this thesis, we have four main works. Chapter 1 and Chapter 2 fall in the first area, i.e., hot-spots detection in spatio-temporal data. Chapter 3 belongs to the second area, i.e., PDE-based model identification. Chapter 4 is for the third area, i.e., optimization in the Lasso-type problem. The detailed description of these four chapters is summarized as follows.

In Chapter 1, we aim at detecting hot-spots in multivariate spatio-temporal dataset that are non-stationary over time. To realize this objective, we propose a statistical method to under the framework of tensor decomposition and our method has three steps. First, we fit the observed data into a Smooth Sparse Decomposition Tensor (SSD-Tensor) model that serves as a dimension reduction and de-noising technique: it is an additive model that decomposes the original data into three components: smooth but non-stationary global mean, sparse local anomalies, and random noises. Next, we estimate the model parameters by the penalized framework that includes a combination of Lasso and fused Lasso penalty to address the spatial sparsity and temporal consistency, respectively. Finally, we apply a Cumulative Sum (CUSUM) Control Chart to monitor the model residuals, which allows us to detect when and where the hot-spot event occurs. To demonstrate the usefulness of our proposed SSD-Tensor method, we compare it with several other methods in extensive numerical simulation studies and a real crime rate dataset. The material of this chapter is published in *Journal of Applied Statistics* in January, 2021 under the title "Rapid Detection of Hot-spots via Tensor Decomposition with Applications to Crime Rate Data" with co-authors Hao Yan, Sarah E. Holte and Yajun Mei.

In Chapter 2, we improve the methodology in Chapter 1 both statistically and compu-

tationally. The statistical improvement is the new methodologies to detect hot-spots with temporal circularity, instead of temporal continuity as in Chapter 1. This helps us handle many bio-surveillance and healthcare applications, where data sources are measured from many spatial locations repeatedly over time, say, daily/weekly/monthly. The computational improvement is the development of a more efficient algorithm. The main tool we use to accelerate the calculation is the tensor decomposition, which is similar to the matrix context where it might be difficult to compute the inverse of a large matrix in general, but it will be straightforward to calculate the inverse of a large block diagonal matrix through the inverse of sub-matrices in the diagonal. The usefulness of the improved methodology is validated through numerical simulations and a real-world dataset in the weekly number of gonorrhea cases from 2006 to 2018 for 50 states in U.S.. The material of this chapter is accepted as a book chapter in *Frontiers in Statistical Quality Control 13* in February 2021 under the title "Rapid Detection of Hot-spot by Tensor Decomposition with Application to Weekly Gonorrhea Data" with co-authors Hao Yan, Sarah E. Holte, Roxanne P. Kerani and Yajun Mei.

In Chapter 3, we propose a two-stage method called *Spline Assisted Partial Differential Equation involved Model Identification (SAPDEMI)* method to efficiently identify the underlying PDE models from the noisy data. In the first stage – functional estimation stage – we employ the cubic spline to estimate the unobservable derivatives, which serve as candidates of the underlying PDE models. The contribution of this stage is that, it is computational efficient because it only requires the computational complexity of the linear polynomial of the sample size, which achieves the lowest possible order of complexity. In the second stage – model identification stage – we apply Lasso to identify the underlying PDE model. The contribution of this stage is that, we focus on the model selections, while the existing literature mostly focuses on parameter estimations. Moreover, we develop statistical properties of our method for correct identification, where the main tool we use is the primal-dual witness (PDW) method. Finally, we validate our theory through various

numerical examples.

In Chapter 4, we focus on developing an algorithm to solve the optimization with a ℓ_1 regularization term, namely the Lasso-type problem. The algorithm developed in this chapter can greatly reduce the computational complexity in Chapter 1, Chapter 2 and Chapter 3, where we try to realize sparse identification. The challenge to develop an efficient algorithm for the Lasso-type problem is that the objective function of the Lasso-type problem is not strictly convex when the number of samples is less than the number of features. This special property of the Lasso-problem leads the existing Lasso-type estimator, in general, cannot achieve the optimal rate due to the undesirable behavior of the absolute function at the origin. To overcome the above challenge, we develop a homotopic method, where we use a sequence of surrogate functions to approximate the ℓ_1 penalty that is used in the Lasso-type of estimators. The surrogate functions will converge to the ℓ_1 penalty in the Lasso estimator. At the same time, each surrogate function is strictly convex, which enables a provable faster numerical rate of convergence. In this chapter, we demonstrate that by meticulously defining the surrogate functions, one can prove a faster numerical convergence rate than any existing methods in computing for the Lasso-type of estimators. Namely, the state-of-the-art algorithms can only guarantee $O(1/\epsilon)$ or $O(1/\sqrt{\epsilon})$ convergence rates, while we can prove an $O([\log(1/\epsilon)]^2)$ for the newly proposed algorithm. Our numerical simulations show that the new algorithm also performs better empirically.

In Chapter 5, we summarize the contributions of the above four chapters. These four chapter fall in the three areas: (1) hot-spots detection in spatial-temporal data, (2) PDE-based model identification, and (3) optimization in the Lasso-type problem. In particular, the above three areas shares the similar technique, i.e., they are all involved with the sparse identification problem. The first area aims at sparse hot-spots detection. The second area focuses on identifying a few underlying derivatives among lots of candidates of derivatives. The third area targets on the computation with the ℓ_1 regularization term. In addition to the contributions, we also discuss the future research of these three areas.

CHAPTER 1
RAPID DETECTION OF HOT-SPOTS VIA TENSOR DECOMPOSITION WITH APPLICATIONS TO CRIME RATE DATA

1.1 Introduction

In many real-world applications such as biosurveillance, epidemiology, and sociology, multiple data sources are often measured from many spatial locations repeatedly over time, say, daily, monthly, or annually. This is commonly referred as *multivariate spatio-temporal* data. When such data are non-stationary over time, compared with detecting the global or system-wise changes as in the traditional statistical process control (SPC) or sequential change-point detection literature, we would be more interested in detecting *hot-spots* with spatially-sparsity and temporally-consistency. Here, we define hot-spots as the anomalies that can occur in the temporal and spatial domains among the multivariate spatio-temporal data.

The primary objective of this paper is to develop an efficient method for hot-spots detection and localization for multivariate spatio-temporal data. From the viewpoint of monitoring non-stationary multivariate spatial-temporal data, there are two kinds of changes: one is the change on the global-level trend (e.g., the *first-order* changes), and the other is the local-level hot-spot (e.g., *second-order* changes). Here we focus on detecting the latter one, and assume that local-level hot-spots have the following two properties: (1) spatial sparsity, i.e., the local changes are sparse in the spatial domain; and (2) temporal consistency, i.e., the local changes last for a reasonable period of time.

Little research has been done on the hot-spot detection for multivariate spatio-temporal, although there are two major related existing research areas for detection in multivariate spatio-temporal data: one is the spatio-temporal cluster detection, and the other is change-

point detection. In the first area, the famous representative is the *scan statistics based method*, which was first developed in the 1960s in [1] and later extended by [2] to detect anomalous clusters in spatio-temporal data. The main idea of scan statistics is to detect the abnormal clusters by utilizing maximal log-likelihood ratio (more mathematical details are provided in subsection 1.8.3). It is worth noting that the scan statistics-based method is a parametric method, in which the parametric families of data distributions are made. For instance, [3] assumes the negative binomial distribution, [2, 4, 5, 6] investigate the Poisson distribution. A limitation of scan statistics is that it assumes that the background is independent and identically distributed (i.i.d.) or follows a rather simple probability distribution, which might not be suitable to handle non-stationary spatio-temporal data.

The second category of existing research is the change-point detection problem for the spatio-temporal data. Below we will further review two approaches that are related to our context: the *Least Absolute Shrinkage and Selection (Lasso)* based methods and *dimension-reduction-based methods*. Note that Lasso has been demonstrated to be an effective method for variable selection to address sparsity issues for high-dimensional data in the past decades since its developments in [7], and thus it is natural to apply it to detect sparse changes in high-dimensional data, see [8, 9, 10, 11, 12, 13]. While the sparse change of Lasso is similar to the hot-spots, unfortunately, as our extensive simulation studies will demonstrate, the Lasso-based control chart is unable to separate the local hot-spots from the non-stationary global trend mean in the spatio-temporal data.

For the dimension-reduction-based change-point detection method, Principal Component Analysis (PCA) or other dimension reduction methods are often used to extract the features from the high-dimensional data. More specifically, [14] reduces the dimensionality in the spatio-temporal data by constructing T^2 and Q charts separately. [15] combines multivariate functional PCA with change-point models to detect the hot-spots. For other dimension-reduction-methods, please see [16, 17, 18, 19, 20, 21] for more details. The drawbacks of PCA or other dimension-reduction-based methods are the restriction of the

change-point detection problem and the failure to consider the spatial sparsity and temporal consistency of hot-spots.

Our proposed method is essential to the application of the Lasso-based method and dimension-reduction-based method for SPC or change-point detection over a model based on *tensor*, which is a multi-dimensional array. It is worth noting that the multivariate spatio-temporal data can often be represented in 3-dimensional tensor format as "Spatial dimension $\times$ Temporal dimension $\times$ Attributes dimension". Therefore, we propose to use tensor to represent the original data, and consider the additive model that decomposes this tensor into three components: (1) smooth but non-stationary global trend mean, (2) sparse local hot-spots, and (3) residuals. We term our proposed decomposition model as *Smooth Sparse Decomposition-Tensor (SSD-Tensor)*. Besides, when fitting the raw data to the SSD-Tensor model, we propose to add two penalty functions: the first one is the Lasso type penalty to guarantee the spatial sparsity of hot-spots, and the second one is the fused-Lasso penalty [see 22] to guarantee the temporal consistency of hot-spots. This allows us to not only detect when the hot-spot happens over the temporal domain (i.e., *hot-spot detection* problem) but also localize where and which types/attributes of the hot-spots occur if the change happens (i.e., *hot-spot localization* problem).

It is useful to highlight the novelty of our proposed method as compared to the existing research on spatio-temporal data. First, our proposed SSD-Tensor method can detect hot-spots when the global trend of the spatio-temporal data is dynamic (i.e., non-stationary or non-i.i.d). That is, our method is robust to the global trend, in the sense that it can detect hot-spots with positive or negative mean shifts on top of the global trend of raw data, no matter whether it is decreasing or increasing. In comparison, the existing SPC or change-point detection methods often assume that the background is i.i.d. and focus on detecting the anomalies under the static and i.i.d. background. Second, we should clarify that the primary goal of our proposed method is not the prediction or model fitting. Instead, we focus on hot-spots detection and localization among the dynamic spatio-temporal data.

Of course, good fitting or estimation of the global trend will be useful to detect hot-spots accurately. Finally, while our paper focuses only on a 3-dimensional tensor arising from our motivating application in crime rates, our proposed hot-spot method can easily be extended to any d-dimensional tensor ($d \geq 3$), as we can simply add corresponding dimensions and bases in the tensor analysis. The capability of extending to high-dimensional tensor data is one of the main advantages of our proposed SSD-Tensor method.

The remainder of this paper is described as follows. In section 1.2, we introduce and visualize the crime rate dataset, which will be used as our motivating example. In section 1.3, we present our proposed SSD-Tensor model and discuss how to estimate model parameters from data. In section 1.4, we describe how to use our proposed SSD-Tensor model to detect and localize hot-spots. In section 1.5, we compare our proposed method with several benchmark methods and demonstrate its usefulness through extensive simulations. In section 1.6, we represent the application of our proposed method in a real crime rate dataset.

1.2 Motivating Example & Background

This section gives a detailed description of the crime rate dataset that is available from the U.S. Department of Justice Federal Bureau of Investigation (see https://www.ucrdatatool.gov/Search/Crime/State/StateCrime.cfm). The crime rates are recorded from 1965 to 2014 for 51 states in the United States annually. In each year and for each state, three types of crime crates are reported: (1) *murder and non-negligent manslaughter*; (2) *legacy rape*; and (3) *revised rape*. Table 1.1 shows the head of the dataset, and the value in the table is the crime rate per 100,000 population in each state.

It is worth noting that the crime rate dataset has three dimensions: (1) the temporal dimension (i.e., years), (2) the spatial dimension (i.e., states), and (3) the attribute/category dimension (i.e., three different types of crime rates). For a visual representation, we plot several figures that show the characteristics of each dimension.

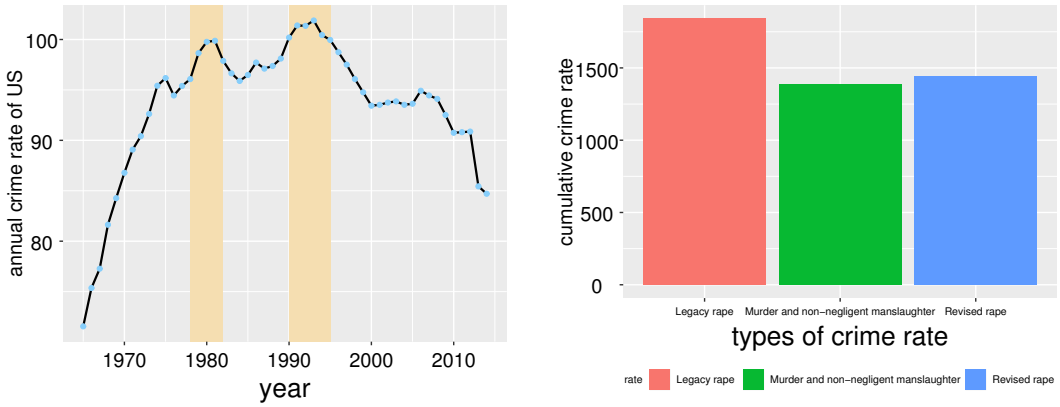
Table 1.1: Head of the crime rate dataset from 1965 to 2014 for 51 states in the U.S. annually. The dataset is publicly available from https://www.ucrdatatool.gov/Search/Crime/State/StateCrime.cfm. The recorded three types of crime rates are murder and non-negligent manslaughter, legacy rape, and revised rape. The value in the table is the crime rate per 100,000 population of each state, and the states are ordered in alphabetical order.

Year	State	Murder and non-negligent manslaughter	Legacy rape	Revised rape
1965	Alabama	11.4	10.6	28.7
1965	Alaska	6.3	17.8	39.9
1965	Arkansas	5.9	10.4	23.7
⋮	⋮	⋮	⋮	⋮
1965	Wyoming	2.9	11.5	17.9
1966	Alabama	10.9	9.7	32
1966	Alaska	12.9	19.5	36
⋮	⋮	⋮	⋮	⋮

To begin with, we first illustrate the temporal dimension (i.e., years), where we plot the time series of the sum of all states' crime rates in the logarithm scale in Figure 1.1(a). The x-axis is the year ranging from 1965 to 2014, and the y-axis is the sum of all states' crime rates in the logarithm scale. We acknowledge that this summation is different from the actual annual crime rate in the United States, which needs to take into account the different population sizes of each state at different years. Here, we use this notation to refer it as the annual crime rate of the United States for simplicity, since our purpose here is only for demonstration of the overall temporal trends. Figure 1.1(a) suggests that the crime rates are increasing in the first 10 years (1965-1975), then become stationary during 1975-1995, and finally have a decreasing trend during 1995-2014. Furthermore, it is interesting to point out the two peaks around 1980 and 1992, since we are interested in finding out whether they are caused by global trends or local hot-spots.

Next, we show the characteristics of the crime rate dataset on the attribute/category dimension (i.e., three types of the crime rates) in Figure 1.1(b), where different bars represents the different types of the crime rates, and the height of the bar represents the cumulative crime rate from 1965 to 2014 in the United States. It can be seen that these three crime rates overall happen with similar frequencies. This can possibly make it challenging

to detect the hot-spots if we analyze the three-dimension data as a whole.

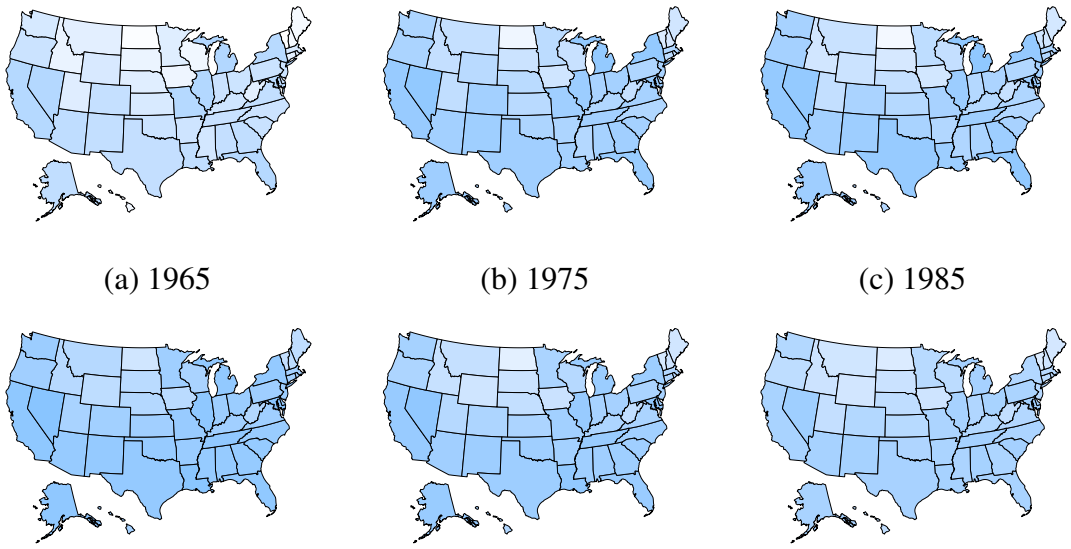


(a) Time Series plot (b) Bar plot

Figure 1.1: (a) Time series of annual crime rates in the United States from1965 to 2014 & (b) Bar plot of three cumulative rates from 1965 to 2014. For (a), the x-axis plot is the year ranging from 1965 to 2014, and the y-axis is the annual crime rate of the U.S. in the logarithm scale. Because the value of the crime rate is the number of crime cases per 100,000 population, it is reasonable for the annual crime rate to be larger than 100 during some years. For plot (b), different bars represent different types of crime rates, and the height of the bar represents the cumulative crime rate from 1965 to 2014 in the U.S. in the logarithm scale.

Finally, we illustrate the crime rate data in the spatial dimension (i.e., states) in Figure 1.2. In Figure 1.2, each map shows the spatial information of the crime rates in six different years. The selected six years are starting from 1965 with a ten-year interval, and the only exception is the sixth map, which uses the year 2014 data, as the data in the year 2015 is not available yet as of August 2020. If a state has a very dark color in Figure 1.2, it has very high crime rates. We can see from the spatial plot in Figure 1.2 that the spatial patterns of crime rates are generally very smooth.

From Figure 1.1 and Figure 1.2, there seems to be a brief increasing trend during 1984-1995, but it is difficult to conclude whether this is due to the global trend or local hot-spots without refined analysis. Note that the global trend might be caused by the U.S. federal governments' policies or the world-wise economic or political situations that are out of control of any local state or certain government branches. However, it is possible that the issues from local hot-spots can be addressed by borrowing other states' successful strategies



(a) 1965 (b) 1975 (c) 1985

(d) 1995 (e) 2005 (f) 2014

Figure 1.2: Each map shows the spatial information of the crime rates in six different years. Darker color represents a higher crime rate. We can see that the spatial patterns of crime rate are generally very smooth.

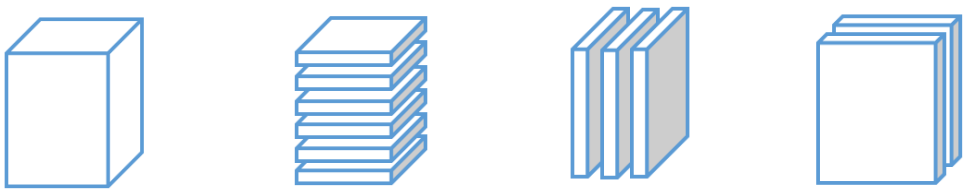
or policies.

Below we provide the technical background on tensor through the crime rate dataset. Note that we can store our data set as a tensor of order three, denoted by $\mathcal{Y} = (\mathcal{Y}_{i,j,t})$, where an element $\mathcal{Y}_{i,j,t}$ represents the j-th crime rate of state i in year t with $i = 1, \ldots, 51$ for 51 states, $j = 1, 2, 3$ for three different type of crime rate and $t = 1, \ldots, 50$ for 50 years from 1965 to 2014.

We are now ready to introduce some basic tensor notation and algebra that are useful in this paper. For the notations throughout the paper, scalars are denoted by lowercase letters (e.g., θ), vectors are denoted by lowercase boldface letters ($\boldsymbol{\theta}$), matrices are denoted by uppercase boldface letter ($\boldsymbol{\Theta}$), and tensors by curlicue letter (ϑ). For example, a tensor of order N is represented by $\vartheta \in \mathbb{R}^{I_1 \times \ldots \times I_N}$, where I_n represent the mode-n dimension of ϑ for $n = 1, \ldots, N$.

Next, we introduce the notation of *slice*, which is a two-dimensional section of a tensor by fixing all but two indices. Let us take tensor $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ as an exam-

ple. Figure 1.3 visualizes its horizontal, lateral, and frontal slides, which are denoted by $\mathcal{Y}_{i::}(\forall i = 1, \ldots, n_1), \mathcal{Y}_{:j:}(\forall j = 1, \ldots, n_2)$, and $\mathcal{Y}_{::t}(\forall t = 1, \ldots, n_3)$, respectively.



(a) the original tensor: $\mathcal{Y}^{n_1 \times n_2 \times n_3}$ (b) horizontal slices: $\mathcal{Y}_{i::}\ \forall i = 1, \ldots, n_1$ (c) lateral slices: $\mathcal{Y}_{:j:}\ \forall j = 1, \ldots, n_2$ (d) frontal slices: $\mathcal{Y}_{::t}\ \forall t = 1, \ldots, n_3$

Figure 1.3: Slices of 3-dimension tenor

Moreover, we introduce the *mode-n product* between a tensor and a matrix. For a given tensor of order N, i.e., $\vartheta \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ and a given matrix $\mathbf{B} \in \mathbb{R}^{J_n \times I_n}$, the mode-$n$ product between ϑ and $\mathbf{B}$, denoted by $\vartheta \times_n \mathbf{B}$, is a new tensor of dimension $\mathbb{R}^{I_1 \times \ldots I_{n-1} \times J_n \times I_{n+1} \times \ldots I_N}$, where its $(i_1, \ldots, i_{n-1}, j_n, i_{n+1}, \ldots, i_N)$-th entry can be computed as $\sum_{i_n} \vartheta_{i_1,\ldots,i_N} \mathbf{B}_{j_n,i_n}$. Here we use the notation $\mathbf{B}_{j_n,i_n}$ to refer the (j_n, i_n)-th entry in matrix $\mathbf{B}$, and $\vartheta_{i_1,\ldots,i_N}$ to refer the $(i_1, \ldots, i_N)$-th entry tensor ϑ.

Finally, we discuss the *Tucker decomposition*, which is a useful technique in tensor algebra. Its main idea is to decompose a tensor $\mathcal{Y} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ into a core tensor multiplied by matrices along each dimension:

$$\mathcal{Y} = \vartheta \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \ldots \times_N \mathbf{B}^{(N)},$$

where $\mathbf{B}^{(n)} \in \mathbb{R}^{I_n \times I_n}$ is an orthogonal matrix for $n = 1, \ldots, N$. The above equation can be equivalently represented by a *Kronecker product*, i.e.,

$$\text{vec}(\mathcal{Y}) = (\mathbf{B}^{(N)} \otimes \mathbf{B}^{(N-1)} \ldots \otimes \mathbf{B}^{(1)})\text{vec}(\boldsymbol{\theta}),$$

where $\text{vec}(\cdot)$ is the vectorized operator. Here the Kronecker product $\otimes$ is defined as follow: suppose $\mathbf{B}_1 \in \mathbb{R}^{m \times n}$ and $\mathbf{B}_2 \in \mathbb{R}^{p \times q}$ are matrices, the Kronecker product of these matrices

is an $mp \times nq$ block matrix defined by

$$\mathbf{B}_1 \otimes \mathbf{B}_2 = \begin{bmatrix} b_{11}\mathbf{B}_2 & \dots & b_{1n}\mathbf{B}_2 \\ \vdots & \ddots & \vdots \\ b_{m1}\mathbf{B}_2 & \dots & b_{mn}\mathbf{B}_2 \end{bmatrix}.$$

The Kronecker product has been shown to have excellent computational efficiency for tensor data [see 23].

1.3 Our Proposed SSD-Tensor Model

This section presents our proposed SSD-Tensor model and its parameter estimation, whereas the discussion of hot-spot detection and localization will be postponed to the next section. The main advantage of using tensor is not only to characterize the complicated "within-dimension" or "between-dimension" correlations but also to simplify the computations. The latter is similar to the matrix context where it might be difficult to compute the inverse of a large matrix in general, but it will be straightforward to calculate the inverse of a large block diagonal matrix through the inverse of sub-matrices in the diagonal.

To better present our main ideas, we split this section into three subsections. In subsection 1.3.1, we present the mathematical formulation of our proposed SSD-Tensor model, and subsection 1.3.2 develops the optimization algorithm for the parameter estimation problem of the model when fitting the observed data. Since the choice of basis in the tensor decomposition plays an important role in representing spatial or temporal patterns, we devote subsection 1.3.3 to discuss the choice of basis in our context.

1.3.1 Our Proposed Model

In this section, we present the mathematical formulation of our proposed SSD-Tensor model. Here let us focus on our motivating data with three-dimension tensor $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, where the (i, j, t)-th entry indicate the j-th crime rate in i-th state in year t, with $i =$

$1, \ldots, n_1 = 51$, $j = 1, \ldots, n_2 = 3$, and $t = 1, \ldots, n_3 = 50$.

At the high level, our proposed SSD-Tensor model is to decompose a raw data $\mathcal{Y}_{i,j,t}$ into three components: the smooth global trend mean $\mathcal{U}_{i,j,t}$, local hot-spots $\mathcal{H}_{i,j,t}$, and residuals $\mathcal{E}_{i,j,t}$. Mathematically, it is an additive model with the form

$$\mathcal{Y}_{i,j,t} = \mathcal{U}_{i,j,t} + \mathcal{H}_{i,j,t} + \mathcal{E}_{i,j,t},$$

where the residuals $\mathcal{E}_{i,j,t}$ are i.i.d. with $N(0, \sigma^2)$. Under the tensor notation, we denote $\mathcal{U}, \mathcal{H}, \mathcal{E}$ as the corresponding tensors of dimension $\mathbb{R}^{n_1 \times n_2 \times n_3}$. Then our proposed model can be rewritten as $\mathcal{Y} = \mathcal{U} + \mathcal{H} + \mathcal{E}$.

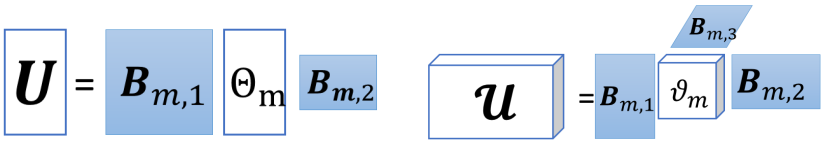
It remains to discuss two main components of our model in more detail. For the global trend mean $\mathcal{U}$, our main idea is to adopt the basis decomposition framework that allows us to address the complicated within-dimension correlation and between-dimension correlations. To be more specific, we propose to decompose the global trend mean tensor $\mathcal{U}$ as

$$\mathcal{U} = \vartheta_m \times_1 \mathbf{B}_{m,1} \times_2 \mathbf{B}_{m,2} \times_3 \mathbf{B}_{m,3},$$

where $\vartheta_m \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is an unknown tensor parameter to be estimated, and matrices $\mathbf{B}_{m,1} \in \mathbb{R}^{n_1 \times n_1}, \mathbf{B}_{m,2} \in \mathbb{R}^{n_2 \times n_2}, \mathbf{B}_{m,3} \in \mathbb{R}^{n_3 \times n_3}$ are pre-specified bases to describe the within-state correlation, within-rate correlation, and within-year correlation in $\mathcal{U}$, respectively. The choices of the base matrices, i,e., $\mathbf{B}_{m,1}, \mathbf{B}_{m,2}$ and $\mathbf{B}_{m,3}$, are very important in practice, and we will discuss them in more details in subsection 1.3.3. In our tensor decomposition, the operator $\times_1, \times_2, \times_3$ is the mode-n product reviewed in the previous section, where $n = 1, 2, 3$. This mode-n product is used to model the between-dimension correlations in $\mathcal{U}$.

Since some readers might not be familiar with the basis decomposition in tensor, let us provide a little more background. Loosely speaking, the basis decomposition for tensor is an extension of the matrix decomposition and is similar to the singular value decomposition

(SVD) in the sense of representing a matrix or tensor as the product of several specialized matrices or tensors. The main difference is that the bases $\mathbf{B}_{m_1}, \mathbf{B}_{m,2}, \mathbf{B}_{m,3}$ are known in the basis decomposition. Figure 1.4 illustrates the relationship between SVD and basis decomposition.



(a) SVD (b) basis decomposition

Figure 1.4: The relationship between SVD and the basis decomposition. The plot (a) is the SVD of matrix $\mathbf{U} \in \mathbb{R}^{n_1 \times n_2}$. The plot (b) is the basis decomposition of tensor $\mathcal{U} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$.

Next, for the local hot-spot tensor $\mathcal{H}$, we follow the similar basis decomposition way as $\mathcal{U}$:

$$\mathcal{H} = \vartheta_h \times_1 \mathbf{B}_{h,1} \times_2 \mathbf{B}_{h,2} \times_3 \mathbf{B}_{h,3},$$

where $\vartheta_h \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is the unknown tensor to be estimated. And matrices $\mathbf{B}_{h,1} \in \mathbb{R}^{n_1 \times n_1}, \mathbf{B}_{h,2} \in \mathbb{R}^{n_2 \times n_2}, \mathbf{B}_{h,3} \in \mathbb{R}^{n_3 \times n_3}$ are pre-specified bases to describe the within-state correlation, within-rate correlation, and within-year correlation in $\mathcal{H}$, respectively. For the selection of these bases, i,e., $\mathbf{B}_{h,1}, \mathbf{B}_{h,2}, \mathbf{B}_{h,3}$, a detailed discussion can be found in subsection 1.3.3. The operator $\times_1, \times_2, \times_3$ is the mode-n $(n = 1, 2, 3)$ product (see the definition of mode-n product in the end of section 1.2) to model the between-dimension correlations in $\mathcal{H}$.

In summary, our proposed SSD-Tensor model can be written in the following tensor format:

$$\mathcal{Y} = \vartheta_m \times_1 \mathbf{B}_{m,1} \times_2 \mathbf{B}_{m,2} \times_3 \mathbf{B}_{m,3} + \vartheta_h \times_1 \mathbf{B}_{h,1} \times_2 \mathbf{B}_{h,2} \times_3 \mathbf{B}_{h,3} + \mathcal{E}. \tag{1.1}$$

This tensor representation above allows us to develop computationally efficient methods for

estimation and prediction. The detailed reason why tensor is more computational efficient can be found in subsection 1.3.2 and subsection 1.8.2. By introducing tensor algebra, the above format of our model can be written equivalently:

$$\mathbf{y} = \underbrace{(\mathbf{B}_{m,1} \otimes \mathbf{B}_{m,2} \otimes \mathbf{B}_{m,3})}_{\mathbf{B}_m} \boldsymbol{\theta}_m + \underbrace{(\mathbf{B}_{h,1} \otimes \mathbf{B}_{h,2} \otimes \mathbf{B}_{h,3})}_{\mathbf{B}_h} \boldsymbol{\theta}_h + \mathbf{e}, \tag{1.2}$$

where vector $\mathbf{y} = \text{vec}(\mathcal{Y})$, vector $\boldsymbol{\theta}_m = \text{vec}(\vartheta_m)$, vector $\boldsymbol{\theta}_h = \text{vec}(\vartheta_h)$, and vector $\mathbf{e} = \text{vec}(\mathcal{E})$. The residual vector $\mathbf{e}$ is assumed to be the Gaussian white noise, i.e., $\mathbf{e} \sim N(0, \sigma^2 \mathbf{I})$.

In our proposed SSD-Tensor model in Equation 1.2, it is crucial to estimate the global mean parameter $\boldsymbol{\theta}_m$ and the local hot-spots parameter $\boldsymbol{\theta}_h$ when fitting to the observed data. Here we propose to estimate them by the penalized likelihood-function framework. To be more concrete, we propose to add two penalties in our parameter estimation. The first one is the Lasso penalty term on $\boldsymbol{\theta}_h$ to ensure the sparsity property of hot-spots: $R_1(\boldsymbol{\theta}_h) = \lambda_1 \|\boldsymbol{\theta}_h\|_1$. The second penalty is the fused Lasso penalty [see 22] on $\boldsymbol{\theta}_h$ to encourage the temporal consistency of the hot-spots: $R_2(\boldsymbol{\theta}_h) = \lambda_2 \sum_{t=2}^{n_3} \|\boldsymbol{\theta}_{h,t} - \boldsymbol{\theta}_{h,t-1}\|_1$, where $\boldsymbol{\theta}_{h,t} = \text{vec}(\vartheta_{h,::t})$.

Thus, by combining these two penalties, we propose to estimate the parameters $(\boldsymbol{\theta}_m, \boldsymbol{\theta}_h, \boldsymbol{e})$ via the following optimization problem:

$$\begin{aligned} \arg \quad & \min_{\boldsymbol{\theta}_m, \boldsymbol{\theta}_h, \boldsymbol{e}} \|\boldsymbol{e}\|_2^2 + \lambda_1 \|\boldsymbol{\theta}_h\|_1 + \lambda_2 \textstyle\sum_{t=2}^{n_3} \|\boldsymbol{\theta}_{h,t} - \boldsymbol{\theta}_{h,t-1}\|_1 \\ s.t. \quad & \boldsymbol{y} = (\mathbf{B}_{m,1} \otimes \mathbf{B}_{m,2} \otimes \mathbf{B}_{m,3})\boldsymbol{\theta}_m + (\mathbf{B}_{h,1} \otimes \mathbf{B}_{h,2} \otimes \mathbf{B}_{h,3})\boldsymbol{\theta}_h + \mathbf{e}. \end{aligned} \tag{1.3}$$

We will discuss how to efficiently solve this optimization problem in the next section.

1.3.2 Optimization Algorithm for Estimation

In this section, we develop an efficient computational algorithm to solve the optimization problem in Equation 1.3. To emphasize the tuning parameters λ_1 and λ_2 in the penalty terms in Equation 1.3, we rewrite $\boldsymbol{\theta}_m$ and $\boldsymbol{\theta}_h$ as $\boldsymbol{\theta}_{m,\lambda_1,\lambda_2}$ and $\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}$, respectively.

Our proposed optimization algorithm to solve Equation 1.3 contains two main steps. The first one is to estimate $\boldsymbol{\theta}_{m,\lambda_1,\lambda_2}$ and $\boldsymbol{e}$ for a given $\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}$. The second step is to estimate $\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}$ by using the fast iterative shrinkage thresholding algorithm (FISTA) in [24] that iteratively updates the estimators. When implementing the FISTA algorithm to our context, at each iteration, we face an optimization problem that involves both Lasso and fused Lasso penalty parameters λ_1 and λ_2. To make the computation feasible, we apply an useful proposition that establishes the relationship of the optimal solutions between $\lambda_1 = 0$ and general $\lambda_1 \neq 0$.

Let us first discuss the estimation of $\boldsymbol{\theta}_{m,\lambda_1,\lambda_2}$ and $\boldsymbol{e}$ for a given $\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}$. It turns out that we have a closed-form solution, as shown in the following proposition.

Proposition 1.3.1. In the optimization problem in Equation 1.3, for a given $\widehat{\boldsymbol{\theta}}_{h,\lambda_1,\lambda_2}$, the optimal solution of $\widehat{\boldsymbol{\theta}}_{m,\lambda_1,\lambda_2}$ is given by:

$$\widehat{\boldsymbol{\theta}}_{m,\lambda_1,\lambda_2} = (\mathbf{B}_m^\top \mathbf{B}_m)^{-1} \left(\mathbf{B}_m^\top \mathbf{y} - \mathbf{B}_m^\top \mathbf{B}_h \widehat{\boldsymbol{\theta}}_{h,\lambda_1,\lambda_2} \right). \tag{1.4}$$

The proof of Proposition 1.3.1 follows the standard argument in the method of least squares in linear regression and thus omitted. Moreover, since $\mathbf{B}_m \in \mathbb{R}^{n_1 n_2 n_3 \times n_1 n_2 n_3}$ can have huge dimension when n_1, n_2, n_3 is large, it might be computationally expensive to solve the inverse of matrix $\mathbf{B}_m^\top \mathbf{B}_m$. By using the tensor algebra, we can greatly simplify the computations, see subsection 1.8.2 for the details.

Next, we discuss the estimation of $\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}$, which is highly non-trivial. By Proposition 1.3.1, the original optimization problem in Equation 1.3 becomes

$$\arg\min_{\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}} \|\mathbf{y}^* - \mathbf{X}\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}\|_2^2 + \lambda_1 \|\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}\|_1 + \lambda_2 \|\mathbf{D}\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}\|_1, \tag{1.5}$$

where $\mathbf{y}^* = [\mathbf{I} - \mathbf{H}_m]\,\mathbf{y}, \mathbf{X} = [\mathbf{I} - \mathbf{H}_m]\,\mathbf{B}_h$ with $\mathbf{H}_m = \mathbf{B}_m(\mathbf{B}_m^\top \mathbf{B}_m)^{-1}\mathbf{B}_m^\top$. Here the matrix $\mathbf{D}$ is defined to make $\lambda_2 \|\boldsymbol{D}\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}\|_1$ equivalent to $\lambda_2 \sum_{t=2}^{n_3} \|\boldsymbol{\theta}_{h,\lambda_1,\lambda_2,t} - \boldsymbol{\theta}_{h,\lambda_1,\lambda_2,t-1}\|_1$ (see subsection 1.8.1 for the explicit definition of $\mathbf{D}$).

It suffices to solve the new optimization problem in Equation 1.5. Note that Equation 1.5 is a generalized Lasso problem, and there are many optimization methods available in the literature. For instance, [25] solves a generalized Lasso problem through transformation to a common Lasso problem, but unfortunately, it is computationally heavy. [26] uses the alternating direction methods of multipliers (ADMM) algorithm to solve a generalized Lasso problem, but its convergence rate is of $O(1/k)$ as shown in [27], where k indicating the iterations. Another popular method is iterative shrinkage thresholding algorithms (ISTA) proposed by [28], which also has a convergence rate of $O(1/k)$. Later, researchers in [24] proposed a faster version of the ISTA, called FISTA, and shown that it has a convergence rate $O(1/k^2)$. Thus in our paper, we decide to choose the FISTA algorithm in [24] as the primary tool to solve Equation 1.5 due to its fast convergence rate.

There is a technical challenge to apply the FISTA algorithm to Equation 1.5. In the FISTA algorithm, each iteration is based on the proximal mapping of the loss function $F(\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}) = \|\mathbf{y}^* - \mathbf{X}\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}\|_2^2$. To be more concrete, the updating rule from i-th FISTA iteration to $(i+1)$-th FISTA iteration is given by

$$\begin{aligned}\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}^{(i+1)} &= \arg\min_{\boldsymbol{\theta}} F(\boldsymbol{\eta}^{(i)}) + \tfrac{\partial F(\boldsymbol{\eta}^{(i)})}{\partial \boldsymbol{\theta}_{h,\lambda_1,\lambda_2}}\left(\boldsymbol{\theta} - \boldsymbol{\eta}^{(i)}\right) + \tfrac{L_\theta}{2}\|\boldsymbol{\theta} - \boldsymbol{\eta}^{(i)}\|_2^2 + \lambda_1\|\boldsymbol{\theta}\|_1 + \lambda_2\|\mathbf{D}\boldsymbol{\theta}\|_1 \\ &\triangleq \pi_{\lambda_2}^{\lambda_1}(\mathbf{v}^{(i)})\end{aligned}$$

where $\boldsymbol{\eta}^{(i)}$ is the auxiliary variable, i.e, $\boldsymbol{\eta}^{(i)} = \boldsymbol{\theta}_{h,\lambda_1,\lambda_2}^{(i)} + \frac{t_i - 1}{t_{i+1}}(\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}^{(i)} - \boldsymbol{\theta}_{h,\lambda_1,\lambda_2}^{(i-1)})$ with $t_0 = 1$, $t_{i+1} = \frac{1+\sqrt{1+4t_i^2}}{2}$, $\mathbf{v}^{(i)} = \boldsymbol{\eta}^{(i)} - \frac{\partial}{L_\theta \partial \boldsymbol{\theta}} F(\boldsymbol{\eta}^{(i)})$ and L_θ is the stepsize which is fixed as the maximal eigenvalue of matrix $\mathbf{X}^\top\mathbf{X}$ (see subsection 1.8.4 for more details).

The challenge of applying the FISTA algorithm is because it is difficult to solve $\pi_{\lambda_2}^{\lambda_1}(\mathbf{v}^{(i)})$ directly, as it involves two penalties with parameters λ_1 and λ_2. To overcome this challenge, we propose to combine a nice theoretical result in [29] with an augmented ADMM algorithm in [30]. Specifically, [29] shows that there is a closed-form relationship between $\pi_{\lambda_2}^{\lambda_1}(\mathbf{v}^{(i)})$ and $\pi_{\lambda_2}^{0}(\mathbf{v}^{(i)})$. Thus we can easily compute $\pi_{\lambda_2}^{\lambda_1}(\mathbf{v}^{(i)})$ if we know how to solve $\pi_{\lambda_2}^{0}(\mathbf{v}^{(i)})$. The latter can be solved by the augmented ADMM algorithm in [30], which is

an extension of the regular ADMM method.

Proposition 1.3.2 summarizes that, in the i-th FISTA iteration, how to derive $\pi_{\lambda_2}^{\lambda_1}(\mathbf{v}^{(i)})$ from $\pi_{\lambda_2}^{0}(\mathbf{v}^{(i)})$, whose iterative applications lead to the estimation of $\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}$ in Equation 1.5:

Proposition 1.3.2. Assume that

1. there is a diagonal matrix $\mathbf{Q} \in \mathbb{R}^{n_1 n_2 n_3 \times n_1 n_2 n_3}$ satisfying $\mathbf{Q} \succeq \mathbf{D}^\top \mathbf{D}$, i.e., $\mathbf{Q} - \mathbf{D}^\top \mathbf{D}$ is a positive semidefinite matrix;

2. there is a scaler $\rho > 0$, which is a positive penalty parameter;

The updating procedure of augmented ADMM algorithm from k-th augmented ADMM iteration to the $(k+1)$-th augmented ADMM iteration is

$$\begin{aligned}
\pi_{\lambda_2}^{0}(\mathbf{v}^{(i)})^{(k+1)} &= \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^{n_1 n_2 n_3}} [F(\boldsymbol{\theta}) + (2\boldsymbol{\alpha}^{(k)} - \boldsymbol{\alpha}^{(k-1)})^\top \mathbf{D}\boldsymbol{\theta} + \\
&\quad \tfrac{\rho}{2}(\boldsymbol{\theta} - \pi_{\lambda_2}^{0}(\mathbf{v}^{(i)})^{(k+1)})^\top \mathbf{Q}(\boldsymbol{\theta} - \pi_{\lambda_2}^{0}(\mathbf{v}^{(i)})^{(k+1)})]; \\
\boldsymbol{\gamma}^{(k+1)} &= \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^{n_3-1}} \left[\mathbf{D}\boldsymbol{\gamma} + \tfrac{\rho}{2}\left\|\mathbf{D}\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\gamma} + \tfrac{1}{\rho}\boldsymbol{\alpha}^{(k)}\right\|_2^2\right]; \\
\boldsymbol{\alpha}^{(k+1)} &= \boldsymbol{\alpha}^{(k)} + \rho(\mathbf{D}\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\gamma}^{(k+1)}).
\end{aligned} \tag{1.6}$$

Suppose the above updating procedure lasts for M_2 iterations with M_2 as the number of augmented ADMM iterations, then we have can well approximate $\widehat{\pi_{\lambda_2}^{0}}(\mathbf{v}^{(i)})$ as $\pi_{\lambda_2}^{0}(\mathbf{v}^{(i)})^{(M_2)}$. Then $\widehat{\pi_{\lambda_2}^{\lambda_1}}(\mathbf{v}^{(i)})$ can be solved as

$$\widehat{\pi_{\lambda_2}^{\lambda_1}}(\mathbf{v}^{(i)}) = \text{sign}(\widehat{\pi_{\lambda_2}^{0}}(\mathbf{v}^{(i)})) \odot \max\left\{|\widehat{\pi_{\lambda_2}^{0}}(\mathbf{v}^{(i)})| - \lambda_1, 0\right\}, \tag{1.7}$$

where $\odot$ is an operator, which multiply two vectors in an element-wise fashion. For example, for two vectors $\mathbf{a}, \mathbf{b}$, the i-th operator of $\mathbf{a} \otimes \mathbf{b}$ is $\mathbf{a}_i \mathbf{b}_i$.

The proof of Proposition 1.3.2 is omitted, as the update rule in Equation 1.6 is an application of the augmented ADMM algorithm of [30], and Equation 1.7 follows directly from Theorem 1 of [29].

Combining the above two propositions, our optimization algorithm to solve the optimization problem in Equation 1.3 can be summarized as the pseudocode in algorithm 1.

Algorithm 1: Algorithm summary for estimation

Input: $\mathbf{y}, \mathbf{B}_m, \mathbf{B}_h, \lambda_1, \lambda_2, M_1, M_2, L_\theta, \rho, \mathbf{Q}$

Output: $\widehat{\boldsymbol{\theta}}_{h,\lambda_1,\lambda_2}, \widehat{\boldsymbol{\theta}}_{m,\lambda_1,\lambda_2}$

initialization: $\boldsymbol{\theta}^{(0)}_{h,\lambda_1,\lambda_2}, t_0 = 1$

for $i = 0, \ldots, M_1$ **do**

$\quad \boldsymbol{\eta}^{(i)} = \boldsymbol{\theta}^{(i)}_{h,\lambda_1,\lambda_2} + \frac{t_i - 1}{t_{i+1}}(\boldsymbol{\theta}^{(i)}_{h,\lambda_1,\lambda_2} - \boldsymbol{\theta}^{(i-1)}_{h,\lambda_1,\lambda_2})$

$\quad \mathbf{v}^{(i)} = \boldsymbol{\eta}^{(i)} - \frac{\partial}{L_\theta \partial \boldsymbol{\theta}} F(\boldsymbol{\eta}^{(i)})$

$\quad \widehat{\pi}^0_{\lambda_2}(\mathbf{v}^{(i)}) = \text{argADMM}(\mathbf{v}^{(i)})$

$\quad \widehat{\pi}^{\lambda_1}_{\lambda_2}(\mathbf{v}^{(i)}) = \text{sign}(\widehat{\pi}^0_{\lambda_2}(\mathbf{v}^{(i)})) \odot \max\{|\widehat{\pi}^0_{\lambda_2}(\mathbf{v}^{(i)})| - \lambda_1, 0\}$

$\quad \boldsymbol{\theta}^{(i+1)}_{h,\lambda_1,\lambda_2} = \widehat{\pi}^{\lambda_1}_{\lambda_2}(\mathbf{v}^{(i)})$

$\quad t_{i+1} = \frac{1+\sqrt{1+4t_i^2}}{2}$

$\widehat{\boldsymbol{\theta}}_{h,\lambda_1,\lambda_2} = \boldsymbol{\theta}^{(M_1)}_{h,\lambda_1,\lambda_2}$

$\widehat{\boldsymbol{\theta}}_{m,\lambda_1,\lambda_2} = (\mathbf{B}_m^\top \mathbf{B}_m)^{-1} \left(\mathbf{B}_m^\top \mathbf{y} - \mathbf{B}_m^\top \mathbf{B}_h \widehat{\boldsymbol{\theta}}_{h,\lambda_1,\lambda_2} \right)$

Function argADMM() refers to the augmented ADMM algorithm with the updating rule in Equation 1.6. M_1 is the number of the FISTA iterations, and M_2 is the number of augmented ADMM iterations. The choice of $L_\theta, \rho, \mathbf{Q}$ can be found in subsection 1.8.4.

1.3.3 Selection of Bases in Our Context

This section discusses how to choose the bases $\mathbf{B}_{m,1}$, $\mathbf{B}_{m,2}$, $\mathbf{B}_{m,3}$, $\mathbf{B}_{h,1}$, $\mathbf{B}_{h,2}$, $\mathbf{B}_{h,3}$ in our contexts. In general, these bases can be Gaussian kernels, Cosine kernels, etc., depending on the nature or characteristics of the data. When one has little to no prior knowledge of the data structure, a simple choice of basis can be an identity matrix.

Let us now discuss our choices of these bases in our simulation studies and case study of the crime rate dataset. For the bases of the global trend mean, it involves the selection of $\mathbf{B}_{m,1}$, $\mathbf{B}_{m,2}$, $\mathbf{B}_{m,3}$, where $\mathbf{B}_{m,1}$ is the basis in the state dimension of the global trend, $\mathbf{B}_{m2}$ is the basis in the crime rate dimension of the global trend, $\mathbf{B}_{m3}$ is the basis in the temporal dimension of the global trend. By Figure 1.2, the data are spatially smooth in the state dimension, and thus we propose to choose $\mathbf{B}_{m,1}$ as the Gaussian kernel matrix whose (i, j)-th element is defined by $\exp\{-d^2/(2c^2)\}$, where d is the distance between the center of i-th

state and j-th state. Here the bandwidth constant c is chosen by Silverman's Rule of thumb of [31]: $c = (4\widehat{\sigma}^5/(3n_3))^{1/5}$, where $\widehat{\sigma}$ is the estimated variance of $\mathbf{y}_1, \ldots, \mathbf{y}_{n_3}$. Meanwhile, we set $\mathbf{B}_{m,2}$ and $\mathbf{B}_{m,3}$ as the identity matrix, since we have little prior knowledge in the crime rate dimension and the temporal dimension.

For the selection of the hot-spots basis, i.e., $\mathbf{B}_{h,1}$, $\mathbf{B}_{h,2}$, $\mathbf{B}_{h,3}$, we propose to set all of them as the identity matrix, since there is no prior knowledge of the hot-spots. It is informative to mention that while the identify matrix seems to lack the temporal consistency of the hot-spots, our optimization problem adds the fused Lasso penalty that might have already addressed the temporal consistency of the hot-spots.

1.4 Detection and Localization of Hot-spots

In this section, we discuss the detection and localization of the hot-spots. For the ease of presentation, we first discuss the detection of the hot-spots, i.e., detect when a hot-spot occurs in subsection 1.4.1. Then, in subsection 1.4.2, we consider the localization of the hot-spot, i.e., determine which states and which crime types are involved in the detected hot-spots.

1.4.1 Detect When Hot Spots Occur?

To detect when hot-spots occur, we develop a control chart based on the following hypothesis test problem:

$$H_0: \ \mathbf{r}_t(\lambda_1, \lambda_2) = 0 \quad \text{v.s.} \quad H_1: \ \mathbf{r}_t(\lambda_1, \lambda_2) = \delta \widehat{\mathbf{h}}_t(\lambda_1, \lambda_2) \quad (\delta > 0), \tag{1.8}$$

where

$$\mathbf{r}_t(\lambda_1, \lambda_2) = \mathbf{y}_t(\lambda_1, \lambda_2) - \widehat{\boldsymbol{\mu}}_t(\lambda_1, \lambda_2)$$

is the residual after removing the global trend mean under the penalty parameters λ_1, λ_2, and the vector $\widehat{\boldsymbol{\mu}}_t(\lambda_1, \lambda_2) = \text{vec}\left(\widehat{\mathcal{U}}_{::t}(\lambda_1, \lambda_2)\right), \widehat{\mathbf{h}}_t(\lambda_1, \lambda_2) = \text{vec}\left(\widehat{\mathcal{H}}_{::t}(\lambda_1, \lambda_2)\right)$ are the

estimated global trend mean and local hot-spots in t-th year. Here, we add (λ_1, λ_2) to emphasize that, $\widehat{\boldsymbol{\mu}}_t(\lambda_1, \lambda_2), \widehat{\mathbf{h}}_t(\lambda_1, \lambda_2)$ are the global trend mean and local hot-spots estimation under penalty parameter λ_1, λ_2, respectively.

The motivation of the above hypothesis test is described as follows. When there are no hot-spots, the residual $\mathbf{r}_t(\lambda_1, \lambda_2)$ is exactly the model noises. However, when hot-spots exist, the residual $\mathbf{r}_t$ includes both hot-spots and noises. By including the hot-spot information of $\widehat{\mathbf{h}}_t(\lambda_1, \lambda_2)$ in the alternative hypothesis, we hope to provide a direction in the alternative hypothesis space, which allows one to construct a test with more power [see 8].

Next, we construct the likelihood ratio test in the above-mentioned hypotheses testing problem. By [32], the test statistics monitoring upward shift is

$$P_t^+(\lambda_1, \lambda_2) = \widehat{\mathbf{h}}_t^+(\lambda_1, \lambda_2)^\top \, \mathbf{r}_t(\lambda_1, \lambda_2) \Big/ \sqrt{\widehat{\mathbf{h}}_t^+(\lambda_1, \lambda_2)^\top \, \widehat{\mathbf{h}}_t^+(\lambda_1, \lambda_2)}$$

where $\widehat{\mathbf{h}}_t^+(\lambda_1, \lambda_2)$ only takes the positive part of $\widehat{\mathbf{h}}_t(\lambda_1, \lambda_2)$ with other entries as zero, because our objective is to detect positive hot-spots. The superscript "+" emphasizes that we aim at detecting upward shifts. In other words, we focus on the hot-spots that have increasing means, partly because increasing crime rates are generally more harmful to societies and communities. If one is also interested in detecting decreasing mean shifts, one could modify it by using a two-sided test.

It remains to discuss how to choose (λ_1, λ_2) suitably in our test. We propose to follow [8] to calculate a series of $P_t^+(\lambda_1, \lambda_2)$ under different combination of $(\lambda_1, \lambda_2) \in \Gamma = \{(\lambda_1^{(1)}, \lambda_2^{(1)}) \ldots (\lambda_1^{(n_\lambda)}, \lambda_2^{(n_\lambda)})\}$ and then select the combination of (λ_1, λ_2) with the largest power. The final chosen test statistics, denoted as $\widetilde{P}_t^+(\lambda_{1,t}^*, \lambda_{2,t}^*)$, can be computed by

$$\widetilde{P}_t^+(\lambda_{1,t}^*, \lambda_{2,t}^*) = \max_{(\lambda_1, \lambda_2) \in \Gamma} \frac{P_t^+(\lambda_1, \lambda_2) - E(P_t^+(\lambda_1, \lambda_2))}{\sqrt{\text{Var}(P_t^+(\lambda_1, \lambda_2))}}, \tag{1.9}$$

where $E(P_t^+(\lambda_1, \lambda_2))$, $\text{Var}(P_t^+(\lambda_1, \lambda_2))$ respectively are the mean and variance of $P_t^+(\lambda_1, \lambda_2)$ under H_0 (e.g., for phase-I in-control samples). Here $(\lambda_{1,t}^*, \lambda_{2,t}^*) \in \Gamma$ is the penalty param-

eter maximizing the above equation.

With the test statistic available, we detect when hot-spots occur based on the widely used cumulative sum (CUSUM) control chart [see 33, 34]. At each time t, we recursively compute the CUSUM statistics as

$$W_t^+ = \max\{0, W_{t-1}^+ + \widetilde{P}_t^+(\lambda_{1,t}^*, \lambda_{2,t}^*) - d^*\}, \tag{1.10}$$

with the initial value $W_{t=0}^+ = 0$, where d^* is a constant and can be chosen according to the degree of the shift that we want to detect. Then we declare that a hot-spot might occurs whenever $W_t^+ > L$ for some pre-specified control limit L.

Note that the CUSUM statistics W_t^+ leads to the optimal control chart to detecting a mean shift from μ_0 to $\mu_1 = 2d^* - \mu_0$ for normally distributed data [see 34]. When the data are not normally distributed, the optimality properties might not hold, but it can still be a reasonable control chart. Also, it is important to choose the control limit L in the CUSUM control chart suitably, and the detailed discussion will be presented in section 1.5 for our simulation studies and in section 1.6 for our case study.

1.4.2 Localize Where and Which the Hot Spots Occur?

In this section, we discuss how to localize the hot-spots if the CUSUM control chart in Equation 1.10 raises an alarm at year t^*. In other words, we want to determine where and which crime rates may account for the hot-spots. To do so, we propose to utilize the matrix $\widehat{\mathcal{H}}_{::t^*}(\lambda_{1,t^*}, \lambda_{2,t^*})$, which is the hot-spot estimation in t^*-th year. If the (i, j)-th entry in $\widehat{\mathcal{H}}_{::t^*}(\lambda_{1,t^*}, \lambda_{2,t^*})$ is non-zero, then we declare that there is a hot-spot for the j-th crime rate type in the i-th state at the t^*-th year.

The mathematical procedure to derive $\widehat{\mathcal{H}}_{::t^*}(\lambda_{1,t^*}, \lambda_{2,t^*})$ is described as follows. First, $\widehat{\mathcal{H}}(\lambda_{1,t^*}, \lambda_{2,t^*})$ is the tensor format of $\widehat{\mathbf{h}}(\lambda_{1,t^*}, \lambda_{2,t^*}) = \mathbf{B}_h\widehat{\boldsymbol{\theta}}_h(\lambda_{1,t^*}, \lambda_{2,t^*})$, where $\widehat{\boldsymbol{\theta}}_h(\lambda_{1,t^*}, \lambda_{2,t^*})$ is the minimizer in Equation 1.2 under the penalty parameter $\lambda_{1,t^*}, \lambda_{2,t^*}$. Second, $\widehat{\mathcal{H}}_{::t^*}(\lambda_{1,t^*}, \lambda_{2,t^*})$

is the t^* slices along the temporal dimension of $\widehat{\mathcal{H}}(\lambda_{1,t^*}, \lambda_{2,t^*})$.

As one reviewer points out, this approach might lead to a relatively high false positive rate (FPR), since some non-zero entries might not be statistically significant. Two possible ways to improve our approach are (1) to conduct the significant test, or (2) to set up a pre-specified threshold and only keep the positive entries that are larger than the threshold. It is useful to investigate how to improve our approach, which is an interesting topic for future research. Here we focus on our main ideas of using tensor decomposition for hot-spots and adopt the simple approach for hot-spots localization.

1.5 Simulation Study

In this section, we report the numerical simulation results of our proposed method as well as its comparison with several benchmark methods in the literature. To better present our results, we divide this section into several subsections. In subsection 1.5.1, we introduce the data generation mechanism for our simulation studies, and subsection 1.5.2 presents the benchmark methods for the comparison purpose. The performance of hot-spot detection and localization are reported in subsection 1.5.3, and the fitness of the global trend mean is evaluated in subsection 1.5.4.

1.5.1 Data Generation

In our simulation, we detect the hot-spots on the complete data, i.e., $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. In order to shed light on the case study, we match the tensor dimension of the crime rate dataset and choose $n_1 = 51, n_2 = 3, n_3 = 250$ in our simulation. To generate the data $\mathcal{Y}$, we generate it by generating its front slices $\mathcal{Y}_{::t}$ for $t = 1 \dots n_3$. Mathematically, we generate $\mathbf{y}_t = \text{vec}(\mathcal{Y}_{::t})$ by

$$\mathbf{y}_{i,t} = (\mathbf{B}\boldsymbol{\theta}_t)_i + \delta \mathbb{1}\{t \geq \tau\} \mathbb{1}\{i \in S_h\} + \mathbf{w}_{i,t}, \tag{1.11}$$

where $\mathbf{y}_{i,t}$ denotes the i-th entry in the vector $\mathbf{y}_t \in \mathbb{R}^{n_1 n_2}$. The last term on the right hand side of the above equation, i.e., $\mathbf{w}_{i,t}$, is the i-th entry in the white noise vector $\mathbf{w}_t \in \mathbb{R}^{n_1 n_2}$. And $\mathbf{w}_t$ is a vector whose entries are independent and follow $N(0, 0.1^2)$ distribution. Note that while we generate the data sequentially over time t, our proposed SSD-Tensor method is actually an off-line method that analyzes the complete tensor.

The first term on the right-hand side of the above equation, i.e., $(\mathbf{B}\boldsymbol{\theta}_t)_i$, is the global mean, where the subscript $(\cdot)_i$ denotes the i-th entry. The matrix $\mathbf{B}$ is a fixed B-spline basis with the degree of three and fifty knots evenly spacing on the interval $[1, 50]$. Note that the B-spline basis is only used in the generative model in simulation to generate data, but is not used in our proposed methodologies. Vector $\boldsymbol{\theta}_t$ is a constant parameter controlling the trend of the global mean. We set $\boldsymbol{\theta}_t$ in two different ways, so we discuss 2 scenarios:

- Scenario 1: The global trend mean is stationary, in which $\boldsymbol{\theta}_{t,i} = 1+\sin\left(\frac{2\pi}{470}(106.75 + i)\right)$ for $i = 1, \ldots n_1 n_2$ and $t = 1, \ldots, n_3$. Here $\boldsymbol{\theta}_{t,i}$ is the i-th entry in $\boldsymbol{\theta}_t$.

- Scenario 2: The global trend mean is decreasing over time, in which $\boldsymbol{\theta}_t = 0.995\boldsymbol{\theta}_{t-1}$ for $t = 1, 2, \ldots, n_3$. And $\boldsymbol{\theta}_{0,i} = 1 + \sin\left(\frac{2\pi}{470}(106.75 + i)\right)$ for $i = 1, \ldots n_1 n_2$.

The second term on the right side hand of Equation 1.11, i.e., $\delta \mathbb{1}\{t \geq \tau\}\mathbb{1}\{i \in S_h\}$ is the local hot-spot, where $\mathbb{1}(A)$ is the indicator function, which has the value 1 for all elements of A and the value 0 for all elements not in A. First, $\mathbb{1}\{t \geq \tau\}$ indicates that the hot-spots only occur after the hot-spot τ. This ensures that the simulated hot-spot is temporal consistent. The second indicator function $\mathbb{1}\{i \in S_h\}$ shows that only those entries whose location index belongs set S_h are assigned as local hot-spots. This ensures that the simulated hot-spot is sparse. Here we assume the change happens at $\tau = 200$ and the hot-spots index set $S_h = \{3, 15, 16, 19, 20, 23, 31, 35, 42, 48, 54, 66, 67, 70, 71, 72, 74, 82, 86, 92, 105, 118, 121, 122, 125, 133, 137, 140, 144, 151\}$. The hot-spots only account for around 20%, so it is sparse. Parameter $\delta \in \mathbb{R}$ denotes the change magnitude. In our simulation studies, two change magnitudes are considered, one is $\delta/\sigma = 1$ (small shift) and the other is $\delta/\sigma = 5$

(large shift). Here we follow the existing research [see 12, 35] to set the variance of the white noise as 0.1^2. Note that the white noise standard deviation $\sigma = 0.1$ might seem to be small, but we want to emphasize that the σ value itself is not crucial here, and the signal-to-noise-ratio (SNR), i.e., δ/σ, is more fundamental.

Here is the detailed implementation of our proposed SSD-Tensor method. For the selection of basis, we use the same bases as in subsection 1.3.3. For the penalty parameters $(\lambda_1, \lambda_2) \in \Gamma$, we set $\Gamma = \{(\lambda_1, \lambda_2) : \lambda_1 \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}$ and $\lambda_2 \in \{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}\}$. Since our proposed method is an off-line method that uses the complete data, our simulation setting on the *average run length under the out-of-control status* (ARL_1) is slightly different from the standard SPC literature. In standard SPC literature, ARL_1 of a procedure with the stopping time N is defined as $E_\tau(N - \tau + 1 | N \geq \tau)$ when the true change occurs at a given time $\tau = 1$. While in our paper, we choose $\tau = 200$ to better illustrate real-world applications. To be more specific, in each Monte Carlo run for ARL_1, we simulate a complete tensor data with $n_3 = 250$ years and the change-time $\tau = 200$ years. Next, we focus only those runs that the control chart raises an alarm at $T \geq 200$ (if $T < 200$ it will be counted as false alarm), and then define the detection delay, or ARL_1, as $\mathbf{E}_\tau(T - \tau + 1 | T \geq \tau)$ with the change time $\tau = 200$.

1.5.2 Benchmark Methods

In this section, we present the description and implementation of benchmark methods that will be used to compare with our method.

The first benchmark method is the scan statistics method in [6], which is a Bayesian extension of Kulldorff's scan statistic. The reason for us to choose [6] is that it has large power to detect clusters and it has a fast runtime. In our paper, we use the a R function called *scan_bayes_negbin()* from the package *scanstatistics*. To implement this function, the population size is needed. For a fair comparison, we will not give more data to scan-stat, and simply assume that the population is $100,000$ for all states and all years.

Because *scan_bayes_negbin()* can only handle one type of crime rate one time, we apply *scan_bayes_negbin()* to three crime rates separately and set the probability of an outbreak as $0.02/3$. Because the scan-statistics-based method does not give the clear calculation of *average run length under the in-control status* (ARL_0) and ARL_1, so we can only use the probability of an outbreak as $0.02/3$ to define the control limit to achieve similar ARL_0 with other benchmarks.

The second benchmark method is the Lasso-based method in [8]. The main idea of [8] is to integrate the multivariate exponentially weighted moving average (EWMA) charting scheme. Under the assumption that the hot-spots are sparse, the Lasso model is applied to the EWMA statistics. If the Mahalanobis distance between the expected response (the Lasso estimator) and observed values is larger than a pre-specified control limit, temporal hot-spots are detected, with non-zero entries of the Lasso estimator are declared as spatial hot-spots. For the control limits and the penalty parameters of the Lasso-based method, we use the same criterion as our proposed SSD-Tensor method.

The third benchmark is the dimension-reduction method of [17] that uses PCA to extract a set of uncorrelated new features that are linear combinations of original variables. Note that [17] fails to localize the spatial hot-spots, and it can only detect the temporal change-point when the PCA-projected Mahalanobis distance is larger than a pre-specified control limit. For this control limit, we set it by using the same criterion as our proposed SSD-Tensor method. In both our simulations and case study, we select three principle components, since they can explain more than 90% cumulative percentage of variance (CPV).

Finally, the fourth benchmark is the traditional T2 control chart [see 36] method with the control limit set using the same criterion as our proposed SSD-Tensor method. Since the T2 control chart method is well-defined, we skip the detailed description, and more details can be found in [36].

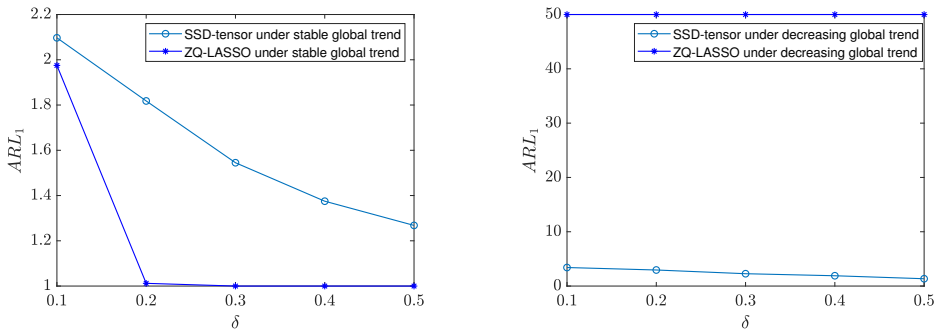
1.5.3 Performance on Hot-spot detection and localization

In this section, we compare our proposed SSD-Tensor method with four benchmark methods with the focus on the performance of hot-spots detection and localization. The four benchmark methods are scan-statistics method proposed by [6] (denoted as 'scan-stat'), Lasso-based control chart proposed by [8] (denoted as 'ZQ-Lasso'), PCA-based control chart proposed by [17] (denoted as 'PCA'), and the traditional Hotelling T^2 control chart in [36] (denoted as 'T2'). All simulation results below are based on 1000 Monte Carlo replications.

Let us first compare the performance of hot-spots detection over the time domain, i.e., when the hot-spots occur. The criterion we use to measure the performance of detection is ARL_1. Because ARL_1 measures the delay after the change occurs, the smaller the ARL_1, the better detection performance. The results can be found in Table 1.2, Table 1.4. For our proposed SSD-Tensor, it has a small ARL_1 under all scenarios, no matter there is a stable or decreasing global trend. This illustrates that our proposed SSD-Tensor method can provide a rapid alarm after the hot-spots occur, even if there are stable or unstable global trends. This good behavior is due to the ability to capture both temporal consistency and spatial sparsity of the hot-spots. For the scan-statistics method, it is hard to estimate the exact ARL_1 because it focuses on the hot-spots localization, not sequential change point detection. So we will not report the ARL_1 of it. For ZQ-Lasso, it successfully detects the hot-spots when the global trend is stable, but unfortunately, it fails to do so when there is a decreasing global trend. The latter is not surprising because ZQ-Lasso is unable to separate the global trend and local hot-spots. For PCA and T2, both of them fail to detect the hot-spots in all scenarios within the entire $n_3 = 250$ (simulated) years, as their $\text{ARL}_1 = 50$. The reason for the unsatisfying results of PCA and T2 is that they are designed based on a multivariate hypothesis test on the global mean change, which cannot take into account the non-stationary global mean trend and the sparsity of the hot-spots.

We visualize the hot-spots detection results in Figure 1.5, which illustrates the trend

of the detection delay, ARL_1, of all methods, as δ changes from 0.1 to 0.5 with the step size of 0.1. Because scan-stat does not represent the ARL_1 and PCA, T2 fail to detect hot-spots in all scenarios, we only plot the ARL_1 of SSD-Tensor and ZQ-Lasso. From the plot, our proposed SSD-Tensor method, compared with ZQ-Lasso, shares similar detection delays when there is a stable global trend. However, our proposed SSD-Tensor method has much smaller detection delays than ZQ-Lasso, particularly when there is a decreasing global trend mean and the magnitude of the hot-spot is small. Also, it is interesting to note that the detection delays of all methods are decreasing as the magnitude of the hot-spot is increasing, which is consistent with our intuition that it is easier to detect larger changes.



(a) ARL_1 under a stable global trend (b) ARL_1 under a decreasing global trend

Figure 1.5: (a) ARL_1 under a stable global trend, (b) ARL_1 under a decreasing global trend.

Next, let us compare the performance on hot-spots localization of these methods, i.e., localize where the hot-spots occur. To evaluate the localization performance of all the methods, we will compute the following four criteria: (1) precision, defined as the proportion of detected hot-spots that are true hot-spots; (2) recall, defined as the proportion of the hot-spots that are correctly identified; (3) F-measure, a single criterion that combines the precision and recall by calculating their harmonic mean. Moreover, we also compare the true positive rate (TPR), true negative rate (TNR), false positive rate (FPR), and false negative rate (FNR). The localization performance measuring in precision, recall, F-measure can be found in Table 1.2, Table 1.4, and the localization performance measuring in TPR, TNR,

FPR, FNR can be found in Table 1.3, Table 1.5. For our proposed SSD-Tensor method, its localization performance is satisfactory no matter there is a stable or unstable global trend. For instance, when there is a decreasing global trend and $\delta = 0.5$, our method has 32.42% precision and 99.78% recall, which outperforms those of scan-stat and ZQ-Lasso. The only weakness of our proposed SSD-Tensor method is that it has a relatively high FPR: this is consistent with our expectation since we did not conduct the significance test of the positive entry in $\widehat{\mathcal{H}}_{::t^*}$. For scan-stat, it has very similar precision as our proposed SSD-Tensor method (both are around 30%), but its recall is much lower. This is because scan-stat tends to detect clustered hot-spots, which results in detecting fewer hot-spots and missing some true hot-spots. This might also explain why scan-stat has low TPR, but high FNR. It is worth noting that the precision/recall/F-measure might be overestimated for scan-stat: we record all the Monte Carlo runs, even if it is a false alarm since scan-stat fails to report ARL_1. For ZQ-Lasso, it has 19.61% precision and 100% recall when the global trend is stable, but it has a very high FPR. This is because ZQ-Lasso fails to detect the significance of the non-zero entries, and declares all non-zero entries as hot-spots. However, when there is a decreasing global trend, ZQ-Lasso fails to detect hot-spots, so we represent its localization performance as 0. This unsatisfactory performance is due to the inability of ZQ-Lasso to separate the global trend and local hot-spots, particularly in Scenario 2 (decreasing global trend mean). For PCA and T2, they cannot localize hot-spots, and thus we do not report the corresponding values on the precision, recall, F-measure, TPR, TFR, FPR, and FNR.

Moreover, we also visualize the hot-spot localization results in Figure 1.6. In the first row, the blue states are the true hot-spots (true positive), whereas the white states are the normal states (true negative). In the second row, the red states are the detected hot-spots (true positive + false positive) by scan-stat. In the third row, the red states are the detected hot-spots by our proposed SSD-Tensor method. Different color represents how likely it is hot-spot: the darker red, the more likely it is. From Figure 1.6, we can see that scan-stat

Table 1.2: Scenario 1 (stable global trend mean): Comparison of hot-spot detection under small and large hot-spots with 4 criterions: precision, recall, F measure and ARL_1

methods	large shift $\delta/\sigma = 5$				small shift $\delta/\sigma = 1$			
	precision	recall	F measure	ARL	precision	recall	F measure	ARL
SSD-Tensor	**0.3210**	**0.9898**	**0.6554**	**1.2680**	**0.3217**	**0.9890**	**0.6553**	**2.0970**
	(0.0345)	(0.0893)	(0.0599)	(0.3321)	(0.0362)	(0.0946)	(0.0634)	(0.8979)
Scan-stat	0.3109	0.4664	0.3887	-	0.2242	0.3364	0.2803	-
	(0.0020)	(0.0030)	(0.0025)	(-)	(0.0301)	(0.0451)	(0.0376)	(-)
ZQ-Lasso	0.1961	1.0000	0.5980	1.0000	0.1961	1.0000	0.5980	1.9750
	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(2.1291)
PCA	-	-	-	50.0000	-	-	-	50.0000
	-	-	-	(0.0000)	-	-	-	(0.0000)
T2	-	-	-	50.0000	-	-	-	50.0000
	-	-	-	(0.0000)	-	-	-	(0.0000)

Table 1.3: Scenario 1 (stable global trend mean): Comparison of hot-spot detection under small and large hot-spots under 4 criteria: TPR, TNR, FPR, FNR.

methods	large shift $\delta/\sigma = 5$				small shift $\delta/\sigma = 1$			
	TPR	TNR	FPR	FNR	TPR	TNR	FPR	FNR
SSD-Tensor	**0.9898**	**0.4848**	**0.5072**	**0.0022**	**0.9890**	**0.4865**	**0.5045**	**0.0020**
	(0.0893)	(0.0616)	(0.0630)	(0.0085)	(0.0946)	(0.0635)	(0.0648)	(0.0080)
Scan-stat	0.4664	0.7479	0.2521	0.5336	0.3364	0.7162	0.2838	0.6636
	(0.0030)	(0.0007)	(0.0007)	(0.0003)	(0.0451)	(0.0110)	(0.0110)	(0.0451)
ZQ-Lasso	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000
	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0000)

Table 1.4: Scenario 2 (decreasing global trend mean): Comparison of hot-spot detection under small and large hot-spots with 4 criterions: precision, recall, F measure and ARL_1

methods	large shift $\delta/\sigma = 5$				small shift $\delta/\sigma = 1$			
	precision	recall	F measure	ARL	precision	recall	F measure	ARL_1
SSD-Tensor	**0.3242**	**0.9978**	**0.6610**	**1.3490**	**0.3245**	**0.9978**	**0.6612**	**3.4120**
	(0.0184)	(0.0085)	(0.0105)	(0.4762)	(0.0188)	(0.0085)	(0.0108)	(0.8217)
Scan-stat	0.3111	0.4666	0.3889	-	0.2567	0.3851	0.3209	-
	(0.0007)	(0.0011)	(0.0009)	(-)	(0.0307)	(0.0460)	(0.0383)	(-)
ZQ-Lasso	0.0000	0.0000	0.0000	50.0000	0.0000	0.0000	0.0000	50.0000
	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0000)
PCA	-	-	-	50.0000	-	-	-	50.0000
	-	-	-	(0.0000)	-	-	-	(0.0000)
T2	-	-	-	50.0000	-	-	-	50.0000
	-	-	-	(0.0000)	-	-	-	(0.0000)

Table 1.5: Scenario 2 (decreasing global trend mean): Comparison of hot-spot detection under small and large hot-spots under 4 criteria: TPR, TNR, FPR, FNR.

methods	large shift $\delta/\sigma = 5$				small shift $\delta/\sigma = 1$			
	TPR	TNR	FPR	FNR	TPR	TNR	FPR	FNR
SSD-Tensor	**0.9978**	**0.4903**	**0.5097**	**0.0022**	**0.9978**	**0.4909**	**0.5091**	**0.0022**
	(0.0085)	(0.0422)	(0.0422)	(0.0085)	(0.0085)	(0.0431)	(0.0431)	(0.0085)
Scan-stat	0.4666	0.7480	0.2520	0.5334	0.3851	0.7281	0.2719	0.6149
	(0.0011)	(0.0003)	(0.0003)	(0.0011)	(0.0460)	(0.0112)	(0.0112)	(0.0460)
ZQ-Lasso	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0000)

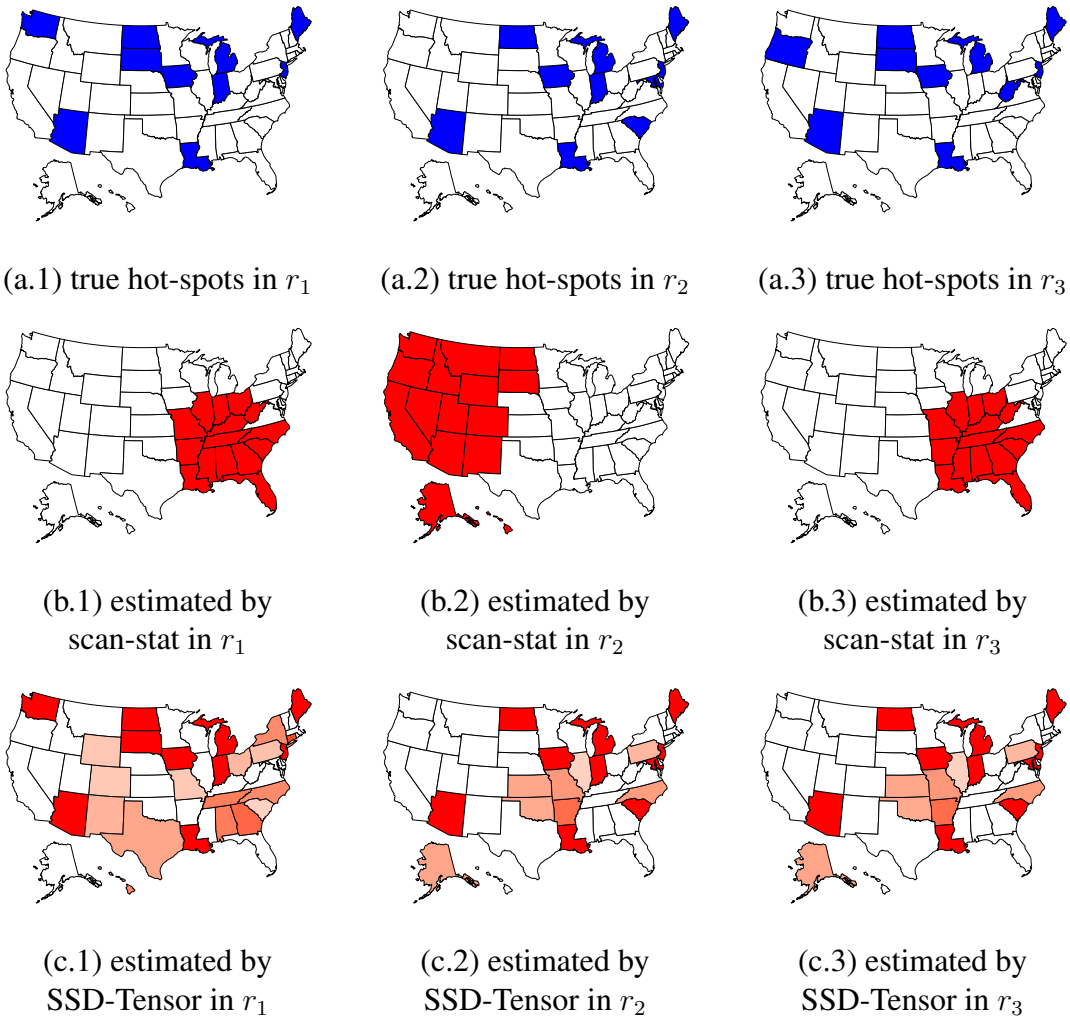
tends to detect clustered hot-spots. Meanwhile, there is no clear pattern for the hot-spots detected by our proposed SSD-Tensor method. We need to acknowledge that while our proposed method can detect most of the true hot-spots (the blue states), but there are some false alarms. We should emphasize that this kind of false alarm is reasonable on the hot-spot localization, which is related to the multiple hypothesis testing in the high-dimensional data.

1.5.4 Background Fitness

In this section, we illustrate that our proposed SSD-Tensor method leads to a reasonably well estimation for the global trend mean. To do this, we compare the squared-root of mean square error (SMSE) of the fitness of a global trend mean in Table 1.6. Here we only show our proposed method, since other baseline methods (scan-stat, ZQ-Lasso, PCA, and T2) cannot model the global trend mean. It is clear from Table 1.6 that our proposed SSD-Tensor method performs well in terms of the background fitness, especially in Scenario 2 (decreasing global trend).

Table 1.6: SMSE in two scenarios under different δ/σ of the hot-spot.

methods	$\delta/\sigma = 1$	$\delta/\sigma = 2$	$\delta/\sigma = 3$	$\delta/\sigma = 4$	$\delta/\sigma = 5$
SSD-Tensor	Scenario 1 (stationary global trend mean)				
	0.0075	**0.0076**	**0.0075**	**0.0077**	**0.0077**
	(7.1331e-04)	(3.4049e-04)	(8.0032e-04)	(1.0799e-05)	(6.9222e-04)
SSD-Tensor	Scenario 2 (decreasing global trend mean)				
	0.0039	**0.0039**	**0.0039**	**0.0039**	**0.0039**
	(1.2357e-05)	(1.2647e-05)	(1.1054e-05)	(1.1845e-05)	(1.2619e-05)



(a.1) true hot-spots in r_1 (a.2) true hot-spots in r_2 (a.3) true hot-spots in r_3

(b.1) estimated by scan-stat in r_1 (b.2) estimated by scan-stat in r_2 (b.3) estimated by scan-stat in r_3

(c.1) estimated by SSD-Tensor in r_1 (c.2) estimated by SSD-Tensor in r_2 (c.3) estimated by SSD-Tensor in r_3

Figure 1.6: Hot-spots detection performance by SSD-Tensor and scan-stat in Scenario 2 (decreasing global trend mean) with large hot-spots ($\delta = 0.5$). In the first row, the blue states are the true hot-spots (true positive), whereas the white states are the normal states (true negative). In the second row, the red states are the detected hot-spots (true positive + false positive) by scan-stat. In the third row, the red states are the detected hot-spots by our proposed SSD-Tensor method. Different color represents how likely it is hot-spot: the darker red, the more likely it is. In these figures, r_1, r_2, r_3 represent the first, second, third category, respectively. To match the dimension of our motivating data set, we choose three categories to match the three types of crime rates in our motivating crime rate dataset, namely legacy rape rate, murder and non-negligent manslaughter, and revised rape rate.

1.6 Case Study

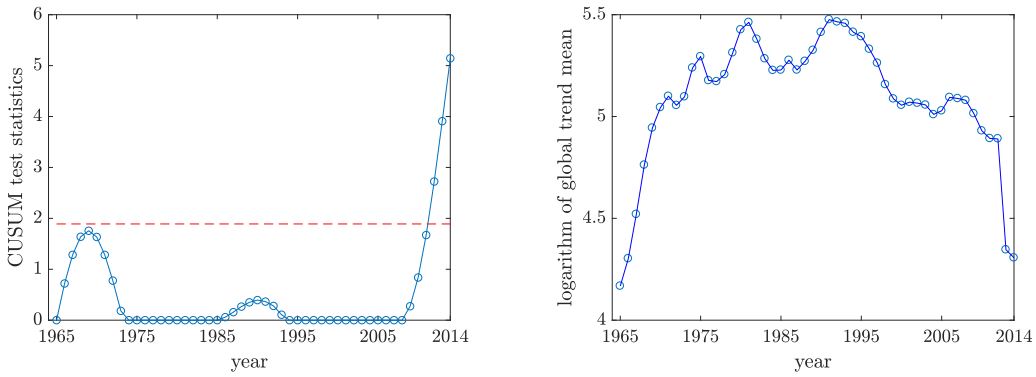
In this section, we apply our proposed SSD-Tensor method to detect and localize hot-spots in the crime rate dataset described in section 1.2, and compare its performance with other benchmarks.

First, we compare the performance of the detection delay of the hot-spot. For all the methods, we set the control limits so that the average run length to false alarm constraint $\text{ARL}_0 = 50$ via Monte Carlo simulation under the assumption that data from the first 20 years are in control. For the setting of the parameters and the selection of basis, they are the same as that in section 1.5. For our proposed SSD-Tensor method, we build a CUSUM control chart utilizing the test statistic in section 1.4, which is shown in Figure 1.7(a). From this figure, we can see that the hot-spots are detected in 2012 by our proposed SSD-Tensor method. For the benchmark methods for comparison, we also apply scan-stat [see 6], ZQ-Lasso [see 8], PCA [see 17] and T2 [see 36] to the crime rate dataset and summarize the performance of the detection of a hot-spot in Table 1.7. Note that the value in Table 1.7 is the first year that raises alarm, i.e, $\min_{t=1986,\ldots,2014}\{t : W_t^+ > L\}$. Our proposed SSD-Tensor method raises an alarm of hot-spots in the year 2012, while other benchmarks fail to detect any hot-spots (we do not represent the hot-spots year of scan-stat, as it does not report the hot-spots). While nobody knows the true hot-spots of the real dataset such as the crime rate data, our numerical simulation experiences suggest that year 2012 is likely a hot-spot.

Table 1.7: Detection of change-point year in crime rate dataset. The label "Year when an alarm is raised" is first year that raises alarm, i.e, $\min_{t=1986,\ldots,2014}\{t : W_t^+ > L\}$, where W_t^+ is the CUSUM statistics defined in Equation 1.10, and L is control limits to achieve the average run length to false alarm constraint $\text{ARL}_0 = 50$ via Monte Carlo simulation under the assumption that data from the first 20 years are in control.

methods	SSD-Tensor	scan-stat	ZQ-Lasso	PCA	T2
Year when an alarm is raised	**2012**	-	None	None	None

Next, after the detection of hot-spots, we need to further localize the hot-spots in the

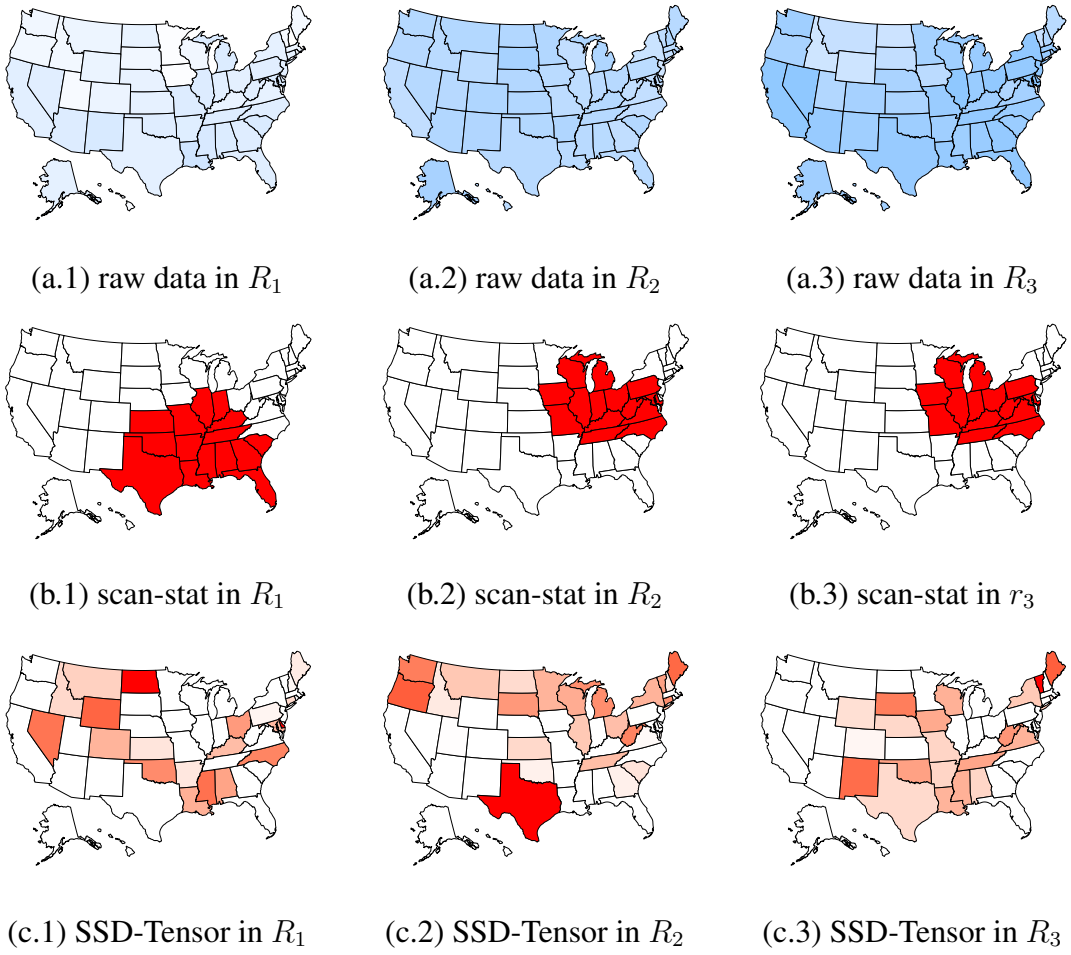


(a) temporal change-point detection (b) observed data points and fitted mean curve

Figure 1.7: (a) Hot-spot detection by SSD-Tensor. Since the CUSUM test statistics of 2012, 2013 and 2014 exceed the threshold (red dashed line), we declare that year 2012 is the first change-points. (b) Fitted global trend mean and the observed annual data points. Each point (blue circle) is an actual observed annual data in logarithm scale, which is fitted by a fitted mean curve (blue line).

sense of determining which state and which type of crime rates may lead to the occurrence of a hot-spot. Because the baseline methods, PCA and T2, can only detect when the changes happen and ZQ-Lasso fails to detect any changes, we only show the localization of hot-spot by SSD-Tensor and scan-stat, where the results are visualized in Figure 1.8. We also represent the raw crime rates in 2012 in the first row of Figure 1.8, where the darker the blue, the high value of the crime rates. The first row (a.1)-(a.3) in this figure represents the raw data, where we can see that all three types of crime rates share a very smooth background, so it becomes difficult to find out the hot-spots directly from the raw data. The second row (b.1)-(b.3) and third (c.1)-(c.3) in this figure represents the hot-spots (TP + FP) detected by scan-stat [see 6] and SSD-Tensor, respectively. Scan-stat tends to detect clustered hot-spots, while there is no obvious pattern of the hot-spots detected by our proposed SSD-Tensor method. Thus, as compared to scan-stat, our proposed SSD-Tensor method seems to detect sparse hot-spots, which might be useful in practice when identifying where the sudden increase of crime happens.

Finally, we also represent the fitted global trend mean curve in 7(b), where the blue line is the fitted mean curve, and the blue circle is the observed annual data points in the



(a.1) raw data in R_1 (a.2) raw data in R_2 (a.3) raw data in R_3

(b.1) scan-stat in R_1 (b.2) scan-stat in R_2 (b.3) scan-stat in r_3

(c.1) SSD-Tensor in R_1 (c.2) SSD-Tensor in R_2 (c.3) SSD-Tensor in R_3

Figure 1.8: Localization of hot-spot. (a.1)-(a.3): the raw data of the three crime rates in 2012. (b.1)-(b.2): the detected hot-spots by scan-stat. (c.1)-(c.3): the detected hot-spots by SSD-Tensor. The red color of the states means that this is a hot-spot state. And the deeper the color, the larger the hot-spots size. In these figures, R_1 is the legacy rape rate, R_2 is the murder and non-negligent manslaughter, and R_3 is the revised rape rate.

logarithm scale. Note that the actual observed data in 2012 is a little bit higher than the fitted mean curve, which may lead to hot-spots detection in 2012. Overall, the fitness of our proposed method works very well, which can help us greatly remove the effect of the global trend.

1.7 Conclusion

In this paper, we propose the *SSD-Tensor* method for hot-spot detection and localization in multivariate spatio-temporal data, which is an important problem in many real-world applications. The main idea is to represent the multi-dimension data as a tensor, and then to decompose the tensor into the global trend mean, local hot-spots, and residuals. The estimation of model parameters and hot-spots is formulated as an optimization that includes the sum of residual squares with both Lasso and fused Lasso penalties, which control the sparsity and the temporal consistency of the hot-spots, respectively. Moreover, we develop an efficient algorithm to solve these optimization problems for parameter estimation by using the FISTA algorithm. In addition, we compare our proposed SSD-Tensor method with other benchmarks through Monte Carlo simulations and the case study of the crime rate dataset.

Clearly, there are many opportunities to improve the algorithms and methodologies. First, it would be interesting to investigate the confounding between the global trend and local hot-spots in future research. In our paper, we assume that they are additive, but it is possible that the increasing global trend yields an increased number of hot-spots. Second, the significance test of the non-zero entries of $\widehat{\mathcal{H}}_{::t^*}$ can be promising future research, so we can reduce the false positive rate (FPR) in localizing the hot-spots. Third, it will be useful to extend our method to the context of missing or incomplete data. We feel it is straightforward to combine our method with the imputation method when missing is (completely) at random, but it remains an open problem if missing is not at random. Fourth, in this paper, we fix the tensor basis, and it will be useful to investigate the robustness effects of different

tensor bases, or better yet, to adopt some data-driven method to learn the bases from the data. Fifth, a promising research direction is to combine our proposed SSD-Tensor method with the spatially adaptive method in [37] for trend filtering. Finally, it will be interesting to derive the theoretical properties of our proposed method including the sufficient or necessary conditions under which our hot-spot estimation properties.

1.8 Supplementary Material

1.8.1 Construction of Matrix $\mathbf{D}$

The main idea to construct matrix $\mathbf{D}$ is to make

$$\lambda_2 \left\| \boldsymbol{D}\boldsymbol{\theta}_{h,\lambda_1,\lambda_2} \right\|_1 = \lambda_2 \sum_{t=2}^{n_3} \left\| \boldsymbol{\theta}_{h,\lambda_1,\lambda_2,t} - \boldsymbol{\theta}_{h,\lambda_1,\lambda_2,t-1} \right\|_1 .$$

Here are more details. Matrix $\mathbf{D}$ can be constructed as:

$$\mathbf{D} = \mathbf{D}_1 \otimes \mathbf{D}_2 \otimes \mathbf{D}_3,$$

where matrix $\mathbf{D}_1 \in \mathbb{R}^{n_1 \times n_1}, \mathbf{D}_2 \in \mathbb{R}^{n_2 \times n_2}$ are the identity matrix. And matrix $\mathbf{D}_3 \in \mathbb{R}^{(n_3-1)\times n_3}$ is defined as

$$\mathbf{D}_{3,i,j} = \begin{cases} -1 & \text{if } i = j \\ 1 & \text{if } i = j-1 \\ 0 & \text{otherwise} \end{cases},$$

where $\mathbf{D}_{3,i,j}$ is the (i,j)-th entry in matrix $\mathbf{D}_3$.

1.8.2 Proof of Fast Calculation of $\mathbf{y}^*$ via Tensor Algebra

Proof.

$$
\begin{aligned}
\mathbf{y}^* &= [\mathbf{I} - \mathbf{B}_m(\mathbf{B}_m^\top \mathbf{B}_m)^{-1}\mathbf{B}_m^\top]\mathbf{y} \\
&= \mathbf{y} - \mathbf{B}_m((\mathbf{B}_{m,1}^\top \otimes \mathbf{B}_{m,2}^\top \otimes \mathbf{B}_{m,3}^\top)(\mathbf{B}_{m,1} \otimes \mathbf{B}_{m,2} \otimes \mathbf{B}_{m,3}))^{-1})\mathbf{B}_m^\top \mathbf{y} \\
&= \mathbf{y} - \mathbf{B}_m((\mathbf{B}_{m,1}^\top \mathbf{B}_{m,1}) \otimes (\mathbf{B}_{m,2}^\top \mathbf{B}_{m,2}) \otimes (\mathbf{B}_{m,3}^\top \mathbf{B}_{m,3}))^{-1})\mathbf{B}_m^\top \mathbf{y} \\
&= \mathbf{y} - \mathbf{B}_m((\mathbf{B}_{m,1}^\top \mathbf{B}_{m,1})^{-1} \otimes (\mathbf{B}_{m,2}^\top \mathbf{B}_{m,2})^{-1} \otimes (\mathbf{B}_{m,3}^\top \mathbf{B}_{m,3})^{-1})\mathbf{B}_m^\top \mathbf{y} \\
&= \mathbf{y} - (\mathbf{B}_{m,1}(\mathbf{B}_{m,1}^\top \mathbf{B}_{m,1})^{-1}\mathbf{B}_{m,1}^\top) \otimes (\mathbf{B}_{m,2}(\mathbf{B}_{m,2}^\top \mathbf{B}_{m,2})^{-1}\mathbf{B}_{m,2}^\top) \\
&\quad \otimes (\mathbf{B}_{m,3}(\mathbf{B}_{m,3}^\top \mathbf{B}_{m,3})^{-1}\mathbf{B}_{m,3}^\top)\mathbf{y} \\
&= \mathbf{y} - \text{vec}(\mathcal{Y} \times_1 (\mathbf{B}_{m,1}(\mathbf{B}_{m,1}^\top \mathbf{B}_{m,1})^{-1}\mathbf{B}_{m,1}^\top) \times_2 (\mathbf{B}_{m,2}(\mathbf{B}_{m,2}^\top \mathbf{B}_{m,2})^{-1}\mathbf{B}_{m,2}^\top) \\
&\quad \times_3 (\mathbf{B}_{m,3}(\mathbf{B}_{m,3}^\top \mathbf{B}_{m,3})^{-1}\mathbf{B}_{m,3}^\top))
\end{aligned}
$$

□

1.8.3 Review of All Benchmarks

For the scan-stat-based method, we select [6] as the representative. The goal of [6] is to find regions (a collection of location index) which has a high posterior probability to be an anomalous cluster. To realize this objective, [6] compare the null hypothesis H_0 is no anomalous clusters with a set of alternative $H_1(S)$, each representing a cluster in some region S. The anomalous cluster S^* is declared as the one with the highest posterior probabilities $P(H_1(S)|D)$, i.e.,

$$S^* = \arg\max_S \frac{P(D_j|H_1(S))}{P(D_j|H_0)},$$

where dataset $D_j = \{\mathcal{C}_{i,j,t}\}_{i=1,\ldots,n_1,t=1,\ldots,n_3}$ with $C_{i,j,t}$ as the number of crime counts in index (i, j, t). And $\mathcal{C}_{i,j,t}$ is assumed to follow Poisson distribution, i.e., $\mathcal{C}_{i,j,t} \sim \text{Poisson}(q\mathcal{B}_{i,j,t})$,

where $\mathcal{B}_{i,j,t}$ represents the (known) population of index (i, j, t) and q is the (unknown) underlying crime rate. To calculate the posterior probability $P(H_1(S)|D_j)$, a hierarchical Bayesian model is used, where q are drawn from Gamma distribution. For selection of prior parameters of the Gamma distribution, please see [6] for the detailed description. For each j in $\{1, \ldots, n_2\}$, the above procedure is repeated to find anomalous regions in all type of rates.

For Lasso-based methods, we select [8] as the representative. In a given time t, [8] builds a multivariate exponentially weighted moving average (EWMA) on data $\mathbf{x}_t = (\mathcal{Y}_{1,1,t}, \ldots, \mathcal{Y}_{n_1,1,t}, \mathcal{Y}_{1,2,t}, \ldots, \mathcal{Y}_{n_1,n_2,t})^\top$: $\mathbf{u}_t = \alpha\mathbf{x}_t + (1-\alpha)\mathbf{u}_{t-1},$ for $t = 1, \ldots, n_3$ where $\mathbf{u}_0 = \mathbf{0}$ and α is a weighting parameter in $(0, 1]$. For each $\mathbf{u}_t$, a Lasso estimator $\widehat{\boldsymbol{\mu}}$ is derived from the penalized likelihood function $(\mathbf{u}_t - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{u}_t - \boldsymbol{\mu}) + \lambda \sum_{k=1}^{n_1 n_2} \frac{|\boldsymbol{\mu}_k|}{\mathbf{u}_{k,t}}$, where $\boldsymbol{\Sigma}$ is the covariance matrix of the normal distribution which $\mathbf{x}_t$ follows. Temporal change point is detected when

$$\max_{k=1,\ldots,q} \frac{W_{t,\lambda^*} - E(W_{t,\lambda^*})}{\sqrt{\text{Var}(W_{t,\lambda^*})}} > L_{LASSO},$$

where $L_{LASSO} > 0$ is the control limit chosen to achieve a given in-control average run length. $W_{t,\lambda^*} = \max_{\lambda \in \{\lambda_1,\ldots,\lambda_q\}} \frac{W_{t,\lambda} - E(W_{t,\lambda})}{\sqrt{\text{Var}(W_{t,\lambda})}}$ and $W_{t,\lambda} = \frac{2-\alpha}{\alpha[1-(1-\lambda)^{2t}]} \frac{(\mathbf{u}_t^\top \boldsymbol{\Sigma}^{-1}\widehat{\boldsymbol{\mu}})^2}{\widehat{\boldsymbol{\mu}}^\top \boldsymbol{\Sigma}^{-1}\widehat{\boldsymbol{\mu}}}$. The spatial hot-spots are localized at non-zero entries of $\widehat{\boldsymbol{\mu}}$.

For the dimension-reduction methods, we select [17] as the representative. The main tool of this method is PCA, which defines a linear relationship between the original variables of the data set, mapping them to a set of uncorrelated variables. [17] fails to localize the spatial hot-spots, and it can only realize detecting the temporal change-point when

$$(\mathbf{x}_t - \bar{\mathbf{x}})^\top \mathbf{P}_k \boldsymbol{\Lambda}_k^{-1} \mathbf{P}_k^\top (\mathbf{x}_t - \bar{\mathbf{x}}) > L_{pca},$$

where L_{pca} is the control limit chosen to achieve a given in-control average run length. Vector $\bar{\mathbf{x}} = \frac{1}{n_1 n_2}\mathbf{X}^\top \mathbf{1}$ with $\mathbf{X} = (x_1, \ldots, x_{n_3})$ and $\mathbf{1} = (1, 1, \ldots, 1)_{n_1 n_2}^\top$. Matrix $\mathbf{P}_k, \boldsymbol{\Lambda}_k$ is derived from the covariance matrix of $\mathbf{X}$, i.e., $\mathbf{S} = \frac{1}{n_1 n_2 - 1}(\mathbf{X} - \mathbf{1}\bar{\mathbf{x}}^\top)^\top(\mathbf{X} - \mathbf{1}\bar{\mathbf{x}}^\top) = \mathbf{P}\boldsymbol{\Lambda}\mathbf{P}^\top$,

where $\mathbf{P}$ is the loading vectors of $\mathbf{S}$ and $\mathbf{\Lambda} = \text{diag}(\lambda^1, \ldots, \lambda^{n_1 n_2})$ contains the eigenvalues of $\mathbf{S}$ in descending order. The correlation of $\mathbf{P}$ and $\mathbf{P}_k$ is that $\mathbf{P}_k$ contains only the first k columns of $\mathbf{P}$, where k is the number of components selected. The selection criterion is the cumulative percentage of variance (CPV), and in our case, we select $k = 3$. The correlation of $\mathbf{\Lambda}$ and $\mathbf{\Lambda}_k$ is that, $\mathbf{\Lambda}_k = \text{diag}(\lambda^1, \ldots, \lambda^k)$. Also, we select the traditional T2 control chart [36] method as a benchmark. Because the T2 control chart method is very well-defined, we ignore the detailed description of it in this paper, and the details of it can be found in [36].

1.8.4 Choice of $L_\theta, \mathbf{Q}, \rho,$ in Algorithm 1

For the choice of L_θ, the theoretical value can set as the Lipschitz constant of $F(\boldsymbol{\theta}) = \|\mathbf{y}^* - \mathbf{X}\boldsymbol{\theta}\|_2^2$. In practice, researchers often use the maximal eigenvalue of the Hessian matrix of $F(\boldsymbol{\theta})$ as the fixed L_θ [see 24], i.e., the maximal eigenvalue of matrix $\mathbf{X}^\top\mathbf{X}$. Please note that the value of L_θ is fixed throughout our algorithm, and guarantees the convergence of the algorithm.

For choice of $\mathbf{Q}$, a simple choice to ensure $\mathbf{Q} \succeq \mathbf{D}^\top\mathbf{D}$ would be $\mathbf{Q} = \delta\mathbf{I}$, where $\mathbf{I}$ is the identity matrix and $\delta \geq \|\mathbf{D}\|_{\text{op}}^2$ with $\|\mathbf{D}\|_{\text{op}}$ denoting the operator norm of matrix $\mathbf{D}$ [see 30]. Please note that $\mathbf{Q}$ is fixed throughout our algorithm, and guarantees the convergence of the algorithm.

For the choice of ρ, it has been theoretically proved that any choice of ρ will lead to the convergence of the algorithm, but a good choice of ρ would help us realize faster convergence [see 30]. In practice, a good choice of ρ is to start with an initial value $\rho^{(0)} = 5$ and then update it in each iteration. The updating rule from $\rho^{(k-1)}$ to $\rho^{(k)}$ is:

$$\rho^{(k)} = \begin{cases} \widetilde{\eta}\rho^{(k-1)} & \text{if } \|\mathcal{R}^{(k)}\|_2/\ddot{\epsilon} \geq \widetilde{\mu}\|\mathcal{S}^{(k+1)}\|_2/\dot{\epsilon} \\ \widetilde{\eta}^{-1}\rho^{(k-1)} & \text{if } \|\mathcal{S}^{(k)}\|_2/\dot{\epsilon} \geq \widetilde{\mu}\|\mathcal{R}^{(k)}\|_2/\ddot{\epsilon}. \end{cases},$$

where $\mathcal{R}^{(k)} = \mathbf{D}\pi^0_{\lambda_2}(\mathbf{v}^{(i)})^{(k)} - \boldsymbol{\gamma}^{(k)}$ and $\mathcal{S}^{(k)} = \rho^{(k)}\mathbf{D}^\top\left(\boldsymbol{\gamma}^{(k)} - \boldsymbol{\gamma}^{(k-1)}\right)$. The explicit format of $\pi^0_{\lambda_2}(\mathbf{v}^{(i)})^{(k)}, \boldsymbol{\gamma}^{(k)}$ can be found in Equation 1.6. Besides, $\dot{\epsilon}, \ddot{\epsilon}$ is the dual and

primal feasibility, i.e., $\|\mathcal{R}^{(k)}\|_2 \leq \ddot{\epsilon}$ and $\left\|\mathcal{R}^{(k)}\right\|_2 \leq \dot{\epsilon}$. In practice, $\dot{\epsilon}, \ddot{\epsilon}$ can be set as $10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}$, depending on the desired precision. The parameter $\widetilde{\eta}, \widetilde{\mu}$ are set to be 2 and 10 as suggested by [38]. More details about the setting of ρ can be found in Section 2.3 in [30] and [38].

CHAPTER 2
RAPID DETECTION OF HOT-SPOT BY TENSOR DECOMPOSITION WITH APPLICATION TO WEEKLY GONORRHEA DATA

2.1 Introduction

In many bio-surveillance and healthcare applications, data sources are measured from many spatial locations repeatedly over time, say, daily, weekly, or monthly. In these applications, we are typically interested in detecting *hot-spots*, which are defined as some structured outliers that are sparse over the spatial domain but persistent over time. A concrete real-world motivating application is the weekly number of gonorrhea cases from 2006 to 2018 of 50 states in the United States, also see the detailed data description in the next section. From the monitoring viewpoint, there are two kinds of changes: one is the global-level trend, and the other is the local-level outliers. Here we are more interested in detecting the so-called hot-spots, which are local-level outliers with the following two properties: (1) spatial sparsity, i.e., the local changes are sparse over the spatial domain; and (2) temporal persistence, i.e., the local changes last for a reasonably long period of time unless one takes some actions.

Generally speaking, the hot-spot detection can be thought of as detecting sparse anomalies in the spatio-temporal data, and there are three different categories of methodologies and approaches in the literature. The first one is the least absolute shrinkage and selection operator (Lasso) based control chart that integrates Lasso estimators for change point detection and declares non-zero components of the Lasso estimators as the hot-spot [see 10, 11, 39]. Unfortunately, the Lasso-based control chart cannot separate the local hot-spots from the global trend of the spatio-temporal data. The second category of methods is the dimension reduction-based control chart where one monitors the features from PCA

or other dimension reduction methods [see 17, 18, 19]. The drawback of PCA or other dimension reduction methods is that it fails to detect sparse hot-spots and cannot take full advantage of the spatial location of hot-spots. The third category of anomaly detection methods is the decomposition-based method that uses the regularized regression methods to separate the hot-spots from the background event, see [12, 40, 41]. However, these existing approaches investigate structured images or curve data under the assumption that the hot-spots are independent over the time domain.

In this paper, we propose a decomposition-based anomaly detection method for spatial-temporal data when the hot-spots are autoregressive, which is typical for time series data. Our main idea is to represent the raw data as a 3-dimensional tensor: states, weeks, years. To be more specific, at each year, we observe a 50×52 data matrix that corresponds to 50 states and 52 weeks (we ignore the leap years). Next, we propose to decompose the 3-dimension tensor into three components: Smooth global trend, Sparse local hot-spot, and Residuals, and term our proposed decomposition model as SSR-Tensor. When fitting the observed raw data to our proposed SSR-Tensor model, we develop a penalized likelihood approach by adding two penalty functions: one is the Lasso type penalty to guarantee the sparsity of hot-spots, and the other is the fused-Lasso type penalty for the autoregressive properties of hot-spots. By doing so, we are able to (1) detect when the hot-spots occur (i.e., the change point detection problem); and (2) localize where and which type of the hot-spots occur (i.e., the spatial localization problem).

We would like to acknowledge that much research has been done on modeling and prediction of the spatio-temporal data. Some popular time series models are auto-regressive (AR), moving-average (MA), auto-regressive-moving-average (ARMA) model, etc., and the parameters can be estimated by Yule-Walker method [see 42], maximum likelihood estimation or least square method [see 43]. In addition, spatial statistics have also been extensively investigated on its own right [see 44, 45, 46, 47, 48]. When one combines time series with spatial statistics, the corresponding spatio-temporal models generally become

more complicated, [see 49, 50, 51].

In principle, it is possible to represent the spatio-temporal process as a sequence of random vector $\mathbf{y}_t$ with weekly observation t, where $\mathbf{y}_t$ is a p-dimensional vector that characterizes the spatial domain (i.e., spatial dimension $p = 50$ in our case study).

However, such an approach might not be computationally feasible in the context of hot-spot detection, in which one needs to specify the covariance structure of $\mathbf{y}_t$, not only over the spatial domain, but also over the time domain. If we write all data into a vector, then the dimension of such vector is $50 \times 52 \times 13 = 33,800$, and thus the covariance matrix is of dimension $33,800 \times 33,800$, which is not computationally feasible [see 52, 53]. Meanwhile, under our proposed SSR-Tensor model, we essentially conduct a dimensional reduction by assuming that such a covariance matrix has a nice sparsity structure, as we reduce the dimensions $50, 52$ and 13 to much smaller numbers, e.g., AR(1) model over the week or year dimension, and local correlation over the spatial domain.

It is useful to point out that while our paper focuses only on 3-dimensional tensor due to our motivating application in gonorrhea, our proposed SSR-Tensor model can easily be extended to any d-dimensional tensor or data with $d \geq 3$, e.g., when we have further information, such as the unemployment rates, economic performance, and so on. As the dimension d increases, we can simply add more corresponding bases, as our proposed model uses *basis* to describe correlation within each dimension, and utilizes *tensor product* for interaction between different dimensions. The capability of extending to high-dimensional data is one of the main advantages of our proposed SSR-Tensor model. Furthermore, our proposed SSR-Tensor model essentially involves a block-wise diagonal covariation matrix, which allows us to develop computationally efficient methodologies by using tensor decomposition algebra, see subsection 2.5.2 for more technical details.

The remainder of this paper is described as follows. In section 2.2, we discuss and visualize the gonorrhea dataset, which is used as our motivating example and in our case study. In section 2.3, we present our proposed SSR-Tensor model, and discuss how to

estimate model parameters from the observed data. In section 2.4, we describe how to use our proposed SSR-Tensor model to find hot-spots, both for temporal detection and spatial localization. Efficient numerical optimization algorithms are discussed in section 2.5. Our proposed methods are then validated through extensive simulations in section 2.6 and a case study in the gonorrhea dataset in section 2.7.

2.2 Data Description

To protect Americans from serious diseases, the National Notifiable Disease Surveillance System (NNDSS) at the Centers for Disease Control and Prevention (CDC) helps public health monitor, control, and prevent about 120 diseases (see its website https://wwwn.cdc.gov/nndss/infectious-tables.html). One disease that receives intensive attention in recent years is gonorrhea, due to the possibility of multi-drug resistance. Historically the instances of antibiotic resistance (in gonorrhea) have first been in the west and then move across the country. Since 1965, the CDC has collected the number of cumulative new infected patients every week in a calendar year. There are several changes to report policies or guidelines, and the latest one is the year 2006. As a result, we focus on the weekly numbers of new gonorrhea patients during January 1, 2006, and December 31, 2018. The new weekly gonorrhea cases are computed as the difference of the cumulative cases in two consecutive weeks. The last week is dropped during this calculation.

Let us first discuss the spatial patterns of the gonorrhea data among 50 states. For this purpose, we consider the cumulative number of gonorrhea cases from week 1 to week 52 by summing up all data during the years 2006-2018. Figure 2.1 plots some selected weeks (#1, #11, #21, #31, #41, #51). In Figure 1, if the state has a deeper and bluer color, then it experiences a higher number of gonorrhea cases. One obvious pattern is that California and Texas have generally higher number of gonorrhea cases as compared to other states. In addition, the number of gonorrhea cases in the northern US is smaller than that in the southern US.

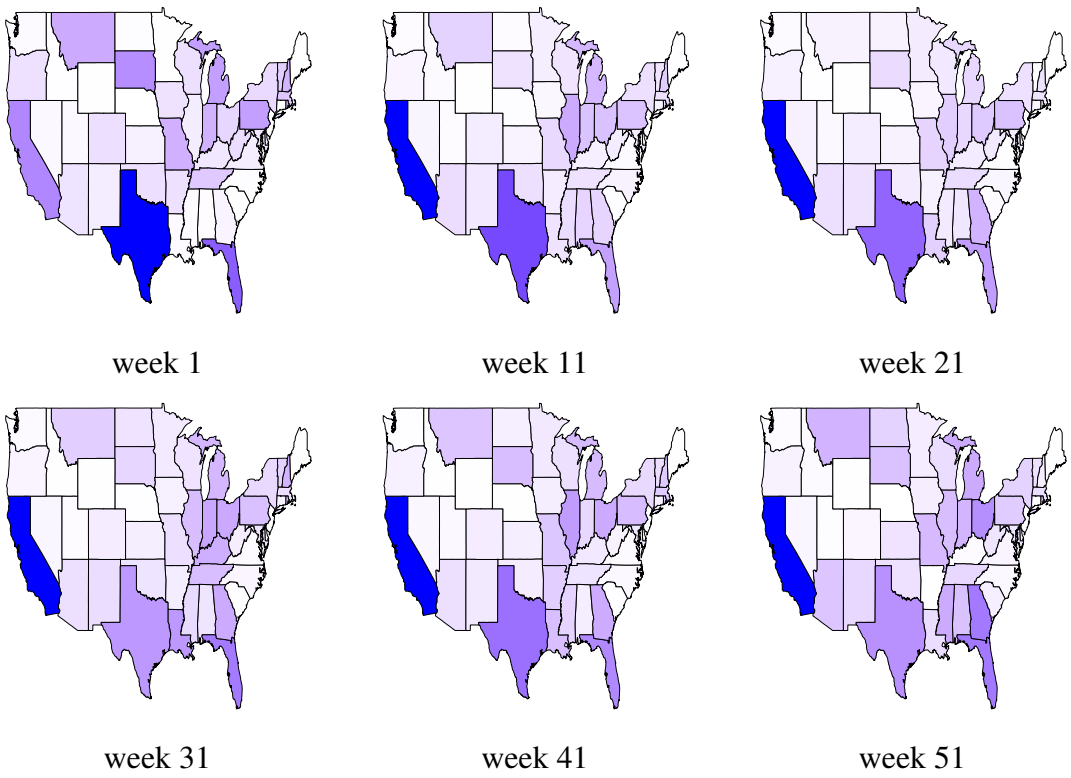


Figure 2.1: The cumulative number of gonorrhea cases at some selected weeks during years 2006-2018. The deeper the color, the higher number of gonorrhea cases.

Next, we consider the temporal pattern of the gonorrhea data set. Figure 2.2 plots the annual number of gonorrhea cases over the years 2006-2018 in the US. It can be seen that there is a decrease during 2007- 2009, and then the number of gonorrhea cases increases. The increasing trend from 2010 to 2014 is very gentle, but the increasing trend after 2015 becomes severe. One possible explanation for the different increase speed is the Affordable Care Act, which was signed into law by President Barack Obama on March 23, 2010. This policy may help to stabilize the increase of gonorrhea disease. As we mentioned before, we are not interested in detecting this type of global changes, and we focus on the detection of the changes in the local patterns, which are referred to as hot-spots in our paper.

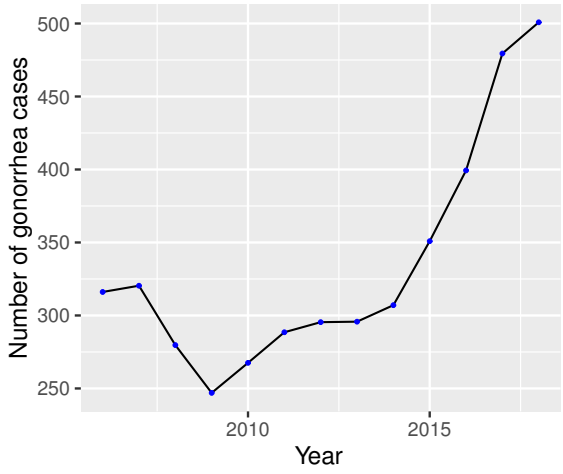


Figure 2.2: Annual number of gonorrhea cases (in thousands) over the years 2006-2018 in the US

Moreover, the gonorrhea data consists of weekly data, and thus it is necessary to address the circular patterns over the direction of "week". Figure 2.3 shows the country-scaled weekly gonorrhea case in the form of a "rose" diagram for some selected years. In this figure, each direction represents a given week, and the length represents the number of gonorrhea cases for a given week. Figure 2.3 reveals differences in the number of gonorrhea cases across a different week of the year. For instance, in July and August (in the direction of 8 o'clock on the circle), the number of gonorrhea cases tends to be larger than in other weeks.

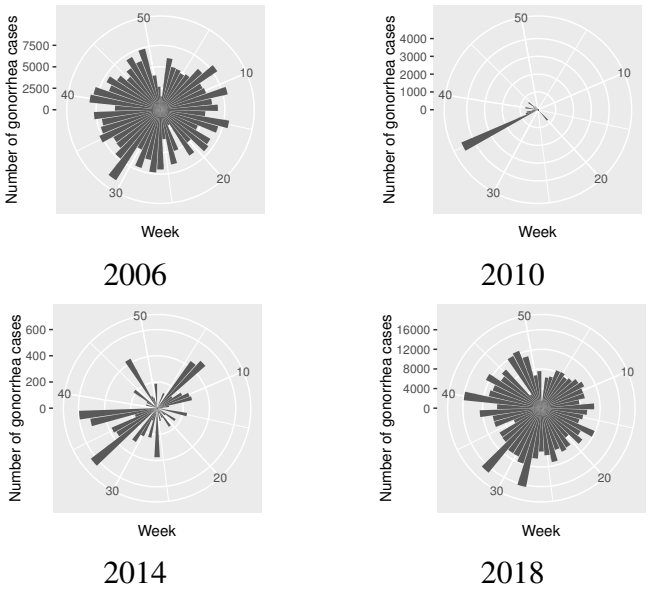


Figure 2.3: Circular histograms of the number of gonorrhea cases of the year 2006, 2010, 2014, 2018. The y-axis is the number of gonorrhea cases, and the circular x-axis is 51 weeks. Each bar represents a given week, and the length represents the number of gonorrhea cases for a given week in the US.

2.3 Proposed Model

In this section, we present our proposed SSR-Tensor model and postpone the discussion of hot-spot detection methodology to the next section. Owing to the fact that the gonorrhea data is of three dimensions, namely, {state, week, year}, it will likely have complex "within-dimension" correlation and "between-dimension" interaction. Within-dimension correlation includes within-state correlation, within-week correlation, and within-year correlation. The between-dimension relationship includes between-state-and-week interaction, between-state-and-year interaction, as well as between-week-and-year interaction. In order to handle these complex "within" and "between" interaction structures, we propose to use the tensor decomposition method, where bases are used to address "within-dimension" correlation, and the tensor product is used for "between-dimension" interaction. Here, the basis is a very important concept where different bases can be chosen for different dimensions. Detailed discussions of the choice of bases are presented in subsection 2.6.2.

For the convenience of notation and easy understanding, we first introduce some basic tensor algebra and notation in subsection 2.3.1. Then subsection 2.3.2 presents our proposed model that is able to characterize the complex correlation structures.

2.3.1 Tensor Algebra and Notation

In this section, we introduce basic notations, definitions, and operators in tensor (multilinear) algebra that are useful in this paper. Throughout the paper, scalars are denoted by lowercase letters (e.g., θ), vectors are denoted by lowercase boldface letters ($\boldsymbol{\theta}$), matrices are denoted by uppercase boldface letter ($\boldsymbol{\Theta}$), and tensors by curlicue letter (ϑ). For example, an order-N tensor is represented by $\vartheta \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, where I_n represent the mode-n dimension of ϑ for $n = 1, \ldots, N$.

The mode-n product of a tensor $\vartheta \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ by a matrix $\mathbf{B} \in \mathbb{R}^{J_n \times I_n}$ is a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \ldots I_{n-1} \times J_n \times I_{n+1} \times \ldots I_N}$, denoted as $\mathcal{A} = \vartheta \times_n \mathbf{B}$, where each entry of $\mathcal{A}$ is defined as the sum of products of corresponding entries in $\mathcal{A}$ and $\mathbf{B}$: $\mathcal{A}_{i_1,\ldots,i_{n-1},j_n,i_{n+1},\ldots,i_N} = \sum_{i_n} \vartheta_{i_1,\ldots,i_N} \mathbf{B}_{j_n,i_n}$. Here we use the notation $\mathbf{B}_{j_n,i_n}$ to refer the (j_n, i_n)-th entry in matrix $\mathbf{B}$. The notation $\vartheta_{i_1,\ldots,i_N}$ is used to refer to the entry in tensor ϑ with index $(i_1, \ldots, i_N)$. The notation $\mathcal{A}_{i_1,\ldots,i_{n-1},j_n,i_{n+1},\ldots,i_N}$ is used to refer the entry in tensor $\mathcal{A}$ with index $(i_1, \ldots, i_{n-1}, j_n, i_{n+1}, \ldots, i_N)$.

The mode-n unfold of tensor $\vartheta \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ is noted by $\vartheta_{(n)} \in \mathbb{R}^{I_n \times (I_1 \times \ldots I_{n-1} \times I_{n+1} \times I_N)}$, where the column vector of $\vartheta_{(n)}$ are the mode-n vector of ϑ. The mode-n vector of ϑ are defined as the I_n dimensional vector obtained from ϑ by varying the index i_n while keeping all the other indices fixed. For example, $\vartheta_{:,2,3}$ is a model-1 vector.

A very useful technique in the tensor algebra is the Tucker decomposition, which decomposes a tensor into a core tensor multiplied by matrices along with each mode: $\mathcal{Y} = \vartheta \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)}$, where $\mathbf{B}^{(n)}$ is an orthogonal $I_n \times I_n$ matrix and is a principal component mode-n for $n = 1, \ldots, N$. Tensor product can be represented equivalently by a Kronecker product, i.e., $\mathrm{vec}(\mathcal{Y}) = (\mathbf{B}^{(N)} \otimes \cdots \otimes \mathbf{B}^{(1)})\mathrm{vec}(\vartheta)$, where

$\mathrm{vec}(\cdot)$ is the vectorized operator. Finally, the definition of Kronecker product is as follow: Suppose $\mathbf{B}_1 \in \mathbb{R}^{m\times n}$ and $\mathbf{B}_2 \in \mathbb{R}^{p\times q}$ are matrices, the Kronecker product of these matrices, denoted by $\mathbf{B}_1 \otimes \mathbf{B}_2$, is an $mp \times nq$ block matrix defined by

$$\mathbf{B}_1 \otimes \mathbf{B}_2 = \begin{bmatrix} b_{11}\mathbf{B}_2 & \cdots & b_{1n}\mathbf{B}_2 \\ \vdots & \ddots & \vdots \\ b_{m1}\mathbf{B}_2 & \cdots & b_{mn}\mathbf{B}_2 \end{bmatrix}.$$

2.3.2 Our Proposed SSR-Tensor Model

Our proposed SSR-Tensor model is built on tensors of order three, as it is inspired by the gonorrhea data, which can be represented as a three-dimension tensor $\mathcal{Y} \in \mathbb{R}^{n_1\times n_2\times T}$ with $n_1 = 50$ states, $n_2 = 51$ weeks, and $T = 13$ years. Note that the i-th, j-th, and k-th slice of the 3-D tensor along the dimension of state, week, and year can be achieved as $\mathcal{Y}_{i::}, \mathcal{Y}_{:j:}, \mathcal{Y}_{::k}$ correspondingly, where $i = 1\cdots n_1$, $j = 1\cdots n_2$ and $k = 1\cdots T$. For simplicity, we denote $\mathbf{Y}_k = \mathcal{Y}_{::k}$. We further denote $\mathbf{y}_k$ as the vectorized form of $\mathbf{Y}_k$, and $\mathbf{y}$ as the vectorized form of $\mathcal{Y}$.

The key idea of our proposed model is to separate the global trend from the local pattern by decomposing the tensor $\mathbf{y}$ into three parts, namely the smooth global trend $\boldsymbol{\mu}$, local hot-spot $\mathbf{h}$, and residual $\mathbf{e}$, i.e. $\mathbf{y} = \boldsymbol{\mu} + \mathbf{h} + \mathbf{h}$. For the first two of the components (e.g. the global trend mean $\boldsymbol{\mu}$ and local hot-spots $\mathbf{h}$), we introduce the basis decomposition framework to represent the structure of the within-dimension correlation in the global background and local hot-spot [see 12].

To be more concrete, we assume that global trend mean and local hot-spot can be represented as $\boldsymbol{\mu} = \mathbf{B}_m\boldsymbol{\theta}_m$ and $\mathbf{h} = \mathbf{B}_h\boldsymbol{\theta}_h$, where $\mathbf{B}_m$ and $\mathbf{B}_h$ are two bases that will discussed below, and $\boldsymbol{\theta}_m$ and $\boldsymbol{\theta}_h$ are the model coefficients vector of length n_1n_2T and needed to be estimated (see section 2.5). Here the subscript of *m* and *h* are abbreviations for mean and hot-spot. Next, it is useful to discuss how to choose the bases $\mathbf{B}_m$ and $\mathbf{B}_h$, so as to charac-

terize the complex "within-dimension" correlation and "between-dimension" interaction. For the "within-dimension" correlation structures, we propose to use pre-specified bases, $\mathbf{B}_{m,s}$ and $\mathbf{B}_{h,s}$, for within-state correlation in global trend and hot-spot, where the subscript of *s* is an abbreviation for states. Similarly, $\mathbf{B}_{m,w}$ and $\mathbf{B}_{h,w}$ are the pre-specified bases for within-correlation of the same week, whereas $\mathbf{B}_{m,y}$ and $\mathbf{B}_{h,y}$ are the bases for within-time correlation over year. As for the "between" interaction, we use tensor product to describe it, i.e, $\mathbf{B}_m = \mathbf{B}_{m,s} \otimes \mathbf{B}_{m,w} \otimes \mathbf{B}_{m,y}$ and $\mathbf{B}_h = \mathbf{B}_{h,s} \otimes \mathbf{B}_{h,w} \otimes \mathbf{B}_{h,y}$. This Kronecker product has been proved to have better computational efficiency in the tensor response data [see 54]. Mathematically speaking, all these bases are matrices, which is pre-assigned in our paper. And the choice of bases in shown in subsection 2.6.2.

With the well-structured "within-dimension" correlation and "between-dimension" interaction, our proposed model can be written as:

$$\mathbf{y} = (\mathbf{e}_{m,s} \otimes \mathbf{B}_{m,w} \otimes \mathbf{B}_{m,y})\boldsymbol{\theta}_m + (\mathbf{B}_{h,s} \otimes \mathbf{B}_{h,w} \otimes \mathbf{B}_{h,y})\boldsymbol{\theta}_h + \mathbf{e}, \tag{2.1}$$

where $\mathbf{e} \sim N(0, \sigma^2\mathbf{I})$ is the random noise. Mathematically speaking, both $\mathbf{B}_{m,s}$ and $\mathbf{B}_{h,s}$ are $n_1 \times n_1$ matrix, $\mathbf{B}_{m,w}$ and $\mathbf{B}_{h,w}$ are $n_2 \times n_2$ matrix and $\mathbf{B}_{m,y}$ and $\mathbf{B}_{h,y}$ are $T \times T$ matrix, respectively.

Mathematically, our proposed model in Equation 2.1 can be rewritten into a tensor format:

$$\mathcal{Y} = \vartheta_m \times_3 \mathbf{B}_{m,y} \times_2 \mathbf{B}_{m,w} \times_1 \mathbf{B}_{m,s} + \vartheta_h \times_3 \mathbf{B}_{h,y} \times_2 \mathbf{B}_{h,w} \times_1 \mathbf{B}_{h,s} + \mathbf{e}, \tag{2.2}$$

where ϑ_m and ϑ_h is the tensor format of $\boldsymbol{\theta}_m$ and $\boldsymbol{\theta}_h$ with dimensional $n_1 \times n_2 \times T$. The tensor representation in Equation 2.2 allows us to develop computationally efficient methods for estimation and prediction.

2.3.3 Estimation of Hot-spots

With the proposed SSR-Tensor model above, we can now discuss the estimation of hot-spot parameters $\boldsymbol{\theta}$'s (including $\boldsymbol{\theta}_m$, $\boldsymbol{\theta}_h$) in our model in Equation 2.1 or Equation 2.2 from the data via the penalized likelihood function. We propose to add two penalties in our estimation. First, because hot-spots rarely occur, we assume that $\boldsymbol{\theta}_h$ is sparse and the majority of entries in the hot-spot coefficient $\boldsymbol{\theta}_h$ are zeros. Thus we propose to add the penalty $R_1(\boldsymbol{\theta}_h) = \lambda_1\|\boldsymbol{\theta}_h\|_1$ to encourage the sparsity property of $\boldsymbol{\theta}_h$. Second, we assume there is temporal continuity of the hot-spots, as the usual phenomenon of last year is likely to affect the performance of the hot-spot in this year. Thus, we add the second penalty $R_2(\boldsymbol{\theta}_h) = \lambda_2\|\mathbf{D}\boldsymbol{\theta}_h\|_1$ to ensure the yearly continuity of the hot-spot, where $\mathbf{D} = \mathbf{D}_s \otimes \mathbf{D}_w \otimes \mathbf{D}_y$ with $\mathbf{D}_s$ as an identical matrix of dimension $n_1 \times n_1$, and matrix $\mathbf{D}_y = \begin{bmatrix} 1 & -1 & & & \\ & \ddots & \ddots & & \\ & & 1 & -1 & \\ & & & 1 & \end{bmatrix} \in \mathbb{R}^{T\times T}$, matrix $\mathbf{D}_w = \begin{bmatrix} 1 & -1 & & & \\ & \ddots & \ddots & & \\ & & 1 & -1 & \\ -1 & & & & 1 \end{bmatrix} \in \mathbb{R}^{n_2\times n_2}$. With the formula of $\mathbf{D}_y$, the hot-spot has the property of yearly continuity. By the formula of $\mathbf{D}_w$, the hot-spot has a weekly circular pattern.

By combining both penalties, we propose to estimate the parameters via the following optimization problem:

$$\begin{aligned} &\arg\min_{\boldsymbol{\theta}_m,\boldsymbol{\theta}_h} \|\boldsymbol{e}\|^2 + \lambda_1\|\boldsymbol{\theta}_h\|_1 + \lambda_2\|\mathbf{D}\boldsymbol{\theta}_h\|_1 \\ &\text{subject to } \ \boldsymbol{y} = (\mathbf{B}_{m,s} \otimes \mathbf{B}_{m,w} \otimes \mathbf{B}_{m,y})\boldsymbol{\theta}_m + (\mathbf{B}_{h,s} \otimes \mathbf{B}_{h,w} \otimes \mathbf{B}_{h,y})\boldsymbol{\theta}_h + \mathbf{e}, \end{aligned} \tag{2.3}$$

where $\boldsymbol{\theta}_m = \text{vec}(\boldsymbol{\theta}_{m,1}, \ldots, \boldsymbol{\theta}_{m,t}, \ldots, \boldsymbol{\theta}_{m,T})$ and $\boldsymbol{\theta}_h = \text{vec}(\boldsymbol{\theta}_{h,1}, , \ldots, \boldsymbol{\theta}_{h,t}, \ldots, \boldsymbol{\theta}_{h,T})$. The choice of the turning parameters λ_1, λ_2 will be discussed in section 2.4.

Note that there are two penalties in Equation 2.3: $\lambda_1\|\boldsymbol{\theta}_h\|_1$ is the Lasso penalty to control both the sparsity of the hot-spots and $\lambda_2\|\mathbf{D}\boldsymbol{\theta}_h\|_1$ is the fused Lasso penalty to control

the temporal consistency of the hot-spots. Traditional algorithms often involve the storage and computation of the matrix $\mathbf{B}_m$ and $\mathbf{B}_h$, which is of the dimension $n_1 n_2 T \times n_1 n_2 T$. Thus they might work to solve the optimization problem in Equation 2.3 when the dimensions are small, but they will be computationally infeasible as the dimensions grow. To address this computational challenge, we propose to simplify the computational complexity by modifying the matrix algebra in traditional algorithm into tensor algebra and will discuss how to optimize the problem in Equation 2.3 computationally efficiently in section 2.5.

2.4 Hot-spot Detection

This section focuses on the detection of the hot-spot, which includes the detection and identification of the year (when), the state (where), and the week (which) of the hot-spots. In our case study, we focus on the upward shift of the number of gonorrhea cases, since the increasing gonorrhea is generally more harmful to societies and communities. Of course, one can also detect the downward shift with a slight modification of our proposed algorithms by multiplying -1 to the raw data.

For the purpose of easy presentation, we first discuss the detection of the hot-spot, i.e., detect when the hot-spot occurs in subsection 2.4.1. Then, in subsection 2.4.2, we consider the localization of the hot-spot, i.e., determine which states and which weeks are involved for the detected hot-spots.

2.4.1 Detect When the Hot Spot Occurs

To determine when the hot-spot occurs, we consider the following hypothesis test and set up the control chart for the hot-spot detection in Equation 2.4.

$$H_0 : \widetilde{\mathbf{r}}_t = 0 \quad v.s. \quad H_1 : \widetilde{\mathbf{r}}_t = \delta \widehat{\mathbf{h}}_t \quad (\delta > 0), \tag{2.4}$$

where $\widetilde{\mathbf{r}}_t$ is the expected residuals after removing the mean. The essence of this test is that, we want to detect whether $\widetilde{\mathbf{r}}_t$ has a mean shift in the direction of $\widehat{\mathbf{h}}_t$, estimated in section 2.5. To test this hypothesis, the likelihood ratio test is applied to the residual $\mathbf{r}_t$ at each time t, i.e. $\mathbf{r}_t = \mathbf{y}_t - \boldsymbol{\mu}_t$, where it assumes that the residuals $\mathbf{r}_t$ is independent after removing the mean and its distribution before and after the hot-spot remains the same. Accordingly, the test statistics monitoring upward shift is designed as $P_t^+ = \widehat{\mathbf{h}}_t^{+\top}\mathbf{r}_t/\sqrt{\widehat{\mathbf{h}}_t^{+\top}\widehat{\mathbf{h}}_t^+}$ [see 32], where $\widehat{\mathbf{h}}_t^+$ only takes the positive part of $\widehat{\mathbf{h}}_t$ with other entries as zero. Here we put a superscript "+" to emphasize that it aims for an upward shift.

The choices of the penalty parameters λ_1, λ_2 are described as follows. In order to select the one with the most power, we propose to calculate a series of P_t^+ under different combination of (λ_1, λ_2) from the set $\Gamma = \{(\lambda_1^{(1)}, \lambda_2^{(1)}) \cdots (\lambda_1^{(n_\lambda)}, \lambda_2^{(n_\lambda)})\}$. For better illustration, we denote the test statistics under penalty parameter (λ_1, λ_2) as $P_t^+(\lambda_1, \lambda_2)$. The test statistics [see 39] with the most power to detect the change, noted as $\widetilde{P}_t^+$, can be computed by

$$\widetilde{P}_t^+ = \max_{(\lambda_1,\lambda_2)\in\Gamma} \frac{P_t^+(\lambda_1, \lambda_2) - E(P_t^+(\lambda_1, \lambda_2))}{\sqrt{Var(P_t^+(\lambda_1, \lambda_2))}}, \tag{2.5}$$

where $E(P_t^+(\lambda_1, \lambda_2)), Var(P_t^+(\lambda_1, \lambda_2))$ respectively are the mean and variance of $P_t(\lambda_1, \lambda_2)$ under H_0 (e.g. for phase-I in-control samples).

Note that the penalty parameter (λ_1, λ_2) to realize the maximization in Equation 2.5 is generally different under different time t. To emphasize such dependence of time t, denote by $(\lambda_{1,t}^*, \lambda_{2,t}^*)$ the parameter pair that attains the maximization in Equation 2.5 at time t, i.e,

$$(\lambda_{1,t}^*, \lambda_{2,t}^*) = \arg\max_{(\lambda_1,\lambda_2)\in\Gamma} \frac{P_t^+(\lambda_1, \lambda_2) - E(P_t^+(\lambda_1, \lambda_2))}{\sqrt{Var(P_t^+(\lambda_1, \lambda_2))}}. \tag{2.6}$$

Thus, the series of the test statistics for the hot-spot at time t is $\widetilde{P}_t^+(\lambda_{1,t}^*, \lambda_{2,t}^*)$ where $t = 1 \cdots T$.

With the test statistic available, we design a control chart based on the CUSUM procedure due to the following reasons: (1) we are interested in detecting the change with

the temporal continuity, therefore, aligns with the objective of CUSUM. (2) In the view of social stability, we want to keep gonorrhea at a target value without sudden changes, which makes the CUSUM chart is a natural better fit.

To be more specific, in the CUSUM procedure, we compute the CUSUM statistics recursively by

$$W_t^+ = \max\{0, W_{t-1}^+ + \widetilde{P}_t^+(\lambda_{1,t}^*, \lambda_{2,t}^*) - d\},$$

and $W_{t=0}^+ = 0$, where d is a constant and can be chosen according to the degree of the shift that we want to detect. Next, we set the control limit L to achieve a desirable ARL for in-control samples. Finally, whenever $W_t^+ > L$ at some time $t = t^*$, we declare that a hot-spot occurs at time t^*.

2.4.2 Localize Where and Which the Hot Spot Occurs?

After the hot-spot t^* has been detected by the CUSUM control chart in the previous section, the next step is to localize where and which week may account for this hot-spot. To do so, we propose to utilize the vector

$$\widehat{\mathbf{h}}_{\lambda_{1,t^*}^*, \lambda_{2,t^*}^*} = \mathbf{B}_h \widehat{\boldsymbol{\theta}}_{h, \lambda_{1,t^*}^*, \lambda_{2,t^*}^*}$$

at the declared hot-spot time t^* and the corresponding parameter λ_{1,t^*}^*, λ_{2,t^*}^* in Equation 2.6. For the numerical computation purpose, it is often easier to directly work with the tensor format of the hot-spot $\widehat{\mathbf{h}}_{\lambda_{1,t^*}^*, \lambda_{2,t^*}^*}$, denoted as $\widehat{\mathcal{H}}_{\lambda_{1,t^*}^*, \lambda_{2,t^*}^*}$, which is a tenor of dimension $n_1 \times n_2 \times T$. If the (i, j, t^*)-th entry in $\widehat{\mathcal{H}}_{\lambda_{1,t^*}^*, \lambda_{2,t^*}^*}$ is non-zero, then we declare that there is a hot-spot for the j-th week in the i-th state in t^*-th year.

2.5 Optimization Algorithm

In this section, we will develop an efficient optimization algorithm for solving the optimization problem in Equation 2.3. For notion convenience, we adjust the notation above a

little bit. Because $\boldsymbol{\theta}_m, \boldsymbol{\theta}_h$ in Equation 2.3 is solved under penalty $\lambda_1 R_1(\boldsymbol{\theta}_h) + \lambda_2 R_2(\boldsymbol{\theta}_h)$, we change $\boldsymbol{\theta}_m$, $\boldsymbol{\theta}_h$ into $\boldsymbol{\theta}_{m,\lambda_1,\lambda_2}, \boldsymbol{\theta}_{h,\lambda_1,\lambda_2}$ to emphasize the penalty parameter λ_1 and λ_2. Accordingly, $\boldsymbol{\theta}_{h,0,\lambda_2}$ refers to the estimator only under the second penalty $\lambda_2 R_2(\boldsymbol{\theta}_h)$, i.e,

$$\boldsymbol{\theta}_{h,0,\lambda_2} = \arg\min_{\boldsymbol{\theta}_m, \boldsymbol{\theta}_h} \left\{ \|\mathbf{e}\|_2^2 + \lambda_2 R_2(\boldsymbol{\theta}_h) \right\}. \tag{2.7}$$

The structure of this section is that, we first develop the procedure of our proposed method in subsection 2.5.1 and then give the computational complexity in subsection 2.5.2.

2.5.1 Procedure of Our Algorithm

In the optimization problem shown in Equation 2.3, there are two unknown vectors, namely $\boldsymbol{\theta}_{m,\lambda_1,\lambda_2}$, $\boldsymbol{\theta}_{h,,\lambda_1,\lambda_2}$. To simplify the optimization above, we first figure out the closed-form correlation between $\boldsymbol{\theta}_{m,\lambda_1,\lambda_2}$ and $\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}$. Then, we solve the optimization by modifying the matrix algebra in the fast iterative shrinkage-thresholding algorithm (FISTA) [see 24] into tensor algebra. The key to realize it is the proximal mapping of $\lambda_1 R_1(\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}) + \lambda_2 R_2(\boldsymbol{\theta}_{h,\lambda_1,\lambda_2})$. To address it, we first aims at the proximal mapping of $\lambda_2 R_2(\boldsymbol{\theta}_{h,0,\lambda_1})$, where SFA via gradient descent [see 29] is used. And then the proximal mapping of $\lambda_1 R_1(\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}) + \lambda_2 R_2(\boldsymbol{\theta}_{h,\lambda_1,\lambda_2})$ can be solved with a closed-form correlation between it and the proximal mapping of $\lambda_2 R_2(\boldsymbol{\theta}_{h,0,\lambda_2})$.

There are three subsections in this section, where each subsection represents one step in our proposed algorithm.

Estimate the mean parameter

To begin with, we first simplify the optimization problem in Equation 2.3, i.e., figure out the closed-form correlation between $\boldsymbol{\theta}_{m,\lambda_1,\lambda_2}$ and $\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}$.

Although there are two sets of parameters $\boldsymbol{\theta}_{m,\lambda_1,\lambda_2}$ and $\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}$ in the model, we note that given $\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}$, the parameter $\boldsymbol{\theta}_{m,\lambda_1,\lambda_2}$ is involved in the standard least squared estima-

tion and thus can be solved in the closed-form solution, see Equation 2.8 in the proposition below.

Proposition 2.5.1. Given $\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}$, the closed-form solution of $\boldsymbol{\theta}_{m,\lambda_1,\lambda_2}$ is given by:

$$\boldsymbol{\theta}_{m,\lambda_1,\lambda_2} = (\mathbf{B}_m^\top \mathbf{B}_m)^{-1}(\mathbf{B}_m^\top \mathbf{y} - \mathbf{B}_m^\top \mathbf{B}_h \boldsymbol{\theta}_{h,\lambda_1,\lambda_2}) \tag{2.8}$$

It remains to investigate how to estimate the parameter $\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}$. After plugging in Equation 2.8 into Equation 2.3, the optimization problem for estimating $\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}$ becomes

$$\arg\min_{\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}} \|\mathbf{y}^* - \mathbf{X}\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}\|_2^2 + \lambda_1 \|\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}\|_1 + \lambda_2 \|\mathbf{D}\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}\|_1, \tag{2.9}$$

where $\mathbf{y}^* = [\mathbf{I} - \mathbf{H}_m]\,\mathbf{y}$, $\mathbf{X} = [\mathbf{I} - \mathbf{H}_m]\,\mathbf{B}_h$ and $\mathbf{H}_m = \mathbf{B}_m(\mathbf{B}_m^\top \mathbf{B}_m)^{-1}\mathbf{B}_m^\top$.

Due to the high dimension, we need to develop an efficient and precise optimization algorithm to optimize Equation 2.9. Obviously, Equation 2.9 is a typical sparse optimization problem. However, most of the sparse optimization frameworks focus on optimizing:

$$\arg\min_{\boldsymbol{\theta}_{h,0,\lambda_2}} \|\mathbf{y}^* - \mathbf{X}\boldsymbol{\theta}_{h,\lambda_1,0}\|_2^2 + \lambda_1 \|\boldsymbol{\theta}_{h,\lambda_1,0}\|_1 , \tag{2.10}$$

for instance [24, 55, 56], where the iterative updating rule is used based either on the gradient information or the proximal mapping. In most cases, the algorithms above works, however, two challenges occur in our paper:

1. When the dimension of $\mathbf{X}$ (of size $n_1 n_2 T \times n_1 n_2 T$) becomes increasingly large, it is difficult for the computer to store and memorize it.

2. When the penalty term is $\lambda_1\|\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}\|_1 + \lambda_2\|\boldsymbol{D}\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}\|_1$, instead of only $\lambda_1\|\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}\|_1$, direct application of the proximal mapping of $\lambda_1\|\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}\|_1$ is not workable.

Therefore, directly applying these above algorithms [see 24, 55, 56] to our case is not feasible. To extend the existing research, we propose an iterative algorithm in algo-

rithm 2 and we explain the approach to solve the proximal mapping of $\lambda_1\|\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}\|_1 + \lambda_2\|\boldsymbol{D}\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}\|_1$ in the next subsection.

Proximal Mapping

The main tool we use to solve the optimization problem in Equation 2.9 is a variation of proximal mapping. Denote that $F(\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}) = \frac{1}{2}\|\mathbf{y}^* - \mathbf{X}\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}\|_2^2$. And in the i-th iteration, the according recursive estimator of $\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}$ is noted as $\boldsymbol{\theta}^{(i)}_{h,\lambda_1,\lambda_2}$. Besides,an auxiliary variable $\boldsymbol{\eta}^{(i)}$ is introduced to update from $\boldsymbol{\theta}^{(i)}_{h,\lambda_1,\lambda_2}$ to $\boldsymbol{\theta}^{(i+1)}_{h,\lambda_1,\lambda_2}$ through

$$
\begin{aligned}
\boldsymbol{\theta}^{(i+1)}_{h,\lambda_1,\lambda_2} &= \arg\min_{\boldsymbol{\theta}} F(\boldsymbol{\eta}^{(i)}) + \frac{\partial}{\partial \boldsymbol{\theta}_{h,\lambda_1,\lambda_2}} F(\boldsymbol{\eta}^{(i)}) \left(\boldsymbol{\theta} - \boldsymbol{\eta}^{(i)}\right) + \\
&\quad \lambda_1\|\boldsymbol{\theta}\|_1 + \lambda_2\|\mathbf{D}\boldsymbol{\theta}\|_1 + \frac{L}{2}\|\boldsymbol{\theta} - \boldsymbol{\eta}^{(i)}\|_2^2 \\
&= \arg\min_{\boldsymbol{\theta}} \left[\frac{1}{2}\left[\boldsymbol{\theta} - \left(\boldsymbol{\eta}^{(i)} - \frac{\partial}{L\partial\boldsymbol{\theta}} F(\boldsymbol{\eta}^{(i)})\right)\right]^2 + \lambda_1\|\boldsymbol{\theta}\|_1 + \lambda_2\|\mathbf{D}\boldsymbol{\theta}\|_1\right] \\
&\triangleq \pi^{\lambda_1}_{\lambda_2}(\mathbf{v})
\end{aligned}
$$

where $\mathbf{v} = \boldsymbol{\eta}^{(i)} - \frac{\partial}{L\partial\boldsymbol{\theta}} F(\boldsymbol{\eta}^{(i)})$, $\boldsymbol{\eta}^{(i)} = \boldsymbol{\theta}^{(i)}_{h,\lambda_1,\lambda_2} + \frac{t_{i-2}-1}{t_{i-1}}(\boldsymbol{\theta}^{(i)}_{h,\lambda_1,\lambda_2} - \boldsymbol{\theta}^{(i-1)}_{h,\lambda_1,\lambda_2})$ and $t_{-1} = t_0 = 1$, $t_{i+1} = \frac{1+\sqrt{1+4t_i^2}}{2}$

Because it is difficult to solve $\pi^{\lambda_1}_{\lambda_2}(\mathbf{v})$ directly, we aim to solve $\pi^{0}_{\lambda_2}(\mathbf{v})$ first. And proved by [29], there is a closed-form correlation between $\pi^{\lambda_1}_{\lambda_2}(\mathbf{v})$ and $\pi^{0}_{\lambda_2}(\mathbf{v})$, which is shown in Proposition 2.5.2.

Proposition 2.5.2. The closed-form relationship between $\pi^{\lambda_1}_{\lambda_2}(\mathbf{v})$ and $\pi^{0}_{\lambda_2}(\mathbf{v})$ is

$$
\pi^{\lambda_1}_{\lambda_2}(\mathbf{v}) = \text{sign}(\pi^{0}_{\lambda_2}(\mathbf{v})) \odot \max\{|\pi^{0}_{\lambda_2}(\mathbf{v})| - \lambda_1, 0\}. \tag{2.11}
$$

where $\odot$ is an element-wise product operator.

With the proximal mapping function in Proposition 2.5.2, we can now develop the algorithm shown in algorithm 2.

Algorithm 2: Iterative updating based on tensor decomposition

Input: $\mathbf{y}^*, \mathbf{B}_s, \mathbf{B}_w, \mathbf{B}_y, \mathbf{D}_s, \mathbf{D}_w, \mathbf{D}_y, K, L, \lambda_1, \lambda_2, L_0, M_1, M_2$

Output: $\boldsymbol{\theta}_{h,\lambda_1,\lambda_2}$

initialization;

$\boldsymbol{\Theta}^{(1)} = \boldsymbol{\Theta}^{(0)}, t_{-1} = 1, t_0 = 1, L = L_0$

for $i = 1 \cdots M_1$ **do**

$\mathcal{N}^{(i)} = \mathcal{N}^{(i)} + \frac{t_{i-2}-1}{t_{i-1}}(\boldsymbol{\Theta}^{(i)} - \boldsymbol{\Theta}^{(i-1)})$

$$\mathcal{V} = \mathcal{N}^{(i)} - \frac{1}{L}\mathcal{N}^{(i)} \times_1 (\mathbf{P}_s^\top \mathbf{P}_s) \times_2 (\mathbf{P}_w^\top \mathbf{P}_w) \times_3 (\mathbf{P}_y^\top \mathbf{P}_y) - \frac{1}{L}\mathcal{Y}^* \times_1 \mathbf{P}_s^\top \times_2 \mathbf{P}_w^\top \times_3 \mathbf{P}_y^\top$$

for $j = 0 \cdots M_2$ **do**

$$\mathcal{G}^{(i)} = \left(\mathcal{Z}^{(j)} \times_1 (\mathbf{D}_s^\top \mathbf{D}_s) \times_2 (\mathbf{D}_w^\top \mathbf{D}_w) \times_3 (\mathbf{D}_y^\top \mathbf{D}_y))\right) - (\mathcal{V} \times_1 \mathbf{D}_s \times_2 \mathbf{D}_w \times_3 \mathbf{D}_y)$$

$\mathcal{Z}^{(j+1)} = P\left(\mathcal{Z}^{(j)} - \mathcal{G}^{(j)}/L\right)$

$\pi^0_{\lambda_2}(\mathcal{V}) = \mathcal{V} - (\mathcal{Z}^{(M_2)}) \times_1 \mathbf{D}_s \times_2 \mathbf{D}_w \times_3 \mathbf{D}_y$

$\pi^{\lambda_1}_{\lambda_2}(\mathcal{V}) = \text{sign}(\pi^0_{\lambda_2}(\mathcal{V})) \odot \max\{\left|\pi^0_{\lambda_2}(\mathcal{V})\right| - \lambda_1, 0\}$

$t_{i+1} = \frac{1+\sqrt{1+4t_i^2}}{2}$

$\widehat{\boldsymbol{\Theta}}_{h,\lambda_1,\lambda_2} = \pi^{\lambda_1}_{\lambda_2}(\mathcal{V})$

$\widehat{\boldsymbol{\theta}}_{h,\lambda_1,\lambda_2} = \text{vector}(\widehat{\boldsymbol{\Theta}}_{h,\lambda_1,\lambda_2})\ \boldsymbol{v} = \text{vector}(\mathcal{V})$

vector$(\cdot)$ is a function that unfolding a order-3 tensor of dimension $n_1 \times n_2 \times T$ into a vector of length$n_1 n_2 T$.

2.5.2 Computational Complexity

This section discusses the computational complexity of our proposed algorithm. The computation complexity of our propose method is of order $O\left(n_1 n_2 T \max\{n_1, n_2, T\}\right)$ (see Proposition 2.5.3).

Proposition 2.5.3. The computational complexity of our proposed algorithm (see algorithm 2) is of order $O\left(n_1 n_2 T \max\{n_1, n_2, T\}\right)$.

The proof of the above proposition can be found in subsection 2.8.1.

2.6 Simulation

In this section, we conduct simulation studies to evaluate our proposed methodologies by comparing it with several benchmark methods in the literature. The structure of this section is described as follows. We first present the data generation mechanism for our simulations in subsection 2.6.1, then discuss the performance of hot-spot detection and localization in subsection 2.6.2.

2.6.1 Generative Model in Simulation

In our simulation, at each time index $t(t = 1 \cdots T)$, we generate a vector $\mathbf{y}_t$ of length $n_1 n_2$ by

$$\mathbf{y}_{i,t} = (\mathbf{B}\boldsymbol{\theta}_t)_i + \delta \mathbb{1}\{t \geq \tau\} \mathbb{1}\{i \in S_h\} + \mathbf{w}_{i,t}. \tag{2.12}$$

To match the dimension in the case study, we choose $n_1 = 50, n_2 = 51$. We use $\mathbf{y}_{i,t}$ to denote the i-th entry in the vector $\mathbf{y}_t$, and $(\mathbf{B}\boldsymbol{\theta}_t)_i$ denotes the i-th entry in the vector $\mathbf{B}\boldsymbol{\theta}_t$, where the matrix $\mathbf{B} = \mathbf{B}_{m,s} \otimes \mathbf{B}_{m,w} \otimes \mathbf{B}_{m,y}$ is calculated with the same choice as that in subsection 2.3.2. The scalar δ measures the change magnitude. We further consider two sub-cases, depending on the value of change magnitude δ in Equation 2.12: one is $\delta = 0.1$ (small shift) and the other is $\delta = 0.5$ (large shift). For the anomaly setup, $\mathbb{1}(A)$

is the indicator function, which has the value 1 for all elements of A and the value 0 for all elements not in A. Accordingly, $\mathbb{1}\{t \geq \tau\}$ indicates that the spatial hot-spots only occur after the temporal hot-spot τ. This ensures that the simulated hot-spot is temporal consistent. Here we assume the change happens at $\tau = 50$ among total $T = 100$ years. The second indicator function $\mathbb{1}\{i \in S_h\}$ shows that only those entries whose location index belongs set S_h are assigned as local hot-spots. This ensures that the simulated hot-spot is sparse. Here we assume that the spatial hot-spots index set is formed by the combination of states Conn, Ohio, West Va, Tex, Hawaii, and the week from 1-10 and 41-51. And $\mathbf{w}_{i,t}$ is the i-th entry in the white noise vector whose entries are independent and follow $N(0, 0.1^2)$ distribution.

2.6.2 Hot-spot Detection Performance

In this section, we compare the performance of our proposed method (denoted as 'SSR-tensor') for the detection of hot-spot with some benchmark methods. Specifically, we compare our proposed method with Hotelling T^2 control chart [see 36] (denoted as 'T2'), Lasso-based control chart proposed by [39] (denoted as 'ZQ Lasso'), PCA-based control chart proposed by [17] (denoted as 'PCA') and SSD proposed by [12] (denoted as 'SSD'). Note that there are two main differences between our SSR-tensor method and the SSD method in [12]. First, SSR-Tensor has the autoregressive or fussed Lasso penalty in Equation 2.3 to ensure the temporal continuity of the hot-spot. Second, SSD uses the Shewhart control chart to monitor temporal changes, while SSR-Tensor utilizes CUSUM instead, which is more sensitive for a small shift.

For the basis choices of our proposed method, to model the spatial structure of the global trend, we choose $\mathbf{B}_{m,1}$ as the kernel matrix to describe the smoothness of the background, whose (i, j) entry is of value $\exp\{-d^2/(2c^2)\}$ where d is the distance between the i-th state and j-th state and c is the bandwidth chosen by cross-validation. In addition, we choose identical matrices for the yearly basis and weekly basis since we do not have any

prior information. Moreover, we use the identity matrix for the spatial and temporal basis of the hot-spots. For SSD in [12], we will use the same spatial and temporal basis in order to have a fair comparison.

For evaluation, we will compute the following four criteria: (i) precision, defined as the proportion of detected anomalies that are true hot-spots; (ii) recall, defined as the proportion of the anomalies that are correctly identified; (iii) F measure, a single criterion that combines the precision and recall by calculating their harmonic mean; and (iv) the corresponding average run length (ARL_1), a measure of the average detection delay in the special scenario when the change occurs at time $t = 1$. All simulation results below are based on 1000 Monte Carlo simulation replications.

Table 2.1 shows the merits of our methodology mainly lies on the higher precision and shorter ARL_1. For example, when the shift is very small, i.e., $\delta = 0.1$, the ARL_1 of our SSR-Tensor method is only 1.6420 compared with 7.4970 of SSD and 9.5890 of ZQ-Lasso. The reason for SSR-Tensor has shorter ARL_1 than that of SSD is that SSD uses the Shewhart control chart to detect temporal changes, which makes it insensitive for a small shift. While for SSR-Tensor, it applies the CUSUM control chart, which is capable to detect the shift of small size. The reason for both SSR-Tensor and SSD have shorter ARL_1 than that of ZQ-Lasso, PCA, and T2 is that ZQ-Lasso fails to capture the global trend mean. Yet, the data generated in our simulation has both decreasing and circular global trend, which makes it hard for ZQ-Lasso to model well.

2.7 Case Study

In this section, we apply our proposed SSR-tensor model and hot-spot detection/localization method to the weekly gonorrhea dataset in section 2.2. For the purpose of comparison, we also consider other benchmark methods mentioned in section 2.6, and consider two performance criteria: one is the temporal detection of hot-spots (i.e., which year it occurs) and the other is the localization of the hot-spots (i.e., which state and which week might involve

Table 2.1: Scenario 1 (decreasing global trend): Comparison of hot-spot detection under small shift and large shift

methods	small shift $\delta = 0.1$				large shift $\delta = 0.5$			
	precision	recall	F measure	ARL_1	precision	recall	F measure	ARL_1
SSR-tensor	**0.0824**	**0.9609**	**0.5217**	**1.6420**	**0.0822**	**0.9633**	**0.5228**	**1.0002**
	(0.0025)	(0.0536)	(0.0270)	(0.7214)	(0.0022)	(0.0549)	(0.0277)	(0.0144)
SSD	0.0404	0.9820	0.5112	7.4970	0.0412	1.0000	0.5206	1.0000
	(0.0055)	(0.1330)	(0.0692)	(9.4839)	(0.0000)	(0.0000)	(0.0000)	(0.0000)
ZQ Lasso	0.0412	1.000	0.5206	9.5890	0.0412	1.0000	0.5206	8.8562
	(0.0000)	(0.0000)	(0.0000)	(7.5414)	(0.0000)	(0.0000)	(0.0000)	(7.1169)
PCA	-	-	-	28.7060	-	-	-	32.0469
	-	-	-	(16.9222)	-	-	-	(17.4660)
T2	-	-	-	50.0000	-	-	-	50.0000
	-	-	-	(0.0000)	-	-	-	(0.0000)

the alarm).

2.7.1 When the temporal changes happen?

Here we consider the performance on the temporal detection of hot-spots of our proposed method and other benchmark methods. For our proposed SSR-Tensor method, we build a CUSUM control chart utilizing the test statistic in subsection 2.4.1, which is shown in Figure 2.4. From this plot, we can see that the hot-spots are detected at 10-th year, i.e., 2016.

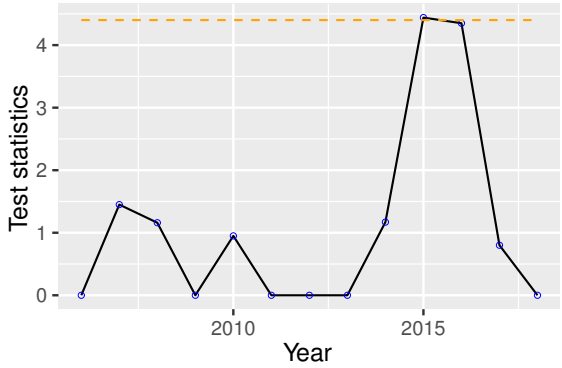


Figure 2.4: CUSUM Control chart of gonorrhea dataset during years 2006-2018.

For the purpose of comparison, we also apply the benchmark methods, SSD [see 12], ZQ Lasso [see 39], PCA [see 17] and T2 [see 36], into the gonorrhea dataset. Unfortunately, all benchmark methods are unable to raise any alarms, but our proposed SSR-tensor method

raises the first hot-spot alarm in the year 2016.

2.7.2 In Which State and Week Do the Spatial Hot-spots Occur?

After the temporal detection of hot-spots, we need to further localize the hot-spots in the sense that we need to find out which state and which week may lead to the occurrence of the temporal hot-spot. Because the baseline methods, ZQ-Lasso, PCA, and T2, can only realize the detection of temporal changes, and SSD fails to detect the temporal changes, we only show the localization of spatial hot-spot by SSR-Tensor, which is visualized in Figure 2.5.

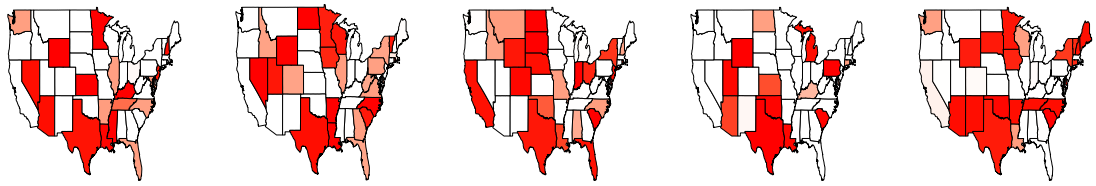


Figure 2.5: Hot-spot detection result of circular pattern of W.S. CENTRAL(Arkansas, Louisiana, Oklahoma, Texas)

There are some circular patterns in specific areas. For example, CENTRAL(Ark, La, Okla, Tex) tends to have a circular pattern every 11 weeks, which is shown in Figure 2.5. Besides, there is also some circular pattern for a certain state, for instance, Kansas has the bi-weekly pattern as shown in Figure 2.6. To validate the bi-weekly circular pattern of Kansas, we plot the time series plot of Kansas in 2016 as well as the auto-correlation function plot in Figure 2.5. Besides, the auto-correlation function plot in the left panel of Figure 2.6 serves as a baseline. It can be seen from the middle and right plot of Figure 2.6 that, Kansas has some bi-weekly or tri-weekly circular pattern.

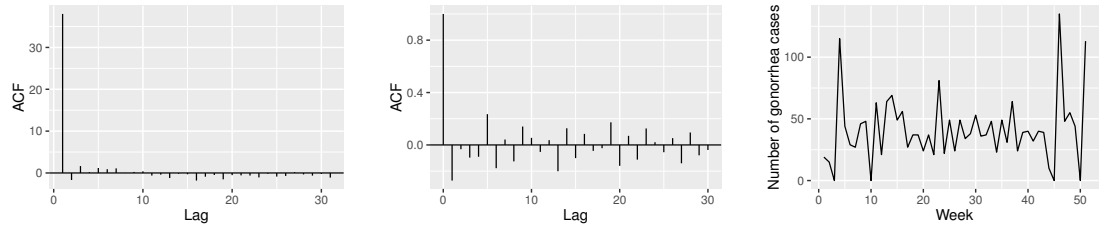


Figure 2.6: Auto-correlation of all US (left) & Kans.(middle) in 2016 and time series plot of Kansas in 2016 (right)

2.8 Supplementary Material

2.8.1 Proof of Proposition 2.5.3

Proof. The main computational load in algorithm 2 is on the calculation of $\mathbf{v}$ (line 4), $\mathbf{g}^{(i)}$(line 5) and $\pi^0_{\lambda_2}(\mathbf{v})$ (line 7). We will take the calculation of $\mathbf{v}$ in line 4 in the algorithm as an example. To begin with, we focus on the computational complexity of

$$\mathcal{N}^{(i)} \times_1 (\mathbf{P}_s^\top \mathbf{P}_s) \times_2 (\mathbf{P}_w^\top \mathbf{P}_w) \times_3 (\mathbf{P}_y^\top \mathbf{P}_y)). \tag{2.13}$$

For better illustration, we denote tensor$(\boldsymbol{\eta}^{(i)})$ as $\mathcal{N}^{(i)}$ and $\mathcal{N}^{(i)} \times_1 (\mathbf{P}_s^\top \mathbf{P}_s)$ as tensor $\mathcal{L}_1$. According to the tensor algebra [see 54, Section 2.5],

$$\mathcal{L}_1 = \mathcal{N}^{(i)} \times_1 (\mathbf{P}_s^\top \mathbf{P}_s) \Longleftrightarrow \mathcal{L}_{1(1)} = \mathbf{P}_s^\top \mathbf{P}_s \mathcal{N}^{(i)}_{(1)}.$$

Therefore, the computational complexity of Equation 2.13 is the same as two-matrix multiplication with order $n_1 \times n_1$ and $n_1 \times n_1 n_2$, which is of order $O\left(n_1 n_2 T(2n_1 - 1)\right)$.

After the calculation of $\mathcal{L}_1$, Equation 2.13 is reduced to

$$\mathcal{L}_1 \times_2 (\mathbf{P}_w^\top \mathbf{P}_w) \times_3 (\mathbf{P}_y^\top \mathbf{P}_y)). \tag{2.14}$$

Similarly, denotes $\mathcal{L}_2 = \mathcal{L}_1 \times_2 (\mathbf{P}_w^\top \mathbf{P}_w)$, then

$$\mathcal{L}_2 = \mathcal{L}_1 \times_2 (\mathbf{P}_w^\top \mathbf{P}_w) \Longleftrightarrow \mathcal{L}_{2(2)} = \mathbf{P}_w^\top \mathbf{P}_w \mathcal{N}_{(2)}.$$

Therefore, the computational complexity of Equation 2.14 is the same as two-matrix multiplication with order $n_2 \times n_2$ and $n_2 \times n_1 T$, which is of order $O\left(n_1 n_2 T(2n_2 - 1)\right)$.

After the calculation of $\mathcal{L}_2$,Equation 2.14 is reduced to

$$\mathcal{L}_2 \times_3 (\mathbf{P}_y^\top \mathbf{P}_y)). \tag{2.15}$$

Similarly, denotes $\mathcal{L}_3 = \mathcal{L}_2 \times_2 (\mathbf{P}_y^\top \mathbf{P}_y)$, then

$$\mathcal{L}_3 = \mathcal{L}_2 \times_3 (\mathbf{P}_y^\top \mathbf{P}_y) \Longleftrightarrow \mathcal{L}_{3(3)} = \mathbf{P}_w^\top \mathbf{P}_w \mathcal{N}_{(3)}.$$

Therefore, the computational complexity of Equation 2.14 is the same as two-matrix multiplication with order $T \times T$ and $T \times n_1 n_2$, which is of order $O\left(n_1 n_2 T(2T - 1)\right)$.

By combining all these blocks built above, we conclude that the computational complexity of Equation 2.13 is of order $O(n_1 n_2 T\left(\max\{n_1, n_2, T\}\right))$.

In the same way, the computational complexity in line 5 and 7 of algorithm 2 is also of order $O(n_1 n_2 T\left(\max\{n_1, n_2, T\}\right))$. Thus, the computational complexity of Algorithm is of order $O(n_1 n_2 T\left(\max\{n_1, n_2, T\}\right))$. □

CHAPTER 3

IDENTIFICATION OF UNDERLYING DYNAMIC SYSTEM FROM NOISY DATA WITH SPLINES

3.1 Introduction

In practice, one is often encountered with noisy data coming from an unknown partial differential equation (PDE):

$$\begin{aligned}\mathcal{D} \ = \{(x_i, t_n, u_i^n) : \ & x_i \in (0, X_{\max}) \subseteq \mathbb{R}, \ \forall \ i = 0, \ldots, M-1, \\ & t_n \in (0, T_{\max}) \subseteq \mathbb{R}, \ \forall \ n = 0, \ldots, N-1\} \in \Omega.\end{aligned} \tag{3.1}$$

Here $t_n \in \mathbb{R}$ is the temporal variable with $t_n \in (0, T_{\max})$ for $n = 0, 1, \ldots, N-1$, and we call N the *temporal resolution*. And $x_i \in \mathbb{R}$ is the spatial variable with $x_i \in (0, X_{\max})$ for $i = 0, 1, \ldots, M-1$, and we call M the *spatial resolution*. We use $T_{\max}, X_{\max}$ to denote the upper bound of the temporal variable and spatial variable, respectively.

In the above dataset $\mathcal{D}$, the variable u_i^n is a representation of ground truth $u(x_i, t_n)$ contaminated by noise following normal distribution with zero mean and stand deviation σ:

$$u_i^n = u(x_i, t_n) + \epsilon_i^n \qquad \epsilon_i^n \overset{i.i.d}{\sim} N(0, \sigma^2), \tag{3.2}$$

where $u(x, t)$ is the underlying PDE model from where $\mathcal{D}$ is generated.

For this type of noisy dataset $\mathcal{D}$, we are typically interested in identifying its underlying PDE model:

$$\begin{aligned}\frac{\partial}{\partial t}u(x,t) \ = \ & \beta_{00}^* + \sum_{k=0}^{q_{\max}} \sum_{i=1}^{p_{\max}} \beta_{k^i}^* \left[\frac{\partial^k}{\partial^k x} u(x,t)\right]^i + \\ & \sum_{\substack{i+j \le p_{\max} \\ i,j>0}} \ \sum_{\substack{0<k<l \\ l \le q_{\max}}} \beta_{k^i,l^j}^* \left[\frac{\partial^k}{\partial^k x} u(x,t)\right]^i \left[\frac{\partial^l}{\partial^l x} u(x,t)\right]^j,\end{aligned} \tag{3.3}$$

where the left-hand side of the above equation is the partial derivative with respect to the temporal variable t, while the right side hand is the $p_{\max}$th order polynomial of the derivatives with respect to the spatial variable x up to the $q_{\max}$th order. For notation simplifications, we denote the ground truth coefficient vector $\boldsymbol{\beta}^* = (\beta^*_{00}, \beta^*_{0^1}, \beta^*_{1^1}, \ldots, \beta^*_{q^{p_{\max}}_{\max}})$ as $\boldsymbol{\beta}^* = (\beta^*_1, \beta^*_2, \beta^*_3, \ldots, \beta^*_K)^\top$ where $K = 1 + (p_{\max}+1)q_{\max} + \frac{1}{2}q_{\max}(q_{\max}+1)(p_{\max}-1)!$. It should be noted that, in practice, the majority of the entries in $\boldsymbol{\beta}^*$ are zero. For instance, in the transport equation $\frac{\partial}{\partial t}u(x,t) = a\frac{\partial}{\partial x}u(x,t)$ with any $a \neq 0$, we only have $\beta^*_3 \neq 0$ and $\beta^*_i = 0$ for any $i \neq 3$ [see 57, Section 2.2]. So we know the coefficient $\boldsymbol{\beta}^*$ in Equation 3.3 is sparse.

To identify the above model, one needs to overcome two technical issues. First, derivatives are unobservable from $\mathcal{D}$ and have to be estimated from noisy observations of the values of the function. Second, there could be lots of data-driven PDE models that suit the noisy data very well. Among all these models, a simple model would be desirable. However, it is not clear how can we identify the simple model.

In this paper, we propose a two-stage method – *Spline Assisted Partial Differential Equation involved Model Identification (SAPDEMI)* – to efficiently identify the underlying PDE models from the noisy data $\mathcal{D}$. The first stage is called *functional estimation stage*, where we estimate all the derivatives from the noisy data $\mathcal{D}$, including $\frac{\partial}{\partial t}u(x,t)$, $\frac{\partial}{\partial x}u(x,t)$ and so on. In this stage, the main tool we use is the cubic spline, where we first use the cubic spline to fit the noisy data, and then we approximate the derivatives of the true dynamic models as the derivatives of the cubic splines. The second stage is called *model identification stage*, where we identify the underlying PDE models from the noisy data $\mathcal{D}$. In this stage, we apply the Least Absolute Shrinkage and Selection Operator (Lasso) [see 7] to identify the derivatives (or their combinations) that are included in the underlying models. To ensure the correctness of the identification, we develop sufficient conditions for correct identification and the asymptotic properties of the identified models, where the main tool we use is the primal-dual witness (PDW) method [see 58, Chapter 11].

The structure of the rest of this section is described as follows. In subsection 3.1.1, we survey the existing methods to solve the above PDE identification problem. In subsection 3.1.2, we articulate our contributions of this paper.

3.1.1 Literature Review

The pioneering representative work to identify the underlying dynamic models from the noisy data is [59]. This method is also a two-stage method, where in the functional estimation stage, [59] use the local polynomial regression to estimate the value of the function and its derivatives. Then, in the model identification stage, [59] use the least squares model. Following this pioneering work, other researchers conduct various extensions.

The first type of extension is to modify the function estimation stage of [59], and we classify the existing extensions into two categories: (1) the numerical differentiation, and (2) the basis expansion.

In the numerical differentiation category [see 60, 61, 62], the derivative $\frac{\partial}{\partial x}u(x,t)$ is naively approximated as

$$\frac{\partial}{\partial x}u(x,t) \approx \frac{u(x+\Delta x,t)-u(x-\Delta x,t)}{2\Delta x},$$

where $(x+\Delta x,t),(x-\Delta x,t)$ are the two closest points of (x,t) in the x-domain. The essence of numerical differentiation is to approximate the first-order derivative as the slope of a nearby secant line. Although the implementation of numerical differentiation is very easy, it could be highly biased because its accuracy is highly dependent on the value of Δx. On the one hand, if Δx is too small, the subtraction will yield a large rounding error [see 63, 64]. In fact, all the finite-difference formulae are ill-conditioned [see 65] and due to cancellation will produce a value of zero if Δx is small enough [see 66]. On the other hand, if Δx is too large, though the calculation of the slope of the secant line will be more accurately calculated, the estimation of the slope of the tangent by using the secant line

could be poor. However, in our case, the size of Δx is decided by the noisy data $\mathcal{D}$, which could be very small or very large. So if we naively use numerical differentiation to estimate the derivatives from the noisy data $\mathcal{D}$, it could be highly possible that we will get biased estimations.

In the basis expansion category, researchers first approximate the unknown dynamic curves by basis expansion and then approximate the derivatives of underlying dynamic curves as the derivatives of the approximation curves. As for the choice of basis in the basis expansion, there are multiple choices in the existing literature. The most popular basis is the polynomial basis, which is already used by [59]. Other examples of polynomial basis can be found in [67, 68, 69, 70, 71]. One popular choice of basis is spline basis [see 72, 73, 60, 74, 75]. The major limitation of the above method is that it evolves with high computational complexity. For instance, the local polynomial basis requires computational complexity of order $\max\{O(M^2N), O(MN^2)\}$ in the functional estimation stage. However, our proposed SAPDEMI method only requires computational complexity of order $O(MN)$. This is the lowest possible bound in theory in the functional estimation stage because it is the complexity of reading in the data set $\mathcal{D}$.

The second type of extension is to modify the model identification stage of [59]. The existing methods fall in the framework of the (penalized) least squares method, and we mainly divide them into three categories: (1) the least squares method, (2) the ℓ_2-penalized least squares method, and (3) the ℓ_1-penalized least squares method.

In the least squares method category, [76] use the least squares method to estimate the parameters in unknown ordinary differential equation (ODE) models. [67] and [60] use the least squares to estimate the parameters in unknown PDE models. The major limitation of this method is that it can lead to overfitting models.

In the ℓ_2-penalized least squares method category, [74, 77] and [75] penalize the smoothness of the unknown PDE models, which shares similar ideas with the reproducing kernel Hilbert space (RKHS). And essentially speaking, this method falls in the framework of the

ℓ_2-penalized least squares method. Although this method can avoid overfitting by introducing the ℓ_2-penalty, it has limited power to do "model selection" instead of "parameter estimation". Specifically, they assume that the form of the dynamic models (either ODE or PDE) is known, and their goal is to estimate the coefficients in the given model. However, in real practice, the form of the dynamic model is potentially unknown. Under this scenario, instead of "parameter estimation", we also need to do "model selection".

In the ℓ_1-penalized least squares method category, [68] identifies the unknown dynamic models through the ℓ_1-penalized least squares method, and later the author discusses the design of an efficient algorithm with proximal mapping method. But the authors do not discuss the asymptotic statistical property of the identified model. Recently, [78] utilize the similar method as [68] to identify the unknown dynamic models. Although [78] demonstrated some empirical successes, the rigorous theoretical justification still remains vague. So this category is not fully explored in terms of the statistical property.

During the investigation of the literature of ODE/PDE identification, we also find some research work published outside the statistical journals. These researchers investigate the importance of the ODE/PDE identification problem, but they do not develop a statistical theory on their methods, which is what we do in this paper. For instance, [79] assume that the derivatives are already known in the functional estimation stage and then they utilize the Akaike information criterion (AIC) to realize the model selection in the model identification stage. But the theoretical property of the selected model is not analyzed. Another example is [69], wherein the model identification stage, the authors first use the ℓ_2-penalized least squares method to estimate the parameter $\boldsymbol{\beta}$, and then manually shrinkage the small coefficients to zero by applying the hard-threshold method to realize the model selection. Other similar papers include [61, 62, 80, 81]. The methods in the above papers may work in special cases, but their statistical properties are not established, and the rigorous proofs remain an open question in these papers.

3.1.2 Our Contribution

In this section, we discuss the contributions of our proposed SAPDEMI method.

First, our proposed SAPDEMI method is computationally efficient in the functional estimation stage. Specifically, we only require computationally complexity of order $O(MN)$, which is the lowest possible order in this stage. And the popularly used local polynomial regression requires computational complexity of order $\max\{O(M^2N), O(MN^2)\}$, which is more computationally expensive than our proposed SAPDEMI method.

Second, our proposed SAPDEMI method realizes "model selection", instead of to "parameter estimations" in the model identification stage. The existing methods, for instance [67, 76, 60, 74, 77, 75], can only realize "parameter estimation", where they always assume that the form of the underlying PDE models is known. However, our proposed SAPDEMI method can identify the underlying PDE models without knowing the form of the underlying PDE models.

Finally, we develop sufficient conditions for correct identification, and we also establish the statistical properties of our identified models, which has not been seen in the literature.

The remaining of the paper is organized as follows. In section 3.2, we develop the technical details of our proposed SAPDEMI method. In section 3.3, we present our main theory, including the sufficient conditions for correct identification, and the statistical properties of our identified models. In section 3.4, we conduct numerical experiments to validate the main theory in section 3.3. In section 3.5, we summarize this paper and discuss the future research.

3.2 Proposed Method: SAPDEMI

In this section, we develop an efficient statistical method called SAPDEMI to identify the underlying PDE model from noisy data $\mathcal{D}$. Our proposed SAPDEMI method is a two-stage method to identify the unknown PDE models. The first stage is called *functional estimation*

stage, where we estimate the function values and their derivatives from the noisy data $\mathcal{D}$ in Equation 3.1. These functional values and their derivatives serve as the input values in the second stage. The second stage is called *model identification stage*, where we identify the underlying PDE model.

For the notations throughout the paper, scalars are denoted by lowercase letters (e.g., β). Vectors are denoted by lowercase bold face letters (e.g., $\boldsymbol{\beta}$), and its ith entry is denoted as β_i. Matrices are denoted by uppercase boldface letter (e.g., $\mathbf{B}$), and its (i,j)th entry is denoted as B_{ij}. For the vector $\boldsymbol{\beta} \in \mathbb{R}^p$, its kth norm is defined as $\|\boldsymbol{\beta}\|_k := \left(\sum_{i=1}^p |\beta_i|^k\right)^{1/k}$. For the matrix $\mathbf{B} \in \mathbb{R}^{m\times n}$, its Frobenius norm is defined as $\|\mathbf{B}\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n |B_{ij}|^2}$, and its p,qth norm is defined as $\|\mathbf{B}\|_{p,q} = \max_{\mathbf{x}\neq\mathbf{0}} \frac{\|\mathbf{B}\mathbf{x}\|_q}{\|\mathbf{x}\|_p}$. We write $f(n) = O(g(n))$, if there exists a positive real number G and a real number n_0 such that $|f(n)| \leq Gg(x)$ for all $n > n_0$.

The structure of this section is described as follows. In subsection 3.2.1, we introduce the function estimation stage. In subsection 3.2.2, we describe the model identification stage.

3.2.1 Functional Estimation Stage

In this section, we discuss the functional estimation stage of our proposed SAPDEMI method, i.e., estimating the functional values and their derivatives from the noisy data $\mathcal{D}$ in Equation 3.1. These derivatives include the derivatives with respect to the spatial variable x and the derivatives with respect to the temporal variable t. In this section, we will take the derivatives with respect to spatial variable x as an example, and the derivatives with respect to the temporal variable t can be derived similarly.

The main tool we use is the cubic spline. Suppose there is a cubic spline $s(x)$ over the knots $\{(x_i, u_i^n)\}_{i=0,1,\ldots,M-1}$ satisfying the following properties [see 82]:

1. $s(x) \in C^2[x_0, x_{M-1}]$, where $C^2[x_0, x_{M-1}]$ denotes the sets of function whose 0th, first and second derivatives are continuous in the domain $[x_0, x_{M-1}]$ with the as-

sumption that $x_0 < x_1 < \ldots < x_{M-1}$;

2. For any $i = 1, \ldots, M-1$, $s(x)$ is a polynomial of degree 3 on the subinterval $[x_{i-1}, x_i]$;

3. For the two end-point x_0, x_{M-1}, we have $s''(x_0) = s''(x_{M-1}) = 0$, where $s''(x)$ is the second derivative of $s(x)$.

By fitting data $\{(x_i, u_i^n)\}_{i=0,1,\ldots,M-1}$ (with a general fixed $n \in \{0, 1, \ldots, N-1\}$) into the above cubic spline $s(x)$, one can solve $s(x)$ as the minimizer of the following optimization problem:

$$J_\alpha(s) = \alpha \sum_{i=0}^{M-1} w_i [u_i^n - s(x_i)]^2 + (1-\alpha) \int_{x_0}^{x_{M-1}} s''(x)^2 dx, \tag{3.4}$$

where the first term $\alpha \sum_{i=0}^{M-1} w_i [u_i^n - s(x_i)]^2$ is the weighted sum of squared residuals, and we take the weight $w_0 = w_1 = \ldots = w_{M-1} = 1$ in our paper. In the second term $(1-\alpha) \int_{x_0}^{x_{M-1}} s''(x)^2 dx$, $s''(x)$ is the second derivative of $s(x)$, and this term is the penalty of smoothness. In the above optimization problem, the parameter $\alpha \in (0, 1]$ trades off the goodness of fit and the smoothness of the cubic spline. By minimizing the above optimization problem with respect to $s(x)$, we can get the estimate of $s(x)$, its first derivative $s'(x)$ and its second derivative $s''(x)$. If the cubic spline approximates the underlying PDE curves very well, then we could declare that the derivatives of the underlying dynamic system can be approximated by the derivatives of the cubic spline $s(x)$, i.e., we have $\widehat{u(x,t_n)} \approx \widehat{s(x)}$, $\frac{\partial}{\partial x}\widehat{u(x,t_n)} \approx \widehat{s'(x)}$, $\frac{\partial^2}{\partial x^2}\widehat{u(x,t_n)} \approx \widehat{s''(x)}$ [see 83, 84, 85].

For the above optimization problem, there is a closed-form solution, which is summarized as follows. First of all, the value of cubic spline $s(x)$ at the point $\{x_0, x_1, \ldots, x_{M-1}\}$, i.e., $\widehat{\mathbf{s}} = \left(\widehat{s(x_0)}, \widehat{s(x_1)}, \ldots, \widehat{s(x_{M-1})}\right)^\top$, can be solved as

$$\widehat{\mathbf{s}} = [\alpha \mathbf{W} + (1-\alpha)\mathbf{A}^\top \mathbf{M} \mathbf{A}]^{-1} \alpha \mathbf{W} \mathbf{u}_\cdot^n, \tag{3.5}$$

which can be used to approximate the 0th order derivative of the underlying PDE models,

i.e., $\widehat{\mathbf{s}} \approx \widehat{\mathbf{f}} = \left(\widehat{u(x_0, t_n)}, \widehat{u(x_1, t_n)}, \ldots, \widehat{u(x_{M-1}, t_n)}\right)^\top$. Here the matrix $\mathbf{W} = \text{diag}(w_0, w_1, \ldots, w_{M-1}) \in \mathbb{R}^{M\times M}$, the vector $\mathbf{u}_{\cdot}^n = \left(u_0^n, \ldots, u_{M-1}^n\right)^\top \in \mathbb{R}^M$, and the matrix $\mathbf{A} \in \mathbb{R}^{(M-2)\times M}, \mathbf{M} \in \mathbb{R}^{(M-2)\times(M-2)}$ are defined as

$$\mathbf{A} = \begin{pmatrix} \frac{1}{h_0} & -\frac{1}{h_0} - \frac{1}{h_1} & \frac{1}{h_1} & 0 & \ldots & 0 & 0 & 0 \\ 0 & \frac{1}{h_1} & -\frac{1}{h_1} - \frac{1}{h_2} & \frac{1}{h_2} & \ldots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & \frac{1}{h_{M-3}} & -\frac{1}{h_{M-3}} - \frac{1}{h_{M-2}} & \frac{1}{h_{M-2}} \end{pmatrix}, \quad (3.6)$$

$$\mathbf{M} = \begin{pmatrix} \frac{h_0+h_1}{3} & \frac{h_1}{6} & 0 & \ldots & 0 & 0 \\ \frac{h_1}{6} & \frac{h_1+h_2}{3} & \frac{h_2}{6} & \ldots & 0 & 0 \\ 0 & \frac{h_2}{6} & \frac{h_2+h_3}{3} & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & \frac{h_{M-4}+h_{M-3}}{3} & \frac{h_{M-3}}{6} \\ 0 & 0 & 0 & \ldots & \frac{h_{M-3}}{6} & \frac{h_{M-3}+h_{M-2}}{3} \end{pmatrix}. \quad (3.7)$$

with $h_i = x_{i+1} - x_i$ for $i = 0, 1, \ldots, M-2$. For the mathematical details on how to derive Equation 3.5 from Equation 3.4, please refer to subsection 3.6.1. Similarly, we can derive the first order derivatives and second order derivatives, which can also be found in subsection 3.6.1.

The advantage of the cubic spline in the functional estimation stage is that, its computational complexity is only a linear polynomial of the sample size. See the following proposition.

Proposition 3.2.1. Given data $\mathcal{D}$ in Equation 3.1, if we use the cubic spline in the functional estimation stage, i.e., estimate $\mathbf{X} \in \mathbb{R}^{MN\times K}$ via the cubic spline in Equation 3.18 with $\alpha \in (0, 1]$ and $\nabla_t \mathbf{u} \in \mathbb{R}^{MN}$ from the cubic spline with $\bar{\alpha} \in (0, 1]$ similar in Equation 3.4,

then the computation complexity in this stage is of order

$$\max\{O(p_{\max}MN), O(K^3)\},$$

where $p_{\max}$ is the highest polynomial order in Equation 3.3, M is the spatial resolution, N is the temporal resolution and K is the number of columns of $\mathbf{X}$.

The proof of the above proposition can be found in subsubsection 3.6.5.

As suggested by Proposition 3.2.1, when $p_{\max}, K \ll M, N$ (which is often the case in practice), it only requires $O(MN)$ numerical operations of the functional estimation stage. This is the lowest possible order of complexity in this stage because MN is exactly the number of the sample size and reading the data is an order $O(MN)$ task. So it can be concluded that it is very efficient to use cubic spline because its computational complexity achieves the lowest possible order of complexity.

For comparison, we discuss the computational complexity of the local polynomial regression, which is widely used in existing literature [see 59, 67, 68, 69, 70, 71].

Proposition 3.2.2. Given data $\mathcal{D}$ in Equation 3.1, if we use the local polynomial regression in the functional estimation stage, i.e., estimate $\mathbf{X} \in \mathbb{R}^{MN\times K}, \nabla_t \mathbf{u} \in \mathbb{R}^{MN}$ via the local polynomial regression described as in subsubsection 3, then the computation complexity of this stage is of order

$$\max\{O(q_{\max}^2 M^2 N), O(MN^2), O(q_{\max}^3 MN), O(p_{\max}MN), O(K^3)\},$$

where $p_{\max}$ is the highest polynomial order in Equation 3.3, $q_{\max}$ is the highest order of derivatives in Equation 3.3, M is the spatial resolution, N is the temporal resolution, and K is the number of columns of $\mathbf{X}$.

If we set $q_{\max} = 2$ to match the derivative order of the local polynomial regression to

the cubic spline, then the computation complexity is of order

$$\max\{O(M^2N), O(MN^2), O(p_{\max}MN), O(K^3)\}.$$

As suggested by Proposition 3.2.2, the computational complexity of local polynomial regression is much higher than that in the cubic spline. But the advantage of local polynomial regression is that it can derive any order of derivatives, i.e., $q_{\max} \geq 0$ in Equation 3.3, while for the cubic spline, $q_{\max} = 2$. In applications, this should be sufficient because most of the PDE models are governed by derivatives up to the second derivative, for instance, heat equation, wave equation, Laplace's equation, Helmholtz equation, Poisson's equation, and so on. In our paper, we mainly use cubic spline as an illustration example due to its simplification and computational efficiency. Readers can extend our proposed SAPDEMI method to the higher-order spline with $q_{\max} > 2$ if they are interested in higher-order derivatives. We summarize the pros and cons of the cubic spline and the local polynomial regression in Table 3.1.

Table 3.1: Pros and cons of the cubic spline and the local polynomial regression in the functional estimation stage

method	cubic spline	local polynomial regression
pros	only requires computational complexity $O(MN)$ in the functional estimation stage	can solve derivatives up to any order
cons	can only solve derivatives up to second order. If higher-order derivatives are required, extensions from cubic spline to higher-order splines are needed.	requires computational complexity $\max\{(M^2N), O(MN^2)\}$ in the functional estimation stage

[1] In this table, we assume that $p_{\max}, q_{\max}, K \ll M, N$ for simplification, where $p_{\max}$ is the highest polynomial order in Equation 3.3, $q_{\max}$ is the highest order of derivatives desirable in Equation 3.3, and K is the number of columns of $\mathbf{X}$ in Equation 3.9.

3.2.2 Model Identification Stage

In this section, we discuss the model identification stage of our proposed SAPDEMI method, where we want to identify the PDE model in Equation 3.3.

The model in Equation 3.3 can be regarded as a linear regression model whose response variable is the derivative with respect to temporal variable t, i.e., $\frac{\partial u(x,t)}{\partial t}$, and the covariates involve with the derivative with respect to spatial variable x, including $\frac{\partial}{\partial x}u(x_i,t_n)$, $\frac{\partial^2}{\partial x^2}u(x_i,t_n)$, $\ldots$, $\left(\frac{\partial^2}{\partial x^2}u(x_i,t_n)\right)^{p_{\max}}$. Because we have MN observations in the dataset $\mathcal{D}$ in Equation 3.1, the response vector is of length MN:

$$\begin{aligned}&\nabla_t \mathbf{u}\\ =\ &(\ \widehat{\frac{\partial u(x_0,t_0)}{\partial t}},\ \widehat{\frac{\partial u(x_1,t_0)}{\partial t}},\ \ldots,\ \widehat{\frac{\partial u(x_{M-1},t_0)}{\partial t}},\ \widehat{\frac{\partial u(x_0,t_1)}{\partial t}},\ \ldots,\ \widehat{\frac{\partial u(x_{M-1},t_{N-1})}{\partial t}}\ )^\top \in \mathbb{R}^{MN},\end{aligned} \tag{3.8}$$

and design matrix is of dimension $MN \times K$:

$$\mathbf{X} \ = \ (\ \widehat{\mathbf{x}_0^0},\ \widehat{\mathbf{x}_1^0},\ \ldots,\ \widehat{\mathbf{x}_{M-1}^0},\ \widehat{\mathbf{x}_1^0},\ \ldots,\ \widehat{\mathbf{x}_{M-1}^{N-1}}\ )^\top \in \mathbb{R}^{MN\times K}, \tag{3.9}$$

where the $nN+i+1$st row of the above matrix $\mathbf{X}$ is

$$\begin{aligned}\widehat{\mathbf{x}_i^n} \ = \ \Bigg(\ 1,\ \ &\widehat{u(x_i,t_n)},\ \ \widehat{\frac{\partial}{\partial x}u(x_i,t_n)},\ \ \widehat{\frac{\partial^2}{\partial x^2}u(x_i,t_n)},\ \ \left(\widehat{u(x_i,t_n)}\right)^2,\ \ \widehat{u(x_i,t_n)}\widehat{\frac{\partial}{\partial x}u(x_i,t_n)},\\ &\widehat{u(x_i,t_0)}\widehat{\frac{\partial^2}{\partial x^2}u(x_i,t_n)},\ldots,\ \left(\widehat{\frac{\partial^2}{\partial x^2}u(x_i,t_n)}\right)^{p_{\max}}\ \Bigg)^\top \in \mathbb{R}^K.\end{aligned}$$

The K components of $\widehat{\mathbf{x}}_i^n$ are candidate terms in the PDE model. And all the derivatives listed in Equation 3.8, Equation 3.9 are estimated from the functional estimation stage in subsection 3.2.1.

After figuring out the response vector $\nabla_t \mathbf{u}$ and the design matrix $\mathbf{X}$, we use Lasso to identify the non-zero coefficients in Equation 3.3:

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2MN}\left\|\nabla_t \mathbf{u} - \mathbf{X}\boldsymbol{\beta}\right\|_2^2, +\lambda\|\boldsymbol{\beta}\|_1 \tag{3.10}$$

where $\lambda > 0$ is a turning parameter that controls the trade off of the sparsity of $\boldsymbol{\beta}$ and the goodness of fit. Given the ℓ_1 penalty in Equation 3.10, $\widehat{\boldsymbol{\beta}}$ will be sparse, i.e., only a few of its entries will likely be non-zero. Accordingly, we can identify the underlying PDE model as

$$\frac{\partial}{\partial t}u(x,t) = \mathbf{x}^\top \widehat{\boldsymbol{\beta}}. \tag{3.11}$$

where

$$\begin{aligned}\mathbf{x} \;=\; \Big(& 1, \quad u(x,t), \quad \tfrac{\partial}{\partial x}u(x,t), \quad \tfrac{\partial^2}{\partial x^2}u(x,t), \quad (u(x,t))^2, \quad u(x_i,t)\tfrac{\partial}{\partial x}u(x,t), \\ & u(x,t)\tfrac{\partial^2}{\partial x^2}u(x,t), \ldots \quad \Big(\tfrac{\partial^2}{\partial x^2}u(x,t)\Big)^{p_{\max}} \Big)^\top \in \mathbb{R}^K.\end{aligned}$$

It remains to discuss the numerical method to solve the optimization problem in Equation 3.10. It is noted that there is no closed-form solution for Equation 3.10 due to the ℓ_1 penalty. The existing algorithms to solve Equation 3.10 is to iteratively update the estimator until convergence. One of the widely used algorithm to solve Equation 3.10 is the coordinate descent, because it is well established in R within a package named *glmnet* [see 86] and Matlab within a function called *lasso(·)*. The main idea of the coordinate descent is to update the estimator in a coordinate-wise fashion, which is the main difference between the coordinate descent and regular gradient descent. And the convergence rate of the coordinate descent to solve Equation 3.10 is $O(1/k)$, where k is the number of iteration executed [see 87, 88].

The implementation of the coordinate descent to Equation 3.10 is presented in algorithm 3 and the detailed description of coordinate gradient descent to solve Equation 3.10 can be found in subsection 3.6.2.

3.2.3 Overview of Our Proposed SAPDEMI method

In this section, we summarized our proposed SAPDEMI method into pseudo-code showing in algorithm 4.

Algorithm 3: Algorithm for the coordinate descent to minimize $F(\boldsymbol{\beta})$

Input: response vector $\nabla_t \mathbf{u}$, design matrix $\mathbf{X}$, and number of iterations M

Output: coefficient estimation $\widehat{\boldsymbol{\beta}}$

Initialize $\boldsymbol{\beta}^{(0)}$

for $\ell = 1, \ldots, \mathcal{L}$ **do**

  for $j = 1, \ldots, K$ **do**

    $\boldsymbol{\beta}_j^{(\ell)} = S\left(\nabla_t \mathbf{u}^\top \mathbf{X}\mathbf{e}_j - \sum_{l \neq j} (\mathbf{X}^\top \mathbf{X})_{jl} \boldsymbol{\beta}_l^{(\ell-1)}, MN\lambda\right) \Big/ (\mathbf{X}^\top \mathbf{X})_{jj}$

$\widehat{\boldsymbol{\beta}} = \boldsymbol{\beta}^{(\mathcal{L})}$

The soft-thresholding function $S(x, \alpha) = (x - \alpha)\mathbb{1}\{x \geq \alpha\} + (x + \alpha)\mathbb{1}\{x \leq -\alpha\} + 0 \times \mathbb{1}\{x \in (-\alpha, \alpha)\}$ where $\mathbb{1}\{x \in A\}$ is an indicator function, i.e., $\mathbb{1}\{x \in A\} = 1$ if $x \in A$, and otherwise $\mathbb{1}\{x \in A\} = 0$.

Algorithm 4: Pseudo code of our proposed SAPDEMI method

Input:

1. Data from the unknown PDE model as in Equation 3.1;
2. Penalty parameter used in the Lasso identify model: $\lambda > 0$;
3. Smoothing parameter used in the cubic spline: $\alpha, \bar{\alpha} \in (0, 1]$.

Output: The identified/recovered PDE model.

Functional estimation stage:

  Estimate $\mathbf{X}, \nabla_t \mathbf{u}$ by cubic spline with $\alpha, \bar{\alpha} \in (0, 1]$.

Model identification stage:

  The unknown PDE system is recovered as: $\frac{\partial}{\partial t} u(x,t) = \mathbf{x}^\top \widehat{\boldsymbol{\beta}}$, where $\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2MN} \|\nabla_t \mathbf{u} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1$ and

  $$\mathbf{x} = \left(1, \quad u(x,t), \quad \frac{\partial u(x,t)}{\partial x}, \quad \frac{\partial^2 u(x,t)}{\partial x^2}, \quad (u(x,t))^2, \quad \ldots, \quad \left(\frac{\partial^2 u(x,t)}{\partial x^2}\right)^{p_{\max}} \right)^\top.$$

3.3 Recovery Theory

In this section, we present our main theorems. These two main theorems serve to evaluate the statistical prosperity of our identified PDE model. The evaluation is done from two aspects.

First, we check if our identified PDE model contains derivatives included in the underlying PDE models. This is the so-called *support set recovery*. Mathematically speaking, it is to check if $\text{supp}(\widehat{\boldsymbol{\beta}}) \subseteq \text{supp}(\boldsymbol{\beta}^*)$, where $\widehat{\boldsymbol{\beta}}$ is the minimizer of Equation 3.10, $\boldsymbol{\beta}^*$ is the ground truth, and $\text{supp}(\cdot)$ is an operator that collects the sets of indices of the non-zero entries of the input variable, i.e., $\text{supp}(\boldsymbol{\beta}) = \{i : \beta_i \neq 0,\ \forall\ i, 1 \leq i \leq K\}$ for a general vector $\boldsymbol{\beta} \in \mathbb{R}^K$. However, the support recovery depends on the choice of the penalty parameter λ. If we choose λ too large, then accordingly $\widehat{\boldsymbol{\beta}}$ would be a vector of all zero entries, which leads to $\text{supp}(\widehat{\boldsymbol{\beta}}) = \emptyset$ (empty set). On the other hand, if we choose λ too small, then $\widehat{\boldsymbol{\beta}}$ is not sparse enough, which makes it fail to identify the PDE models. The proper way to select λ hopefully leads to correct recovery of the support set recovery, i.e., we have $\text{supp}(\widehat{\boldsymbol{\beta}}) \subseteq \text{supp}(\boldsymbol{\beta}^*)$. We will discuss the selection of λ to realize the above objective in Theorem 3.3.1.

Second, we are interested in the estimation error bound of our estimator, i.e., $\left\|\widehat{\boldsymbol{\beta}}_{\mathcal{S}} - \boldsymbol{\beta}^*_{\mathcal{S}}\right\|_\infty$, where $\mathcal{S} = \text{supp}(\boldsymbol{\beta}^*)$, vector $\widehat{\boldsymbol{\beta}}_{\mathcal{S}}$ is the subvector of $\widehat{\boldsymbol{\beta}}$ only containing elements whose indices are in $\mathcal{S}$, and vector $\boldsymbol{\beta}^*_{\mathcal{S}}$ is the subvector of $\boldsymbol{\beta}^*$ only contains elements whose indices are in $\mathcal{S}$. The upper bound of the above estimation error will be discussed in Theorem 3.3.2.

The structure of this section is described as follows. In subsection 3.3.1, we present the conditions for our main theorems. In subsection 3.3.2, we state our two main theorems.

3.3.1 Our Conditions for the Theorems

In this section, we introduce some key conditions used in our paper. We begin with three frequently used conditions in ℓ_1-regularized regression models. They were typically used to provide sufficient conditions for exact sparse recovery [see 58, Chapter 11]. Besides, we also introduce some conditions from cubic splines, which serve for bounding the estimation error of the cubic splines [see 89, (2.5)-(2.8)]. For the verification of these conditions, please refer to subsection 3.4.4 for more details.

Condition 3.3.1 (Invertibility Condition)**.** Suppose for the design matrix $\mathbf{X}$ defined in Equation 3.9 which is constructed by candidates of derivatives, we have that matrix $\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}}$ is invertible almost surely, where $\mathbf{X}_{\mathcal{S}}$ is the columns of $\mathbf{X}$ whose indices are in $\mathcal{S}$. Here $\mathcal{S} = \text{supp}(\boldsymbol{\beta}^*)$ where $\boldsymbol{\beta}^*$ is the ground truth and $\text{supp}(\boldsymbol{\beta}^*) = \{i : \beta_i^* \neq 0,\ \forall\, i, 1 \leq i \leq K\}$.

Condition 3.3.2 (Mutual Incoherence Condition)**.** For some *incoherence parameter* $\mu \in (0, 1]$ and $P_\mu \in [0, 1]$, we have

$$\mathbb{P}\left(\left\|\mathbf{X}_{\mathcal{S}^c}^{\top}\mathbf{X}_{\mathcal{S}}(\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}})^{-1}\right\|_{\infty} \leq 1-\mu\right) \geq P_\mu\ ,$$

where the matrix $\mathbf{X}_{\mathcal{S}}$ is the columns of $\mathbf{X}$ whose indices are in $\mathcal{S}$ and the matrix $\mathbf{X}_{\mathcal{S}^c}$ is the complement of $\mathbf{X}_{\mathcal{S}}$.

Condition 3.3.3 (Minimal Eigenvalue Condition)**.** There exists some constant $C_{\min} > 0$ such that:

$$\Lambda_{\min}\left(\frac{1}{NM}\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}}\right) \geq C_{\min},$$

almost surely. Here $\Lambda_{\min}(\mathbf{A})$ denotes the minimal eigenvalue of a square matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$. This condition can be considered as a strengthened version of Condition 3.3.1.

Condition 3.3.4 (Knots c.d.f. Convergence Condition)**.** Suppose for the sequence of the empirical distribution function over the design points $x_0 < x_1 < \ldots < x_{M-1}$ with different

sample size M is denoted as $F_M(x)$, i.e., $F_M(x) = \frac{1}{M}\sum_{i=0}^{M-1} \mathbb{1}\{x_i \leq x\}$, there exists an absolutely continuous distribution function F on $[x_0, x_{M-1}]$ such that $F_M \to F$ uniformly as $M \to +\infty$. Here $\mathbb{1}\{A\}$ is the indicator of event A. Suppose for the sequence of the empirical distribution function over the design points $t_0 < t_1 < \ldots < t_{N-1}$ with different sample size N is denoted as $G_N(x)$, there exists an absolutely continuous distribution function G on $[t_0, t_{N-1}]$ such that $G_N \to G$ uniformly as $N \to +\infty$.

Condition 3.3.5 (Knots p.d.f. Convergence Condition)**.** Suppose the first derivative of the function F, G (defined in Condition 3.3.4) is denoted as f, g, respectively, then we have

$$0 < \inf_{[x_0, x_{M-1}]} f \leq \sup_{[x_0, x_{M-1}]} f < +\infty \text{ and } 0 < \inf_{[t_0, t_{N-1}]} g \leq \sup_{[t_0, t_{N-1}]} g < +\infty,$$

and f, g also have bounded first derivatives on $[x_0, x_{M-1}]$, $[t_0, t_{N-1}]$, respectively.

Condition 3.3.6 (Gentle Decrease of Smoothing Parameter in Splines Condition)**.** Suppose that $\zeta(M) = \sup_{[x_0, x_{M-1}]} |F_M - F|, \bar{\zeta}(N) = \sup_{[t_0, t_{N-1}]} |G_N - G|$, where F_M, G_M, F, G are defined in Condition 3.3.4. The smoothing parameter $\alpha, \bar{\alpha}$ in Equation 3.4, which are used to estimate the derivatives with respective to x, t, respectively, depend on M, N in such a way that $\alpha \to 0$ and $\alpha^{-1/4}\zeta(M) \to 0$ as $M \to +\infty$. and $\bar{\alpha} \to 0$ and $\bar{\alpha}^{-1/4}\bar{\zeta}(N) \to 0$ as $N \to +\infty$.

3.3.2 Main Theory

In this section, we present our main theory, where Theorem 3.3.1 develops the lower bound of λ to realize the correct recovery of the support set, and Theorem 3.3.2 develops the upper bound of the estimation error.

First, we develop the theory on the lower bound of λ to realize the correct recovery of the support set, i.e., $\mathcal{S}(\widehat{\boldsymbol{\beta}}) \subseteq \mathcal{S}(\boldsymbol{\beta}^*)$, where $\mathcal{S}(\widehat{\boldsymbol{\beta}}) = \{i : \widehat{\beta}_i \neq 0, \ \forall\ i, 1 \leq i \leq K\}$ and $\mathcal{S}(\boldsymbol{\beta}^*) = \{i : \beta_i^* \neq 0, \ \forall\ i, 1 \leq i \leq K\}$. And $\widehat{\boldsymbol{\beta}}$ is the optimum of Equation 3.10, $\boldsymbol{\beta}^*$ is the ground truth of the underlying PDE models.

Theorem 3.3.1. Provided with the data in Equation 3.1 and suppose the conditions in Lemma 3.6.1 and Corollary 3.6.1 hold and Condition 3.3.1 - 3.3.6 also hold, if we take $M = O(N)$, then there exists a constant $\mathscr{C}_{(\sigma,\|u\|_{L^\infty(\Omega)})} > 0$, which is independent of spatial resolution M and temporal resolution N, such that if we set the cubic spline smoothing parameter with the spatial variable x in Equation 3.4 as $\alpha = O\left(\left(1 + M^{-4/7}\right)^{-1}\right)$, set the cubic spline smoothing parameter with the temporal variable t as $\bar{\alpha} = O\left(\left(1 + N^{-4/7}\right)^{-1}\right)$, and set the turning parameter

$$\lambda \geq \mathscr{C}(\sigma, \|u\|_{L^\infty(\Omega)}) \frac{\sqrt{K}\log(N)}{\mu N^{3/7-r}}, \tag{3.12}$$

to identify the PDE model in Equation 3.10 for some $r \in \left(0, \frac{3}{7}\right)$ with sufficient large N, then with probability greater than $P_\mu - \underbrace{O\left(Ne^{-N^r}\right)}_{P'}$, we can have

$$\mathcal{S}(\widehat{\boldsymbol{\beta}}) \subseteq \mathcal{S}(\boldsymbol{\beta}^*).$$

Here K is the number of columns of the design matrix $\mathbf{X}$ in Equation 3.10, and μ, P_μ are defined in Condition 3.3.2.

The proof of the above theorem can be found in subsubsection 3. To ease the understanding of the proofs of main theorems, readers can refer to some lemmas in subsection 3.6.3.

The above theorem states the lower bound of λ to realize the correct recovery of the support set recovery. And this lower bound in Equation 3.12 is affected by several factors. First, it is affected by the temporal resolution N: as N increases, there is more flexibility in tuning this penalty parameter λ. Second, the lower bound in Equation 3.12 is affected by the incoherence parameter μ: if μ is small, then the lower bound increases. This is because small μ means that the group of feature variable candidates are similar to each other. It should be noted that μ is decided by the dataset $\mathcal{D}$ itself (see Condition 3.3.2). Third, this

lower bound in Equation 3.12 is affected by the number of columns of the matrix $\mathbf{X}$. If the number of columns in $\mathbf{X}$ is very large, then it requires larger λ to identify the significant feature variables among lots of feature variable candidates.

We also point out that, the large probability $P_\mu - P'$ converges to P_μ as $N \to +\infty$. This limiting probability P_μ is determined by the data $\mathcal{D}$ (see Condition 3.3.2). So we know that when N is very large, our proposed SAPDEMI method can realize $\mathcal{S}(\widehat{\boldsymbol{\beta}}) \subseteq \mathcal{S}(\boldsymbol{\beta}^*)$.

Now, we develop the theorem to obtain the estimation error bound.

Theorem 3.3.2. Suppose the conditions in Theorem 3.3.1 hold, then with probability greater than $1 - O(Ne^{-N^r}) \to 1$, there exist a $\dot{N} > 0$, such that when $N > \dot{N}$, we have

$$\left\| \widehat{\boldsymbol{\beta}}_{\mathcal{S}} - \boldsymbol{\beta}^*_{\mathcal{S}} \right\|_\infty \leq \sqrt{K} C_{\min} \left(\sqrt{K} \mathscr{C}_{(\sigma, \|u\|_{L^\infty(\Omega)})} \frac{\log(N)}{N^{3/7 - r}} + \lambda \right),$$

where K is the number of columns of the matrix $\mathbf{X}$, $\mathcal{S}$ is the support set of $\boldsymbol{\beta}^*$, i.e., $\mathcal{S} := \{i : \beta_i^* \neq 0,\ \forall i = 1, \ldots, K\}$ and vector $\widehat{\boldsymbol{\beta}}_{\mathcal{S}}, \boldsymbol{\beta}^*_{\mathcal{S}}$ are the subvector of $\widehat{\boldsymbol{\beta}}, \boldsymbol{\beta}^*$ only contains elements whose indices are in $\mathcal{S}$. Viewing from this theorem, we can see that when $N \to +\infty$, the error bound will convergence to 0.

The proof of the above theorem can be found in subsubsection 3.

From the above theorem, we can see that, the estimation error bound for the ℓ_∞-norm of the coefficient error in 3.13 consists of two components. The first component is affected by the temporal resolution N, and the number of feature variable candidates K. As $N \to +\infty$, this first component convergence to 0 without explicit dependence on the choice of feature variable selected from Equation 3.10. The second component is $\sqrt{K} C_{\min} \lambda$. When N increases to $+\infty$, this second component will also converge to 0. This is because, as stated in Theorem 3.3.1, we find that when $N \to +\infty$, the lower bound of λ – which realizes correct support recovery – converges to 0. So the accuracy of the coefficient estimation will improve if we increase the temporal resolution N.

By combining statements in Theorem 3.3.1 and Theorem 3.3.2 together, we find that when the minimum absolute value of the nonzero entries of $\boldsymbol{\beta}^*$ is large enough, then with the adequate choice of lambda, the exact recovery can be guaranteed. Mathematically speaking, when $\min_{i\in\mathcal{S}} |(\boldsymbol{\beta}^*_{\mathcal{S}})_i| > \sqrt{K}C_{\min}\left(\sqrt{K}\mathscr{C}_{(\sigma,\|u\|_{L^\infty(\Omega)})}\frac{\log(N)}{N^{3/7-r}} + \lambda\right)$, the vector $\widehat{\boldsymbol{\beta}}$ will have a correct signed-support, where $(\boldsymbol{\beta}^*_{\mathcal{S}})_i$ refers to the ith element in vector $\boldsymbol{\beta}^*_{\mathcal{S}}$. This helps for the selection of the penalty parameters λ. Besides, the solution paths plot also helps with the selection of the penalty parameters λ, and we will discuss it in section 3.4 under concrete examples.

3.4 Numerical Examples

In this section, we conduct numerical examples to verify the computational efficiency and the statistical accuracy of our proposed SAPDEMI method. The computational efficiency refers to the computational complexity of the functional estimation stage is of a linear polynomial of the sample size MN; The statistical accuracy meas our proposed SAPDEMI can correctly identify the underlying PDE models with high probability.

The numerical examples include (1) the transport equation, (2) the inviscid Burgers' equation and (3) the viscous Burgers' equation. We select these three PDE models as representatives because all these PDE models play fundamental roles in modeling physical phenomenon and demonstrate characteristic behaviors shared by a more complex system, such as dissipation and shock-formation [see 90]. Besides, the difficultly to identify the above PDE models increase from the first example — the transport equation – to the last example – the viscous Burgers' equation..

For the computational efficiency, the results of these three examples are the nearly same, because the computational complexity only depends on the dimension of the noisy data, i.e., the dimension of $\mathbf{X}, \nabla_t \mathbf{u}$, and it is independent of which type of PDE model it comes from. So we only present a detailed discussion of the computational efficiency in the first example – the transport equation.

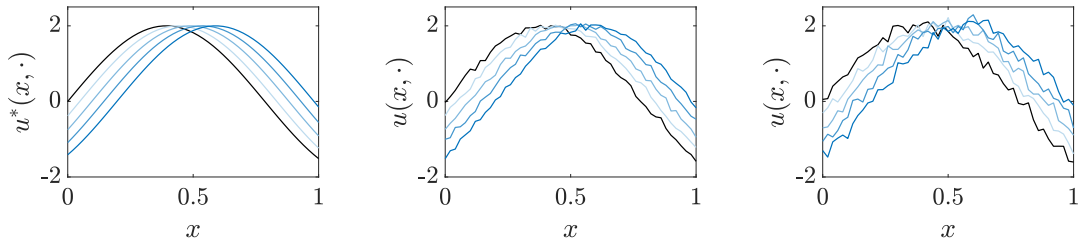
3.4.1 Example 1: Transport Equation

The PDE problem we used in this subsection is the transport equation [see 57, Section 2.2]:

$$\begin{cases} \frac{\partial}{\partial t}u(x,t) &= a\frac{\partial}{\partial x}u(x,t) \quad \forall\, 0 \leq x \leq X_{\max},\ 0 \leq t \leq T_{\max}; \\ u(x,0) &= f(x); \end{cases} \tag{3.13}$$

where we set $f(x) = 2\sin(4x), a = -2, X_{\max} = 1, T_{\max} = 0.1$. From the above seetings, we know there is a closed-form solution, which is $u(x,t) = 2\sin(4x - 8t)$.

The dynamic pattern of the above transport equation is visualized in Figure 3.1. In this figure, (a),(b),(c) are the ground truth, noisy observation under $\sigma = 0.05$ and $\sigma = 0.1$, respectively. From this figure, we can see that the larger the noise, the more un-smoothed the shape of the transport equation would be, which potentially leads to more difficulties in the PDE model identification.



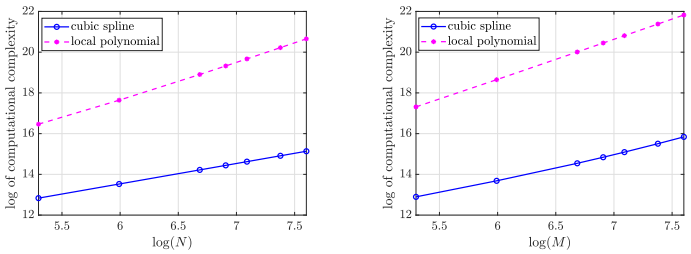
(a) true (b) $\sigma = 0.05$ (c) $\sigma = 0.1$

Figure 3.1: The curves of the transport equation ($M = N = 100$)

First of all, let us take a look at the computational complexity of the functional estimation stage. To show the efficiency of the cubic spline, which is used in our SAPDEMI method, we select the local polynomial regression as a benchmark, whose detailed descriptions can be found in subsubsection 3. We visualize the number of numerical operations in the functional estimation stage of the above two methods in Figure 3.2, where the x-axis is the logarithm of M or N, and the y-axis is the logarithm of the number of numerical operations. The numbers to plot this figure is summarized in Table 3.2 in subsection 3.6.4. In Figure 3.2, two scenarios are discussed: (1) M is fixed as 20 and N varies from 200 to

2000; (2) N is fixed as 20 and M varies from 200 to 2000. As we can see from Figure 3.2 that, as M or N increases, the number of numerical operations in the functional estimation stage becomes larger. And cubic spline needs fewer numerical operations, compared with local polynomial regression. Furthermore, if we conduct a simple linear regression of the four lines in Figure 3.2, we find that in (a), the slope of the cubic spline (blue solid line) is 0.9998, and as N goes to infinity, the slope will get closer to 1. This validates that the computational complexity of cubic spline is of order $O(N)$ when M is fixed (given $K, p_{\max} \ll N$). Besides, in (b) the slope of the cubic spline (blue solid line) is 1.274, and as M goes to infinity, the slope will get closer to 1. This validates that the computational complexity of cubic spline is of order $O(M)$ when N is fixed (given $K, p_{\max} \ll M$). Therefore, we numerically verify that when $K \ll M, N$ and $p_{\max} \ll M, N$, the computational complexity of cubic spline is of order $O(MN)$ (given $K, p_{\max} \ll N$). Similarly, for local polynomial (see Proposition 3.2.2), we know that when M is fixed (Figure 3.2(a)), the slope of the local polynomial regression (pink dashed line) is 1.822, which validates that the computational complexity of the local polynomial is of order $O(N^2)$ when M is fixed. And as N goes to infinity, the slope gets closer to 2. Besides, when N is fixed (Figure 3.2(b)), the slope of the local polynomial regression (pink dashed line) is 1.960, which validate that the computational complexity of local polynomial regression when N is fixed is of order $O(M^2)$. And as M goes to infinity, the slope will get closer to 2. This validates that the computational complexity of the local polynomial regression method is $\max\{O(M^2N), O(MN^2)\}$ when $K, p_{\max} \ll N$.

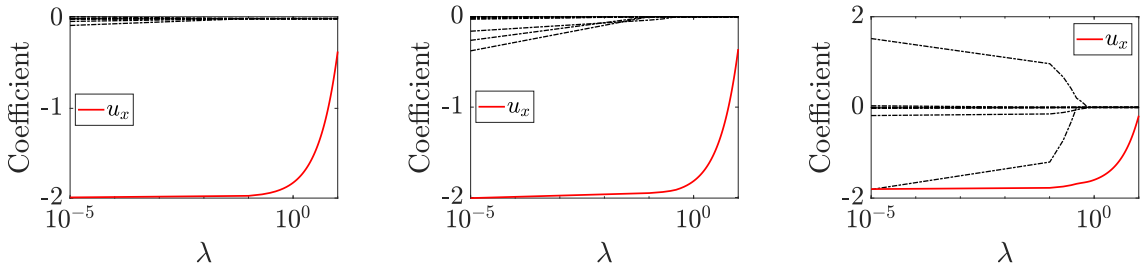
Second, we numerically verify that with high probability, our proposed SAPDEMI can correctly identify the underlying PDE models. From the formula of the transport equation in Equation 3.13, we know that the correct feature variable is only $\frac{\partial}{\partial x}u(x,t)$, which should be identified. While other feature variables, for examples, $u(x,t)$, $\frac{\partial^2}{\partial x^2}u(x,t)$ etc., should not be identified. We discuss the identification accuracy our proposed SAPDEMI under different sample size ($M = N = 100, M = N = 150, M = N = 200$) and different



(a) (b)

Figure 3.2: (a) Computational complexity of cubic spline (blue solid line) & local polynomial regression (red dash line) with fixed M=20, (b) computational complexity of cubic spline (blue solid line) & local polynomial regression (red dashed line) with fixed N=20

magnitude of noise level ($\sigma = \{0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 0.8, 0.9, 1\}$). We find that, for this transport equation, the accuracy stays at 100% under the different magnitude of σ and sample size M, N. To explain the high accuracy, we plot the solution paths in Figure 3.3 under different magnitude of σ, i.e., $\sigma = 0.01, 0.1, 1$. The x-axis of Figure 3.3 is λ, which increases from a very small number to a large number. The y-axis of Figure 3.3 is the coefficients corresponding to all candidates of feature variables. The red lines present the coefficient corresponding to $\frac{\partial}{\partial x}u(x,t)$, which is the correct feature variable. While the black dashed lines present the coefficient corresponding to other incorrect feature variables, which shouldn't be selected. It can be seen from Figure 3.3 that, though large noise increases the difficulty to realize correct identification, we can increase λ to overcome this difficulty, and thus realize correct PDE identification. This solution paths plot can help with the selection of the penalty parameters λ. For (a),(b) in Figure 3.3, we find there is only one solution path that keeps non-zero as λ increase, then this covariate should be identified. For Figure 3.3(c), there are several solution paths that keep non-zero, then the selection of λ depends on (1) how many covariates desirable in the model, (2) the change-points of λ. The change-points refer to the value of λ where the support set changes. For instance, in Figure 3.3(c), the change-points are $\lambda^1 = 0.2, \lambda^2 = 0.4, \lambda^3 = 1$. Since the distance between λ^1, λ^2 is very close, we would much prefer to select λ^3 for correct support recovery.



(a) $\sigma = 0.01$ (b) $\sigma = 0.1$ (c) $\sigma = 1$

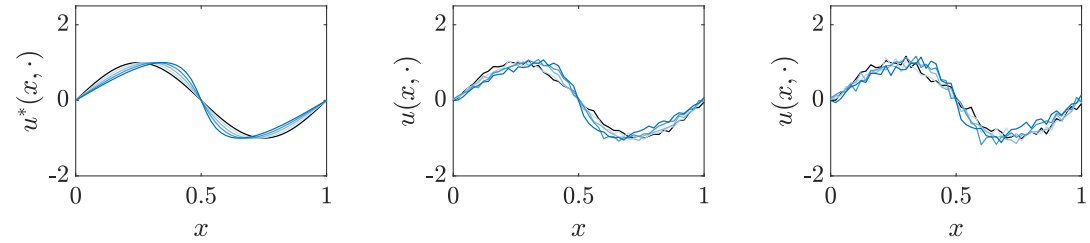
Figure 3.3: The solution paths of the identification in the transport equation under different magnitude of noise levels, i.e., $\sigma = 0.01, 0.1, 1$. The red lines present the coefficient corresponding to $\frac{\partial}{\partial x}u(x,t)$, which is the correct feature variable. While the black dashed lines present the coefficient corresponding to other incorrect feature variables, which shouldn't be selected. Here we set $M = N = 100$, and u_x is the simplification of $\frac{\partial}{\partial x}u(x,t)$.

3.4.2 Example 2: Inviscid Burgers' Equation

In this section, we take a little more challenging example – the inviscid Burgers' equation [see 57, Section 8.4], whose definition is shown as follows:

$$\begin{cases} \frac{\partial}{\partial t}u(x,t) &= -\frac{1}{2}u(x,t)\frac{\partial}{\partial x}u(x,t) & \\ u(x,0) &= f(x) & 0 \le x \le X_{\max} \\ u(0,t) &= u(1,t) = 0 & 0 \le t \le T_{\max} \end{cases}, \tag{3.14}$$

where we set $f(x) = \sin(2\pi x), X_{\max} = 1, T_{\max} = 0.1$. Figure 3.4(a),(b),(c) show the shape of its ground truth, noisy observation under $\sigma = 0.05$, $\sigma = 0.1$, respectively.

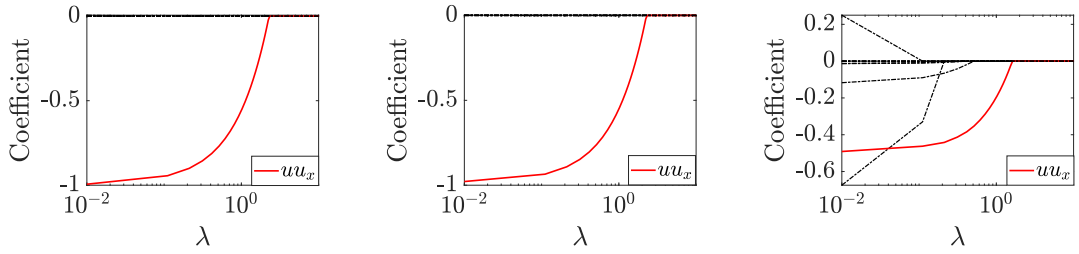


(a) true (b) $\sigma = 0.05$ (c) $\sigma = 0.1$

Figure 3.4: The curves of the inviscid Burgers' equation ($M = 50, N = 50$)

For this inviscid Burgers' equation, we declare that with high probability, our proposed

SAPDEMI can correctly identify it. The simulation results are summarized in Figure 3.8(a) and . From this figure, we find the accuracy stays above 99% when σ ranges from 0.01 to 1. Also, as suggested by Figure 3.8(a), the accuracy decrease as σ increase, which makes sense because large noise makes it more difficult to realize correct identification. The effect of noise to PDE identification can be found in the solution paths plot in Figure 3.5, where we can see that the length of λ-interval for correct identification decreases as σ increases.



(a) $\sigma = 0.01$ (b) $\sigma = 0.5$ (c) $\sigma = 1$

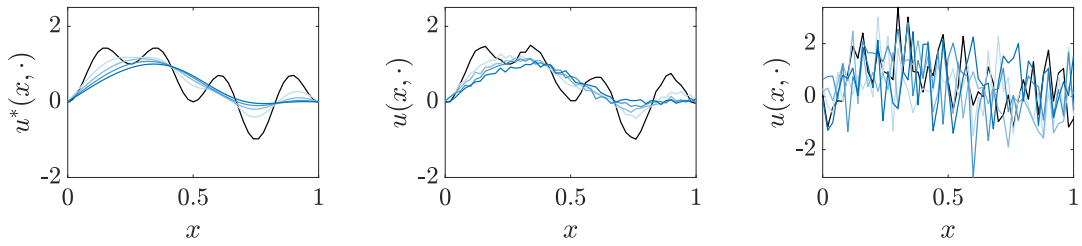
Figure 3.5: The solution paths of the identification in the inviscid Burger's equation under different magnitude of noise levels, i.e., $\sigma = 0.01, 0.5, 1$. The red lines present the coefficient corresponding to $u(x,t)\frac{\partial}{\partial x}u(x,t)$, which is the correct feature variable. While the black dashed lines present the coefficient corresponding to other incorrect feature variables, which shouldn't be selected. Here we set $M = N = 100$ and t he label u, u_x are the simplification of to $u(x,t), \frac{\partial}{\partial x}u(x,t)$, respectively.

3.4.3 Example 3: Viscous Burgers' Equation

In this section, we take a more challenging example, i.e., viscous Burgers' equation [see 57, Section 8.4]:

$$\begin{cases} \frac{\partial}{\partial t}u(x,t) &= -\frac{1}{2}u(x,t)\frac{\partial}{\partial x}u(x,t) + \nu\frac{\partial^2}{\partial x^2}u(x,t) & \\ u(x,0) &= f(x) & 0 \le x \le X_{\max} \\ u(0,t) &= u(1,t) = 0 & 0 \le t \le T_{\max} \end{cases}, \tag{3.15}$$

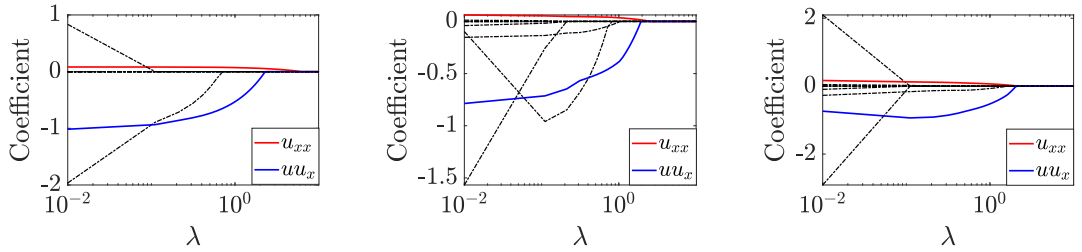
where we set $f(x) = \sin^2(4\pi x) + \sin^3(2\pi x), X_{\max} = 1, T_{\max} = 0.1, \nu = 0.1$. Figure 3.6 shows the shape of the viscous Burgers' equation, where (a),(b),(c) are the ground truth, noisy observation under $\sigma = 0.05$ and $\sigma = 0.1$, respectively.



(a) true (b) $\sigma = 0.05$ (c) $\sigma = 1$

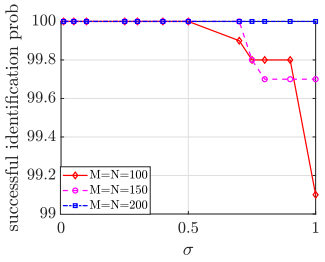
Figure 3.6: The curves of the viscous Burgers' equation ($M = 50, N = 50$)

Based on the simulation results in Figure 3.8(b) and Table 3.3, we conclude that with high probability, our proposed SAPDEMI can correctly identify the underlying viscous Burgers' equation, with the reasons given as follows. When $M = N = 200$, the accuracy stays at 100% for all levels of $\sigma \in [0.01, 1]$. When $M = N = 150$, the accuracy stays above 90% for all levels of $\sigma \in [0.01, 1]$. When $M = N = 100$, the accuracy are above 70% when $\sigma \in [0.01, 0.5]$, and reduces to around 50% when $\sigma = 1$, which makes sense because as reselected by Figure 3.7, when σ increase from 0.01 to 1, the length of λ-interval for correct identification decreases, which make it more difficult to realize correct identification. So if we encounter a heavily noised dataset $\mathcal{D}$, a larger sample size is preferred.

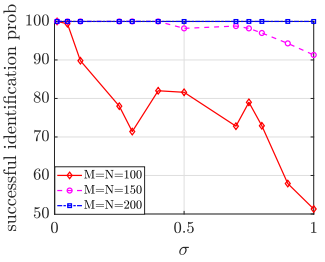


(a) $\sigma = 0.01$ (b) $\sigma = 0.5$ (c) $\sigma = 1$

Figure 3.7: The solution paths of the identification in the viscous Burger's equation under different magnitude of noise levels, i.e., $\sigma = 0.01, 0.5, 1$ The red and blue lines present the coefficient corresponding to $\frac{\partial^2}{\partial x^2}u(x,t)$ and $u(x,t)\frac{\partial}{\partial x}u(x,t)$, respectively, which are the correct feature variables, while the black dashed lines present the coefficient corresponding to other incorrect feature variables, which shouldn't be selected. Here $M = N = 100$ and the label u_{xx}, uu_x are the simplification of $u(x,t)\frac{\partial}{\partial x}u(x,t)$, $\frac{\partial^2}{\partial x^2}u(x,t)$, respectively.



(a) example 2: inviscid Burgers equation



(b) example 3: viscous Burgers equation

Figure 3.8: The successful identification probability curves under different magnitude of σ and sample size M, N. The successful identification probability of the example 1 – transport equation – stays in 100% for $\sigma \in [0.01, 1]$ and $M, N \in \{100, 150, 200\}$, and we neglect the figure for this 100% accuracy. Seeing from these figures, we can find that example 3 – viscous Burgers equation – is the hardest, and example 1 – transport equation – is the easiest. (The numbers to plot this figure can be found in Table 3.3 in subsection 3.6.4.)

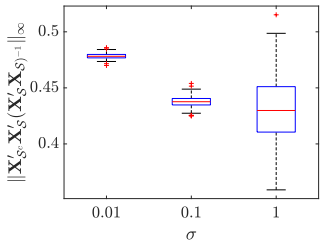
3.4.4 Checking Conditions of Example 1,2,3

In this section, we check Condition 3.3.1 - Condition 3.3.6 of the above three examples: (1) example 1 (the transport equation), (2) example 2 (the inviscid Burgers' equation) and (3) example 3 (the viscous Burgers' equation).

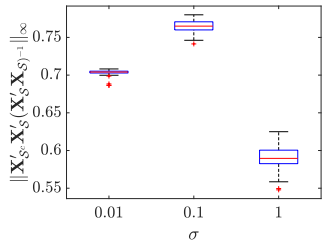
Verification of Condition 3.3.1, 3.3.2, 3.3.3

In this section, we check the Condition 3.3.1 - Condition 3.3.3 under example 1,2,3, though the applicability of the results is by no means restricted to these.

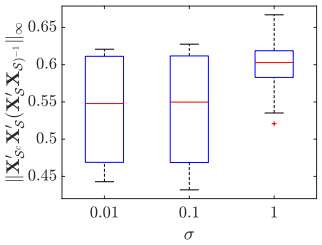
The verification results can be found in Figure 3.9 and Figure 3.10, where (a),(b),(c) are the box plot of $\left\|\mathbf{X}_{\mathcal{S}^c}^\top \mathbf{X}_{\mathcal{S}} (\mathbf{X}_{\mathcal{S}}^\top \mathbf{X}_{\mathcal{S}})^{-1}\right\|_\infty$ and the minimal eigenvalue of matrix $\frac{1}{NM}\mathbf{X}_{\mathcal{S}}^\top \mathbf{X}_{\mathcal{S}}$ of these three examples under $\sigma = 0.01, 0.1, 1$, respectively. From Figure 3.9, we find the value of $\left\|\mathbf{X}_{\mathcal{S}^c}^\top \mathbf{X}_{\mathcal{S}} (\mathbf{X}_{\mathcal{S}}^\top \mathbf{X}_{\mathcal{S}})^{-1}\right\|_\infty$ is smaller than 1, so there exist a $\mu \in (0, 1]$ such that Condition 3.3.2 is met. From Figure 3.10, we find the minimal eigenvalue of matrix $\frac{1}{MN}\mathbf{X}_{\mathcal{S}}^\top \mathbf{X}$ are all strictly larger than 0, so we declare Condition 3.3.3 is satisfied, and thus its weak version – Condition 3.3.1 – is also satisfied.



(a) example 1: transport equation

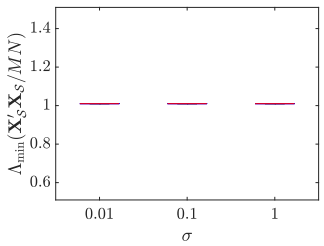


(b) example 2: inviscid Burgers' equation

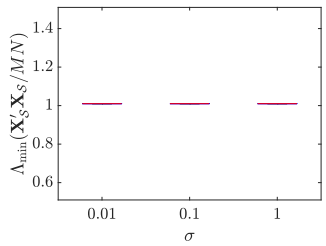


(c) example 3: viscous Burgers' equation

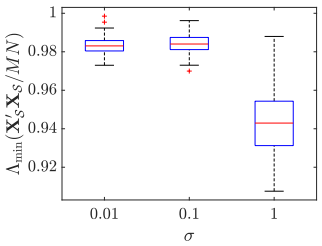
Figure 3.9: Box plots of $\left\|\mathbf{X}_{\mathcal{S}^c}^\top \mathbf{X}_{\mathcal{S}}(\mathbf{X}_{\mathcal{S}}^\top \mathbf{X}_{\mathcal{S}})^{-1}\right\|_\infty$ under $\sigma = 0.01, 0.1, 1$ when $M = N = 100$.



(a) example 1: transport equation



(b) example 2: inviscid Burgers' equation



(c) example 3: viscous Burgers' equation

Figure 3.10: Box plots of the minimal eigenvalue of matrix $\frac{1}{NM}\mathbf{X}_{\mathcal{S}}^\top \mathbf{X}_{\mathcal{S}}$ under $\sigma = 0.01, 0.1, 1$ when $M = N = 100$.

Verification of Condition 3.3.4 and Condition 3.3.5

In example 1,2,3, the design points $x_0, x_1, \ldots, x_{M-1}$ and $t_0, t_1, \ldots, t_{N-1}$ are equally spaced, i.e., $x_0 = 1/M, x_1 = 2/M, \ldots, x_{M-1} = 1$ and $t_0 = 0.1/N, t_1 = 0.2/N, \ldots, t_{N-1} = 0.1$. Under this scenario, there exist an absolutely continuous distribution $F(x) = x$ for $x \in [1/M, 1]$ and $G(t) = 0.1t$ for $t \in [0.1/N, 0.1]$, where the empirical c.d.f. of the design points $x_0, x_1, \ldots, x_{M-1}$ and $t_0, t_1, \ldots, t_{N-1}$ will converge to $F(x), G(t)$, respectively, as $M, N \to +\infty$. For the $F(x), G(t)$, we know their first derivatives is bounded for $x \in [1/M, 1]$ and $t \in [0.1/N, 0.1]$, respectively. In the simulation of this paper, we take the equally spaced design points as an illustration example, and its applicability is by no means restricted to this case.

Verification of Condition 3.3.6.

The Condition 3.3.6 ensures that the smoothing parameter does not tend to zero too rapidly. [89] shows that for the equally spaced design points, this condition meets. For other types of design points, for instance, randomly and independently distributed design points, it can also be verified that Condition 3.3.6 is satisfied [see 89, Section 2].

3.5 Conclusion

In this paper, we propose a two-stage method called SAPDEMI to efficiently identify the underlying PDE models from the noisy data in $\mathcal{D}$. In the first stage – functional estimation stage – we employ the cubic spline to estimate the unobservable derivatives, which serve as input variables for the second stage. In the second stage – model identification stage – we apply the Lasso to identify the underlying PDE model. The contributions of our proposed SAPDEMI method are: (1) it is computationally efficient because it only requires the computational complexity of order $O(MN)$, which achieves the lowest possible order of complexity; (2) we focus on the model selections, while the existing literature mostly

focuses on parameter estimations; (3) we develop asymptotic properties of our method for correct identification, which is not reported by the existing literature.

After developing this SAPDEMI method, we realize there are lots of promising future research directions. First, in our paper, we take $x \in \mathbb{R}$ as an illustration example, and it would be interesting to investigate the case when the spatial variable $\mathbf{x} \in \mathbb{R}^d$ ($d \geq 2$) due to its wide existence in practice. Second, future research could consider the interaction between the spatial variable and the temporal variable. For instance, we can explore the time-varying coefficient $\boldsymbol{\beta}(t) = (\beta_1(t), \ldots, \beta_K(t))^\top$ in Equation 3.3. Besides, we can also consider the case when the ϵ_i^n in Equation 3.2 has spatial-temporal patterns. In our paper, because we aim at showing the methodology to solve the PDE identification problem, we do not discuss the above future research directions in detail and hopefully, our paper provides a good starting point for further research.

3.6 Supplementary Material

3.6.1 Derivation of the 0-th, First, Second Derivative of the Cubic Spline

In this section, we focus on solving the derivatives of $u(x, t_n)$ with respective to x, i.e., $\left\{u(x_i, t_n), \frac{\partial}{\partial x}u(x_i, t_n), \frac{\partial^2}{\partial x^2}u(x_i, t_n)\right\}_{i=0,1\ldots,M-1}$ for any $n = 0, 1, \ldots, N-1$. To realize this objective, we first fix t as t_n for a general $n \in \{0, 1, \ldots, N-1\}$. Then we use cubic spline to fit data $\{(x_i, u_n^i)\}_{i=0,1,\ldots,M-1}$.

Suppose the cubic polynomial spline over the knots $\{(x_i, u_n^i)\}_{i=0,1,\ldots,M-1}$ is $s(x)$. So under good approximation, we can regard $s(x), s'(x), s''(x)$ as the estimators of $u(x_i, t_n)$, $\frac{\partial}{\partial x}u(x, t_n)$, $\frac{\partial^2}{\partial x^2}u(x, t_n)$, where $s'(x), s''(x)$ is the first and second derivatives of $s(x)$, respectively.

Let first take a look at the zero-order derivatives of $s(x)$. By introducing matrix algebra, the objective function in Equation 3.4 can be rewritten as

$$J_\alpha(s) = \alpha(\mathbf{u}_\cdot^n - \mathbf{f})^\top \mathbf{W}(\mathbf{u}_\cdot^n - \mathbf{f}) + (1-\alpha)\mathbf{f}^\top \mathbf{A}^\top \mathbf{M}^{-1}\mathbf{A}\mathbf{f} \tag{3.16}$$

where vector

$$\mathbf{f} = \begin{pmatrix} s(x_0) \\ s(x_1) \\ \vdots \\ s(x_{M-1}) \end{pmatrix} \triangleq \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{M-1} \end{pmatrix}, \mathbf{u}_{:}^{n} = \begin{pmatrix} u_0^n \\ u_1^n \\ \vdots \\ u_{M-1}^n \end{pmatrix}$$

and matrix $\mathbf{W} = \text{diag}(w_0, w_1, \ldots, w_{M-1})$ and matrix $\mathbf{A}$ is defined in Equation 3.6. By taking the derivative of Equation 3.16 with respective to $\mathbf{f}$ and set it as zero, we have

$$\widehat{\mathbf{f}} = [\alpha\mathbf{W} + (1-\alpha)\mathbf{A}^\top\mathbf{M}\mathbf{A}]^{-1}\alpha\mathbf{W}\mathbf{u}_{:}^{n}. \tag{3.17}$$

Then we solve the second-order derivative with respective to x. Let us first suppose that the cubic spline $s(x)$ in $[x_i, x_{i+1}]$ is denoted $s_i(x)$, and we denote $s_i''(x_i) = \sigma_i, s_i''(x_{i+1}) = \sigma_{i+1}$. Then we have $\forall x \in [x_i, x_{i+1}]$ $(0 \leq i \leq M-2)$,

$$s_i''(x) = \sigma_i \frac{x_{i+1} - x}{h_i} + \sigma_{i+1}\frac{x - x_i}{h_i},$$

where matrix $\mathbf{M}$ is defined in Equation 3.7. This is because $s_i''(x)$ with $x \in [x_i, x_{i+1}]$ is a linear function. By taking a double integral of the above equation, we have

$$s_i(x) = \frac{\sigma_i}{6h_i}(x_{i+1} - x)^3 + \frac{\sigma_{i+1}}{6h_i}(x - x_i)^3 + c_1(x - x_1) + c_2(x_{i+1} - x), \tag{3.18}$$

where c_1, c_2 is the unknown parameters to be estimated. Because $s_i(x)$ interpolates two endpoints (x_i, f_i) and (x_{i+1}, f_{i+1}) , if we plug x_i, x_{i+1} into the above $s_i(x)$, we have

$$\begin{cases} f_i &= s_i(x_i) = \frac{\sigma_i}{6}h_i^2 + c_2 h_i \\ f_{i+1} &= s_i(x_{i+1}) = \frac{\sigma_{i+1}}{6}h_i^2 + c_1 h_i, \end{cases}$$

where we can solve c_1, c_2 as

$$\begin{cases} c_1 &= (f_{i+1} - \frac{\sigma_{i+1}}{6}h_i^2)/h_i \\ c_2 &= (f_i - \frac{\sigma_i}{6}h_i^2)/h_i. \end{cases}$$

By plugging in the value of c_1, c_2 into Equation 3.18, we have ($0 \leq i \leq M-2$)

$$s_i(x) = \frac{\sigma_i}{6h_i}(x_{i+1}-x)^3 + \frac{\sigma_{i+1}}{6h_i}(x-x_i)^3 + \left(\frac{f_{i+1}}{h_i} - \frac{\sigma_{i+1}h_i}{6}\right)(x-x_i) + \left(\frac{f_i}{h_i} - \frac{\sigma_i h_i}{6}\right)(x_{i+1}-x)$$

with its first derivative as

$$s_i'(x) = -\frac{\sigma_i}{2h_i}(x_{i+1}-x)^2 + \frac{\sigma_{i+1}}{2h_i}(x-x_i)^2 + \frac{f_{i+1}-f_i}{h_i} - \frac{h_i}{6}(\sigma_{i+1}-\sigma_i). \tag{3.19}$$

Because $s_{i-1}'(x_i) = s_i'(x_i)$, we have ($1 \leq i \leq M-2$)

$$\frac{1}{6}h_{i-1}\sigma_{i-1} + \frac{1}{3}(h_{i-1}+h_i)\sigma_i + \frac{1}{6}h_i\sigma_{i+1} = \frac{f_{i+1}-f_i}{h_i} - \frac{f_i - f_{i-1}}{h_{i-1}}. \tag{3.20}$$

Equation Equation 3.20 gives $M-2$ equations. Recall $\sigma_0 = \sigma_{M-1} = 0$, so totally we get M equations, which is enough to solve M parameters, i.e., $\sigma_0, \sigma_1, \ldots, \sigma_{M-1}$. We write out the above system of linear equations, where we hope to identify a fast numerical approach to solve it. The system of linear equations is:

$$\begin{cases} & \frac{1}{3}(h_0+h_1)\sigma_1 & + & \frac{1}{6}h_1\sigma_2 & = & \frac{u_2^n-u_1^n}{h_1} - \frac{f_1-u_0}{h_0} \\ \frac{1}{6}h_1\sigma_1 & + & \frac{1}{3}(h_1+h_2)\sigma_2 & + & \frac{1}{6}h_2\sigma_3 & = & \frac{f_3-f_2}{h_1} - \frac{f_2-f_1}{h_0} \\ & & \vdots & & & & \\ \frac{1}{6}h_{M-4}\sigma_{M-4} & + & \frac{1}{3}(h_{M-4}+h_{M-3})\sigma_{M-3} & + & \frac{1}{6}h_{M-3}\sigma_{M-2} & = & \frac{f_{M-2}-f_{M-3}}{h_{M-3}} - \frac{f_{M-3}-f_{M-4}}{h_{M-4}} \\ \frac{1}{6}h_{M-3}\sigma_{M-3} & + & \frac{1}{3}(h_{M-3}+h_{M-2})\sigma_{M-2} & & & = & \frac{f_{M-1}-f_{M-2}}{h_{M-2}} - \frac{f_{M-2}-f_{M-3}}{h_{M-3}} \end{cases}.$$

From the above system of equation, we can see that the second derivative of cubic spline $s(x)$ can be solved by the above system of linear equation, i.e.,

$$\widehat{\boldsymbol{\sigma}} = \mathbf{M}^{-1}\mathbf{A}\widehat{\mathbf{f}} \tag{3.21}$$

where vector $\widehat{\mathbf{f}}$ is defined in Equation 3.17, matrix $\mathbf{A} \in \mathbb{R}^{(M-2)\times M}$ is defined in Equation 3.6, and matrix $\mathbf{M} \in \mathbb{R}^{(M-2)\times(M-2)}$ is defined as Equation 3.7.

Finally, we focus on solving the first derivative of cubic spline $s(x)$. Let $\theta_i = s'(x_i)$ for $i = 0, 1, \ldots, M-1$, then we have

$$\begin{aligned}
s_i(x) &= \theta_i \tfrac{(x_{i+1}-x)^2(x-x_i)}{h_i^2} - \theta_{i+1}\tfrac{(x-x_i)^2(x_{i+1}-x)}{h_i^2} + f_i \tfrac{(x_{i+1}-x)^2[2(x-x_i)+h_i]}{h_i^3} + \\
&\quad f_{i+1}\tfrac{(x-x_i)^2[2(x_{i+1}-x)+h_i]}{h_i^3} \\
s_i'(x) &= \theta_i \tfrac{(x_{i+1}-x)(2x_i+x_{i+1}-3x)}{h_i^2} - \theta_{i+1}\tfrac{(x-x_i)(2x_{i+1}+x_i-3x)}{h_i^2} + 6\tfrac{u_{i+1}^n-u_i^n}{h_i^3}(x_{i+1}-x)(x-x_i) \\
s_i''(x) &= -2\theta_i \tfrac{2x_{i+1}+x_i-3x}{h_i^2} - 2\theta_{i+1}\tfrac{2x_i+x_{i+1}-3x}{h_i^2} + 6\tfrac{u_{i+1}^n-u_i^n}{h_i^3}(x_{i+1}+x_i-2x)
\end{aligned}$$

By plugging x_i into $s_i''(x)$ and $s_{i-1}''(x)$, we have

$$\begin{cases}
s_i''(x) &= -2\theta_i \tfrac{2x_{i+1}+x_i-3x}{h_i^2} - 2\theta_{i+1}\tfrac{2x_i+x_{i+1}-3x}{h_i^2} + 6\tfrac{f_{i+1}-f_i}{h_i^3}(x_{i+1}+x_i-2x) \\
s_{i-1}''(x) &= -2\theta_{i-1} \tfrac{2x_i+x_{i-1}-3x}{h_{i-1}^2} - 2\theta_i\tfrac{2x_{i-1}+x_i-3x}{h_{i-1}^2} + 6\tfrac{f_i-f_{i-1}}{h_{i-1}^3}(x_i+x_{i-1}-2x)
\end{cases}$$

which gives

$$\begin{cases}
s_i''(x) &= \tfrac{-4}{h_i}\theta_i + \tfrac{-2}{h_i}\theta_{i+1} + 6\tfrac{f_{i+1}-f_i}{h_i^2} \\
s_{i-1}''(x) &= \tfrac{2}{h_{i-1}}\theta_{i-1} + \tfrac{4}{h_{i-1}}\theta_i - 6\tfrac{f_i-f_{i-1}}{h_{i-1}^2}.
\end{cases}$$

Because $s_i''(x_i) = s_{i-1}''(x_i)$, we have ($\forall i = 1, 2, \ldots, M-2$)

$$\begin{aligned}
&\tfrac{-4}{h_i}\theta_i + \tfrac{-2}{h_i}\theta_{i+1} + 6\tfrac{f_{i+1}-f_i}{h_i^2} = \tfrac{2}{h_{i-1}}\theta_{i-1} + \tfrac{4}{h_{i-1}}\theta_i - 6\tfrac{f_i-f_{i-1}}{h_{i-1}^2} \\
\Leftrightarrow\quad &\tfrac{2}{h_{i-1}}\theta_{i-1} + (\tfrac{4}{h_{i-1}} + \tfrac{4}{h_i})\theta_i + \tfrac{2}{h_i}\theta_{i+1} = 6\tfrac{f_{i+1}-f_i}{h_i^2} + 6\tfrac{f_i-f_{i-1}}{h_{i-1}^2} \\
\Leftrightarrow\quad &\tfrac{1}{h_{i-1}}\theta_{i-1} + (\tfrac{2}{h_{i-1}} + \tfrac{2}{h_i})\theta_i + \tfrac{1}{h_i}\theta_{i+1} = 3\tfrac{f_{i+1}-f_i}{h_i^2} + 3\tfrac{f_i-f_{i-1}}{h_{i-1}^2}.
\end{aligned}$$

By organizing the above system of equation into matrix algebra, we have

$$\begin{pmatrix} \frac{1}{h_0} & \frac{2}{h_0}+\frac{2}{h_1} & \frac{1}{h_1} & 0 & \dots & 0 & 0 & 0 \\ 0 & \frac{1}{h_1} & \frac{2}{h_1}+\frac{2}{h_2} & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \frac{1}{h_{M-3}} & \frac{2}{h_{M-3}}+\frac{2}{h_{M-2}} & \frac{1}{h_{M-2}} \end{pmatrix} \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{M-1} \end{pmatrix}$$

$$= \begin{pmatrix} 3\frac{f_2-f_1}{h_1^2}+3\frac{f_1-f_0}{h_0^2} \\ 3\frac{f_3-f_2^n}{h_2^2}+3\frac{f_2-f_1}{h_1^2} \\ \vdots \\ 3\frac{f_{M-1}-f_{M-2}}{h_{M-2}^2}+3\frac{f_{M-2}^n-f_{M-3}}{h_{M-3}^2} \end{pmatrix}.$$

For the endpoint θ_0, because $s_0''(x_0)=0$, we have

$$s_0''(x) = -2\theta_0\frac{2x_1+x_0-3x}{h_0^2} - 2\theta_1\frac{2x_0+x_1-3x}{h_0^2} + 6\frac{f_1-f_0}{h_0^3}(x_1+x_0-2x).$$

When we take the value of x as x_0, we have

$$\begin{aligned} s_0''(x_0) &= -2\theta_0\tfrac{2x_1+x_0-3x_0}{h_0^2} - 2\theta_1\tfrac{2x_0+x_1-3x_0}{h_0^2} + 6\tfrac{f_1-f_0}{h_0^3}(x_1+x_0-2x_0) \\ &= \tfrac{-4}{h_0}\theta_0 + \tfrac{-2}{h_0}\theta_1 + 6\tfrac{f_1-f_0}{h_0^2} \\ &= 0 \end{aligned}$$

For the two endpoint θ_{M-1}, because $s_{M-2}''(x_{M-1})=0$, we have

$$\begin{aligned} s_{M-2}''(x) &= -2\theta_{M-2}\tfrac{2x_{M-1}+x_{M-2}-3x}{h_{M-2}^2} - 2\theta_{M-1}\tfrac{2x_{M-2}+x_{M-1}-3x}{h_{M-2}^2} + \\ &\qquad 6\tfrac{f_{M-1}-f_{M-2}}{h_{M-2}^3}(x_{M-1}+x_{M-2}-2x) \end{aligned}$$

When we take the value of x as x_{M-1}, we have

$$\begin{aligned} s''_{M-2}(x_{M-1}) &= -2\theta_{M-2}\tfrac{2x_{M-1}+x_{M-2}-3x_{M-1}}{h^2_{M-2}} - 2\theta_{M-1}\tfrac{2x_{M-2}+x_{M-1}-3x_{M-1}}{h^2_{M-2}} + \\ &\quad 6\tfrac{f_{M-1}-f_{M-2}}{h^3_{M-2}}(x_{M-1}+x_{M-2}-2x_{M-1}) \\ &= \tfrac{2}{h_{M-2}}\theta_{M-2} + \tfrac{4}{h_{M-2}}\theta_{M-1} - 6\tfrac{f_{M-1}-f_{M-2}}{h^2_{M-2}} \\ &= 0. \end{aligned}$$

So the first order derivative $\boldsymbol{\theta} = (\theta_0, \theta_1, \ldots, \theta_{M-1})^\top$ can be solved by

$$\underbrace{\begin{pmatrix} \frac{2}{h_0} & \frac{1}{h_0} & 0 & 0 & \ldots & 0 & 0 & 0 \\ \frac{1}{h_0} & \frac{2}{h_0}+\frac{2}{h_1} & \frac{1}{h_1} & 0 & \ldots & 0 & 0 & 0 \\ 0 & \frac{1}{h_1} & \frac{2}{h_1}+\frac{2}{h_2} & 0 & \ldots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & \frac{1}{h_{M-3}} & \frac{2}{h_{M-3}}+\frac{2}{h_{M-2}} & \frac{1}{h_{M-2}} \\ 0 & 0 & 0 & 0 & \ldots & 0 & \frac{1}{h_{M-2}} & \frac{2}{h_{M-2}} \end{pmatrix}}_{\mathbf{Q}\in\mathbb{R}^{M\times M}} \underbrace{\begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \vdots \\ \theta_{M-1} \end{pmatrix}}_{\boldsymbol{\theta}}$$

$$= \underbrace{\begin{pmatrix} 3\frac{f_1-f_0}{h_0^2} \\ 3\frac{f_2-f_1}{h_1^2} + 3\frac{f_1-f_0}{h_0^2} \\ 3\frac{f_3-f_2}{h_2^2} + 3\frac{f_2-f_1}{h_1^2} \\ \vdots \\ 3\frac{f_{M-1}-f_{M-2}}{h_{M-2}^2} + 3\frac{f_{M-2}-f_{M-3}}{h_{M-3}^2} \\ 3\frac{f_{M-1}-f_{M-2}}{h_{M-2}^2} \end{pmatrix}}_{\mathbf{q}}$$

In matrix algebra, the first order derivative $\boldsymbol{\theta} = (\theta_0, \theta_1, \ldots, \theta_{M-1})^\top$ can be solved by

$$\widehat{\boldsymbol{\theta}} = \mathbf{Q}^{-1}\widehat{\mathbf{q}} = \mathbf{Q}^{-1}\mathbf{B}\widehat{\mathbf{f}}, \tag{3.22}$$

where $\widehat{\mathbf{f}}$ is defined in Equation 3.17, and matrix $\mathbf{B} \in \mathbb{R}^{M\times M}$ is defined as

$$\mathbf{B} = \begin{pmatrix} \frac{-3}{h_0^2} & \frac{3}{h_0^2} & 0 & 0 & \dots & 0 & 0 & 0 \\ \frac{-3}{h_0^2} & \frac{3}{h_0^2} - \frac{3}{h_1^2} & \frac{3}{h_1^2} & 0 & \dots & 0 & 0 & 0 \\ 0 & \frac{-3}{h_1^2} & \frac{3}{h_1^2} - \frac{3}{h_2^2} & \frac{3}{h_2^2} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ & 0 & 0 & 0 & 0 & \frac{-3}{h_{M-3}^2} & \frac{3}{h_{M-3}^2} - \frac{3}{h_{M-2}^2} & \frac{3}{h_{M-2}^2} \\ & 0 & 0 & 0 & 0 & 0 & \frac{-3}{h_{M-2}^2} & \frac{3}{h_{M-2}^2} \end{pmatrix}.$$

3.6.2 Coordinate Gradient Descent Used in the Model Identification Stage

In this section, we briefly review the implement of the coordinate descent algorithm in [86] to solve Equation 3.10. The main idea of the coordinate descent is to update the estimator in a coordinate-wise fashion, which is the main difference between the coordinate descent and regular gradient descent. For instance, in the k-th iteration, the coordinate descent updates the iterative estimator $\boldsymbol{\beta}^{(k)}$ by using partial of the gradient information, instead of the whole gradient information. Mathematically speaking, in the k-th iteration, the coordinate descent optimizes $F(\boldsymbol{\beta}) = \frac{1}{2MN}\|\nabla_t \mathbf{u} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1$ with respective to $\boldsymbol{\beta}$ by

$$\beta_j^{(k+1)} = \arg\min_{\beta_j} F((\beta_1^{(k)}, \beta_2^{(k)}, \dots, \beta_{j-1}^{(k)}, \beta_j, \beta_{j+1}^{(k)}, \dots, \beta_K^{(k)}))$$

for all $j = 1, 2, \dots, K$. To minimize the above optimization problem, we can derive the first derivative and set it as 0:

$$\frac{\partial}{\partial \beta_j} F(\boldsymbol{\beta}^{(k)}) = \frac{1}{MN}\left(\mathbf{e}_j^\top \mathbf{X}^\top \mathbf{X}\boldsymbol{\beta}^{(k)} - \nabla_t \mathbf{u}^\top \mathbf{X}\mathbf{e}_j\right) + \lambda \text{sign}(\boldsymbol{\beta}_j) = 0,$$

where $\mathbf{e}_j$ is a vector of length K whose entries are all zero expect the j-th entry is 1. By solving the above equation, we can solve $\beta_j^{(k+1)}$ by

$$\beta_j^{(k+1)} = S\left(\nabla_t \mathbf{u}^\top \mathbf{X}\mathbf{e}_j - \sum_{l \neq j} (\mathbf{X}^\top \mathbf{X})_{jl} \boldsymbol{\beta}_l^{(k)}, MN\lambda\right) \Big/ (\mathbf{X}^\top \mathbf{X})_{jj},$$

where $S(\cdot)$ is the soft-thresholding function defined as

$$S(x, \alpha) = \begin{cases} x - \alpha & \text{if } x \geq \alpha \\ x + \alpha & \text{if } x \leq -\alpha \\ 0 & \text{otherwise} \end{cases}.$$

The detailed procedure of this algorithm is summarized in algorithm 3.

3.6.3 Some Important Lemmas

In this section, we present some important preliminaries, which are important blocks for the proofs of the main theories. To begin with, we first give the upper bound of $\widehat{u(x, t_n)} - u(x, t_n)$ for $x \in \{x_0, x_1, \ldots, x_{M-1}\}$, which is distance between the ground truth $u(x, t_n)$ and the estimated zero-order derivatives by cubic spline $\widehat{u(x, t_n)}$.

Lemma 3.6.1. Assume that

1. for any fixed $n = 0, 1, \ldots, N-1$, we have the spatial variable x is sorted in nondecreasing order, i.e., $x_0 < x_1 \ldots < x_{M-1}$;

2. for any fixed $n = 0, 1, \ldots, N-1$, we have the ground truth function $f^*(x) := u(x, t_n) \in C^4$, where C^4 refers to the set of functions that is forth-time differentiable;

3. for any fixed $n = 0, 1, \ldots, N-1$, we have $\frac{\partial^2}{\partial x^2} u(x_0, t_n) = \frac{\partial^2}{\partial x^2} u(x_{M-1}, t_n) = 0$, and $\frac{\partial^3}{\partial x^3} u(x_0, t_n) \neq 0$, $\frac{\partial^3}{\partial x^3} u(x_{M-1}, t_n) = 0$;

4. for any fixed $n = 0, 1, \ldots, N-1$, the value of third order derivative of function $f^*(x) := u(x, t_n)$ at point $x = 0$ is bounded, i.e., $\frac{d^3}{dx^3} f^*(0) < +\infty$;

5. for any U_i^n generated by the underlying PDE system $U_i^n = u(x_i, t_n) + w_i^n$ with $w_i^n \overset{i.i.d}{\sim} N(0, \sigma^2)$, we have $\eta^2 := \max_{i=0,\ldots,M-1, n=0,\ldots,N-1} E(U_i^n)^2$ is bounded;

6. for function $K(x) = \frac{1}{2} e^{-|x|/\sqrt{2}} \left[\sin(|x|\sqrt{2}) + \pi/4\right]$, we assume that it is uniformly continuous with modulus of continuity w_K and of bounded variation $V(K)$ and we also assume that $\int |K(x)|dx$, $\int |x|^{1/2}|dK(x)|$, $\int |x \log |x||^{1/2}|dK(x)|$ are bounded and denote $K_{\max} := \max_{x \in \max_{x\in[0,X_{\max}]\cup[0,T_{\max}]} K(x)}$;

7. the smoothing parameter in Equation 3.4 is set as $\alpha = \left(1 + M^{-4/7}\right)^{-1}$;

8. the Condition 3.3.4 - Condition 3.3.5 hold.

Then there exist finite positive constant $\mathscr{C}_{(\sigma,\|u\|_{L^\infty(\Omega)})} > 0, C_{(\sigma,\|u\|_{L^\infty(\Omega)})} > 0, \widetilde{C}_{(\sigma,\|u\|_{L^\infty(\Omega)})} > 0, Q_{(\sigma,\|u\|_{L^\infty(\Omega)})} > 0, \gamma_{(M)} > 0, \omega_{(M)} > 1$, such that for any ϵ satisfying

$$\begin{aligned} \epsilon \;>\; \mathscr{C}_{(\sigma,\|u\|_{L^\infty(\Omega)})} \max \;& \Big\{ 4K_{\max} M^{-3/7}, 4AM^{-3/7}, 4\sqrt{2}\tfrac{d^3}{dx^3} f^*(0) M^{-3/7}, \\ & \tfrac{16\left[C_{(\sigma,\|u\|_{L^\infty(\Omega)})} \log(M) + \gamma_{(M)}\right]\log(M)}{M^{3/7}}, \\ & 16\sqrt{\tfrac{\omega_{(M)}}{7}} \widetilde{C}_{(\sigma,\|u\|_{L^\infty(\Omega)})} \tfrac{\sqrt{\log(M)}}{M^{3/7}} \Big\}, \end{aligned}$$

there exist a $\dot{M} > 0$, such that when $M > \dot{M}$, we have

$$\begin{aligned} & P\left[\sup_{x\in[0,X_{\max}]} \left| \frac{\partial^k}{\partial x^k} \widehat{u(x,t_n)} - \frac{\partial^k}{\partial x^k} u(x,t_n) \right| > \epsilon \right] \\ < \;& 2Me^{-\frac{(M^{3/7} - \|u\|_{L^\infty(\Omega)})^2}{2\sigma^2}} + \\ & Q_{(\sigma,\|u\|_{L^\infty(\Omega)})} e^{-L\gamma_{(M)}} + 4\sqrt{2}\eta^4 M^{-\omega_{(M)}/7} \end{aligned}$$

for $k = 0, 1, 2$. Here $A = \sup_\alpha \int |u|^s f_M(\alpha, u)du \times \int_{x\in[0,X_{\max}]} |K(x)|dx$.

Proof. See subsubsection 3. □

In the above lemma, we add $(\sigma, \|u\|_{L^\infty(\Omega)})$ as the subscript of constants $\mathscr{C}, C, \widetilde{C}, Q$ to emphasize that these constant are independent of the temporal resolution N and spatial

resolution M, and only depends on the noisy data $\mathcal{D}$ in Equation 3.1 itself. We add M as the subscript of constants γ, ω to emphasize that γ, ω are function of the spatial resolution M, and we will discuss the value of γ, ω in Lemma 3.6.2.

The above lemma show the closeness between $\frac{\partial^k}{\partial x^k}\widehat{u(x,t_n)}$ and $\frac{\partial^k}{\partial x^k}u(x,t_n)$ for $k = 0,1,2$. This results can be easily extend of the closeness between $\frac{\partial}{\partial t}\widehat{u(x_i,t)}$ and $\frac{\partial}{\partial t}u(x_i,t)$, which is shown in the following corollary.

Corollary 3.6.1. Assume that

1. for any fixed $i = 0,1,\ldots,M-1$, we have the spatial variable t is sorted in nondecreasing order, i.e., $t_0 < t_1 \ldots < t_{N-1}$;

2. for any fixed $i = 0,1,\ldots,M-1$, we have the ground truth function $f^*(t) := u(x_i,t) \in C^4$, where C^4 refers to the set of functions that is forth-time differentiable;

3. for any fixed $i = 0,1,\ldots,M-1$, we have $\frac{\partial^2}{\partial t^2}u(x_i,t_0) = \frac{\partial^2}{\partial t^2}u(x_i,t_{N-1}) = 0$, and $\frac{\partial^3}{\partial t^3}u(x_i,t_0) \neq 0, \frac{\partial^3}{\partial t^3}u(x_i,t_{N-1}) = 0$;

4. for any fixed $i = 0,1,\ldots,M-1$, the value of third order derivative of function $\bar{f}^*(x) := u(x_i,t)$ at point $t = 0$ is bounded, i.e., $\frac{d^3}{dt^3}\bar{f}^*(0) < +\infty$;

5. for any U_i^n generated by the underlying PDE system $U_i^n = u(x_i,t_n) + w_i^n$ with $w_i^n \overset{i.i.d}{\sim} N(0,\sigma^2)$, we have $\max_{i=0,\ldots,M-1,n=0,\ldots,N-1} E(U_i^n)^2$ is bounded;

6. for function $K(x) = \frac{1}{2}e^{-|x|/\sqrt{2}}\left[\sin(|x|\sqrt{2}) + \pi/4\right]$, we have $K(x)$ is uniformly continuous with modulus of continuity w_K and of bounded variation $V(K)$, and we also assume that $\int_{x\in[0,X_{\max}]}|K(x)|dx$, $\int|x|^{1/2}|dK(x)|$, $\int|x\log|x||^{1/2}|dK(x)|$ are bounded and denote $K_{\max} := \max_{x\in[0,X_{\max}]\cup[0,T_{\max}]}K(x)$;

7. the smoothing parameter in Equation 3.4 is set as $\bar{\alpha} = O\left(\left(1+N^{-4/7}\right)^{-1}\right)$;

8. the Condition 3.3.4 - Condition 3.3.5 hold.

then there exist finite positive constant $\mathscr{C}_{(\sigma,\|u\|_{L^\infty(\Omega)})} > 0, C_{(\sigma,\|u\|_{L^\infty(\Omega)})} > 0, \widetilde{C}_{(\sigma,\|u\|_{L^\infty(\Omega)})} > 0, Q_{(\sigma,\|u\|_{L^\infty(\Omega)})} > 0, \gamma_{(N)} > 0, \omega_{(N)} > 1$, such that for any ϵ satisfying

$$\begin{aligned}\epsilon \;>\; \mathscr{C}_{(\sigma,\|u\|_{L^\infty(\Omega)})} \max \;\Big\{ & 4K_{\max}N^{-3/7}, 4\bar{A}N^{-3/7}, 4\sqrt{2}\tfrac{d^3}{dx^3}f^*(0)N^{-3/7}, \\ & \tfrac{16\left[C_{(\sigma,\|u\|_{L^\infty(\Omega)})}\log(N)+\gamma_{(N)}\right]\log(N)}{N^{3/7}}, \\ & 16\sqrt{\tfrac{\omega_{(N)}}{7}}\widetilde{C}_{(\sigma,\|u\|_{L^\infty(\Omega)})}\tfrac{\sqrt{\log(N)}}{N^{3/7}}\Big\},\end{aligned}$$

there exist a $\dot{N} > 0$, such that when $N > \dot{N}$, we have

$$\begin{aligned}P\left[\sup_{t\in[0,T_{\max}]}\left|\frac{\partial}{\partial t}\widehat{u(x_i,t)} - \frac{\partial}{\partial t}u(x_i,t)\right| > \epsilon\right] \;<\; & 2Ne^{-\frac{(N^{3/7}-\|u\|_{L^\infty(\Omega)})^2}{2\sigma^2}} + \\ & Q_{(\sigma,\|u\|_{L^\infty(\Omega)})}e^{-L\gamma_{(N)}} + 4\sqrt{2}\eta^4 N^{-\omega_{(N)}/7}.\end{aligned}$$

Here $\bar{A} = \sup_\alpha \int |u|^s \bar{f}_N(\alpha,u)du \times \int_{t\in[0,T_{\max}]} |K(x)|dx$.

After bounding the error of all the derivatives, we then aim to bound $\|\nabla_t \mathbf{u} - \mathbf{X}\boldsymbol{\beta}^*\|_\infty$. It is important to bound $\|\nabla_t \mathbf{u} - \mathbf{X}\boldsymbol{\beta}^*\|_\infty$, with the reason described as follows in Lemma 3.6.2.

Lemma 3.6.2. Suppose the conditions in Lemma 3.6.1 and Corollary 3.6.1 hold and we set $M = O(N)$, then there exist finite positive constant $\mathscr{C}_{(\sigma,\|u\|_{L^\infty(\Omega)})} > 0$ such that for any ϵ satisfying

$$\epsilon > \mathscr{C}_{(\sigma,\|u\|_{L^\infty(\Omega)})}\frac{\log(N)}{N^{3/7-r}},$$

and any $r \in \left(0, \frac{3}{7}\right)$, there exist $\dot{N} > 0$, such that when $N > \dot{N}$, we have

$$P\left(\|\nabla_t \mathbf{u} - \mathbf{X}\boldsymbol{\beta}^*\|_\infty > \epsilon\right) < Ne^{-N^r},$$

where $\mathscr{C}_{(\sigma,\|u\|_{L^\infty(\Omega)})}$ is a constant which do not depend on the temporal resolution M and spatial resolution N.

Proof. See subsubsection 3. □

3.6.4 Tables to Draw the Curves in the Numerical Examples

In this section, we present the table to draw the curves in Figure 3.2,Figure 3.8 in Table 3.2, Table 3.3, respectively.

Table 3.2: Computational complexity of the functional estimation by cubic spline and local polynomial regression in transport equation

	M = 20						
	N=200	N=400	N=800	N=1000	N=1200	N=1600	N=2000
cubic spline	374,389	748,589	1,496,989	1,871,189	2,245,389	2,993,789	3,742,189
local poly	14,136,936	45,854,336	162,089,136	246,606,536	348,723,936	605,758,736	933,193,536
	N = 20						
	M=200	M=400	M=800	M=1000	M=1200	M=1600	M=2000
cubic spline	398,573	875,773	207,0173	2,787,373	3,584,573	5,418,973	7,573,373
local poly	33,046,336	125,596,136	489,255,736	760,365,536	1,090,995,336	1,930,814,936	3,008,714,536

Table 3.3: Correct identification probability of transport equation, inviscid Burgers equation and viscous Burgers's equation

	σ											
	0.01	0.05	0.1	0.25	0.3	0.4	0.5	0.7	0.75	0.8	0.9	1
	transport equation											
$M = N = 100$	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
$M = N = 150$	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
$M = N = 200$	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	inviscid Burgers equation											
$M = N = 100$	100%	100%	100%	100%	100%	100%	100%	99.9%	99.8%	99.8%	99.8%	99.1%
$M = N = 150$	100%	100%	100%	100%	100%	100%	100%	100%	99.8%	99.7%	99.7%	99.7%
$M = N = 200$	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	viscous Burgers equation											
$M = N = 100$	100%	99.4%	89.8%	78.0%	71.4%	82.0%	91.6%	72.8%	79.0%	72.9%	57.9%	51.3%
$M = N = 150$	100%	100%	100%	97.3%	96.5%	96.2%	97.6%	95.6%	93.3%	86.6%	79.9%	73.6%
$M = N = 200$	100%	100%	100%	100%	99.6%	99.6%	98.2%	98.8%	98.2%	97.0%	94.3%	91.3%

[1] The simulation results are based on 1000 times of simulations.

3.6.5 Proofs

Proof of Proposition 3.2.1

Proof. The computational complexity in the functional estimation stage lies in calculating all elements in matrix $\mathbf{X}$ and vector $\nabla_t \mathbf{u}$, including

$$\left\{ \widehat{u(x_i,t_n)}, \widehat{\frac{\partial}{\partial x}u(x_i,t_n)}, \widehat{\frac{\partial^2}{\partial x^2}u(x_i,t_n)}, \widehat{\frac{\partial}{\partial t}u(x_i,t_n)} \right\}_{i=0,\ldots,M-1,n=0,\ldots,N-1}.$$

by cubic spline in Equation 3.4.

We divide our proof into two scenarios: (1) $\alpha = 1$ and (2) $\alpha \in (0, 1)$.

- First of all, we discuss a very simple case, i.e., $\alpha = 1$. When $\alpha = 1$, we call the cubic spline as *interpolating cubic spline* since there is no penalty on the smoothness.

  For the zero-order derivative, i.e., $\left\{\widehat{u(x_i, t_n)}\right\}_{i=0,\ldots,M-1,n=0,\ldots,N-1}$, it can be estimated as $\widehat{u(x_i, t_n)} = u_i^n$ for $i = 0, 1, \ldots, M-1, n = 0, 1, \ldots, N-1$. So there is no computational complexity involved.

  For the second order derivatives, i.e., $\left\{\frac{\partial^2}{\partial x^2}\widehat{u(x_i, t_n)}\right\}_{i=0,\ldots,M-1}$, with $n \in \{0, \ldots, N-1\}$ fixed, it can be solved in a closed-form, i.e.,

$$\widehat{\boldsymbol{\sigma}} = \mathbf{M}^{-1}\mathbf{A}\mathbf{u}_{\cdot}^n$$

  where $\widehat{\boldsymbol{\sigma}} = \left(\frac{\partial^2}{\partial x^2}\widehat{u(x_0, t_n)}, \frac{\partial^2}{\partial x^2}\widehat{u(x_1, t_n)}, \ldots, \frac{\partial^2}{\partial x^2}\widehat{u(x_{M-1}, t_n)}\right)^\top$. So the main computational load lies in the calculation of $\mathbf{M}^{-1}$. Recall $\mathbf{M} \in \mathbb{R}^{(M-2)\times(M-2)}$ is a tri-diagonal matrix:

$$\mathbf{M} = \begin{pmatrix} \frac{h_0+h_1}{3} & \frac{h_1}{6} & 0 & \ldots & 0 & 0 \\ \frac{h_1}{6} & \frac{h_1+h_2}{3} & \frac{h_2}{6} & \ldots & 0 & 0 \\ 0 & \frac{h_2}{6} & \frac{h_2+h_3}{3} & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \ddots & \frac{h_{M-4}+h_{M-3}}{3} & \frac{h_{M-3}}{6} \\ 0 & 0 & 0 & \ldots & \frac{h_{M-3}}{6} & \frac{h_{M-3}+h_{M-2}}{3} \end{pmatrix}.$$

  For this type of tri-diagonal matrix, there exist a fast algorithm to calculate its inverse. The main idea of this fast algorithm is to decompose $\mathbf{M}$ through Cholesky decomposition as

$$\mathbf{M} = \mathbf{L}\mathbf{D}\mathbf{L}^\top,$$

where $\mathbf{L} \in \mathbb{R}^{(M-2)\times(M-2)}, \mathbf{D} \in \mathbb{R}^{(M-2)\times(M-2)}$ has the form of

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ l_1 & 1 & 0 & \dots & 0 \\ 0 & l_2 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & l_{M-3} & 1 \end{pmatrix}, \mathbf{D} = \begin{pmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & d_{M-2} \end{pmatrix}.$$

After decomposing matrix $\mathbf{M}$ into $\mathbf{LDL}^\top$, the second derivatives $\widehat{\boldsymbol{\sigma}}$ can be solved as

$$\widehat{\boldsymbol{\sigma}} = (\mathbf{L}^\top)^{-1}\mathbf{D}^{-1}\mathbf{L}^{-1}\underbrace{\mathbf{A}\mathbf{u}_{\cdot}^{n}}_{\boldsymbol{\xi}}.$$

In the remaining of the proof in this scenario, we will verify the following two issues:

1. the computational complexity to decompose $\mathbf{M}$ into $\mathbf{LDL}^\top$ is $O(M)$ with $n \in \{0, \dots, N-1\}$ fixed;
2. the computational complexity to compute $\widehat{\boldsymbol{\sigma}} = (\mathbf{L}^\top)^{-1}\mathbf{D}^{-1}\mathbf{L}^{-1}\boldsymbol{\xi}$ is $O(M)$ with $n \in \{0, \dots, N-1\}$ fixed and $\mathbf{L}, \mathbf{D}$ available.

For the decomposition of $\mathbf{M} = \mathbf{LDL}^\top$, its essence is to derive $l_1, \dots l_{M-3}$ in matrix $\mathbf{L}$ and $d_1, \dots, d_{M-2}$ in matrix $\mathbf{D}$. By utilizing the method of undetermined coefficients

to inequality $\mathbf{M} = \mathbf{L}\mathbf{D}\mathbf{L}^\top$, we have:

$$\begin{aligned}
&\begin{bmatrix}
d_1 & d_1 l_1 & 0 & \dots & 0 & 0 \\
d_1 l_1 & d_2 & d_2 l_2 & \dots & 0 & 0 \\
0 & d_2 l_2 & d_3 & \dots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \dots & d_{M-3} l_{M-3} & d_{M-3} l_{M-3}^2 + d_{M-2}
\end{bmatrix} \\
= &\begin{bmatrix}
M_{11} & M_{12} & \dots & 0 \\
M_{21} & M_{22} & \dots & 0 \\
0 & M_{32} & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \dots & M_{M-2,M-2}
\end{bmatrix},
\end{aligned}$$

where $M_{i,j}$ is the (i,j)th entry in matrix $\mathbf{M}$. Through the above method of undetermined coefficients, we can solve the exact value of the entries in matrix $\mathbf{L}, \mathbf{D}$, which is summarized in algorithm 5. It can be seen from algorithm 5 that, the computational complexity of solve $\mathbf{L}, \mathbf{D}$ is of order $O(M)$.

For the calculation of $\widehat{\boldsymbol{\sigma}} = (\mathbf{L}^\top)^{-1}\mathbf{D}^{-1}\mathbf{L}^{-1}\boldsymbol{\xi}$ with matrix $\mathbf{L}, \mathbf{D}$ available, we will first verify that the computational complexity to solve $\bar{\boldsymbol{\xi}} = \mathbf{L}^{-1}\boldsymbol{\xi}$ is $O(M)$. Then, we will verify that the computational complexity to solve $\bar{\bar{\boldsymbol{\xi}}} = \mathbf{D}^{-1}\bar{\boldsymbol{\xi}}$ is $O(M)$. Finally, we will verify that the computational complexity to solve $\bar{\bar{\bar{\boldsymbol{\xi}}}} = (\mathbf{L}^\top)^{-1}\bar{\bar{\boldsymbol{\xi}}}$ is $O(M)$. First, the computational complexity of calculating $\bar{\boldsymbol{\xi}} = \mathbf{L}^{-1}\boldsymbol{\xi}$ is $O(M)$, this is because by $\mathbf{L}\bar{\boldsymbol{\xi}} = \boldsymbol{\xi}$, we have the following system of equations:

$$\begin{cases}
\xi_1 = \bar{\xi}_1 \\
\xi_2 = \bar{\xi}_2 + l_1 \bar{\xi}_1 \\
\vdots \\
\xi_{M-2} = \bar{\xi}_{M-2} + l_{M-3} \bar{\xi}_{M-3}
\end{cases}$$

where $\xi_i, \bar{\xi}_i$ is the i-th entry in $\boldsymbol{\xi}, \bar{\boldsymbol{\xi}}$,respectively. Through the above system of equations, we can solve the values of all entries in $\bar{\boldsymbol{\xi}}$, which is summarized in algorithm 6. From algorithm 6, we know that the computational complexity of solving $\mathbf{L}^{-1}\boldsymbol{\xi}$ is $O(M)$. Next, it is obvious that the computational complexity of $\bar{\bar{\boldsymbol{\xi}}} = \mathbf{D}^{-1}\bar{\boldsymbol{\xi}}$ is $O(M)$, because $\mathbf{D}$ is a diagonal matrix. Finally, with the similar logic flow, we can verify that the computational complexity of $\bar{\bar{\bar{\boldsymbol{\xi}}}} = (\mathbf{L}^\top)^{-1}\bar{\bar{\boldsymbol{\xi}}}$ is still $O(M)$. So, the computational complexity is to calculate $\widehat{\boldsymbol{\sigma}} = (\mathbf{L}^\top)^{-1}\mathbf{D}^{-1}\mathbf{L}^{-1}\boldsymbol{\xi}$, with known $\mathbf{L}, \mathbf{D}$ is $O(M)$.

As a summary, the computational complexity is to calculate $\left\{ \widehat{\frac{\partial^2}{\partial x^2} u(x_i, t_n)} \right\}_{i=0,\ldots,M-1}$ with a fixed $n \in \{0, 1, \ldots, N-1\}$ is $O(M)$. Accordingly, the computational complexity to solve $\left\{ \widehat{\frac{\partial^2}{\partial x^2} u(x_i, t_n)} \right\}_{i=0,\ldots,M-1,n=0,\ldots,N-1}$ is $O(MN)$.

For the first order derivatives, i.e., $\left\{ \frac{\partial}{\partial x} u(x_i, t_n), \widehat{\frac{\partial}{\partial x} u(x_i, t_n)} \right\}_{i=0,\ldots,M-1,n=0,\ldots,N-1}$, we can verify the computational complexity to solve them is also $O(MN)$ with the similar logic as that in the second order derivatives.

Algorithm 5: Pseudo code to solve $\mathbf{L}, \mathbf{D}$

Input: matrix $\mathbf{M}$
Output: matrix $\mathbf{L}, \mathbf{D}$
Initialize $d_1 = M_{1,1}$
for $i = 1, 2, \ldots, M-3$ **do**
    $l_i = M_{i,i+1}/d_i$
    $d_{i+1} = M_{i+1,i+1} - d_i l_i^2$

Algorithm 6: Pseudo code to solve $\mathbf{L}^{-1}\boldsymbol{\xi}$

Input: matrix $\mathbf{L}, \boldsymbol{\xi}$
Output: matrix $\bar{\boldsymbol{\xi}}$
Initialize $\bar{\xi}_1 = \xi_1$
for $i = 2, \ldots, M-2$ **do**
    $\bar{\xi}_i = \xi_i - l_{i-1}\bar{\xi}_{i-1}$

- Next, we discuss the scenario when $\alpha \in (0, 1)$.

  Since all the derivatives has similar closed-form formulation as shown in Equation 3.5, Equation 3.22, Equation 3.21, we take the zero-order derivative

$\{u(x_i, t_n)\}_{i=0,\ldots,M-1,n=0,\ldots,N-1}$ as an illustration example, and other derivatives can be derived similarly.

Recall that in subsection 3.2.1, the zero-order derivative $\{u(x_i, t_n)\}_{i=0,\ldots,M-1}$ with $n \in \{0, 1, \ldots, N-1\}$ fixed can be estimated through cubic spline as in Equation 3.5:

$$\widehat{\mathbf{f}} = \underbrace{[\alpha\mathbf{W} + (1-\alpha)\mathbf{A}^\top\mathbf{M}\mathbf{A}]}_{\mathbf{Z}}^{-1} \underbrace{\alpha\mathbf{W}\mathbf{u}_{:}^{n}}_{\mathbf{y}},$$

where $\alpha \in (0, 1)$ trades off the fitness of the cubic spline and the smoothness of the cubic spline, vector $\widehat{\mathbf{f}} = \left(\widehat{u(x_0, t_n)}, \widehat{u(x_1, t_n)}, \ldots, \widehat{u(x_{M-1}, t_n)}\right)^\top$, vector $\mathbf{u}_{:}^{n} = \left(u_0^n, \ldots, u_{M-1}^n\right)^\top$, matrix $\mathbf{W} = \text{diag}(w_0, w_1, \ldots, w_{M-1})$, matrix $\mathbf{A} \in \mathbb{R}^{(M-2)\times M}$, $\mathbf{M} \in \mathbb{R}^{(M-2)\times(M-2)}$ are defined as

$$\mathbf{A} = \begin{pmatrix} \frac{1}{h_0} & -\frac{1}{h_0} - \frac{1}{h_1} & \frac{1}{h_1} & 0 & \ldots & 0 & 0 & 0 \\ 0 & \frac{1}{h_1} & -\frac{1}{h_1} - \frac{1}{h_2} & \frac{1}{h_2} & \ldots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & \frac{1}{h_{M-3}} & -\frac{1}{h_{M-3}} - \frac{1}{h_{M-2}} & \frac{1}{h_{M-2}} \end{pmatrix},$$

$$\mathbf{M} = \begin{pmatrix} \frac{h_0+h_1}{3} & \frac{h_1}{6} & 0 & \ldots & 0 & 0 \\ \frac{h_1}{6} & \frac{h_1+h_2}{3} & \frac{h_2}{6} & \ldots & 0 & 0 \\ 0 & \frac{h_2}{6} & \frac{h_2+h_3}{3} & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & \frac{h_{M-4}+h_{M-3}}{3} & \frac{h_{M-3}}{6} \\ 0 & 0 & 0 & \ldots & \frac{h_{M-3}}{6} & \frac{h_{M-3}+h_{M-2}}{3} \end{pmatrix}$$

with $h_i = x_{i+1} - x_i$ for $i = 0, 1, \ldots, M-2$.

By simple calculation, we know that matrix $\mathbf{Z} = \alpha\mathbf{W} + (1-\alpha)\mathbf{A}^\top\mathbf{M}\mathbf{A} \in \mathbb{R}^{M\times M}$

is a symmetric seventh-diagonal matrix:

$$\mathbf{Z} = \begin{pmatrix} z_{11} & z_{12} & z_{13} & z_{14} & 0 & \dots \\ z_{21} & z_{22} & z_{23} & z_{24} & z_{25} & \dots \\ z_{31} & z_{32} & z_{33} & z_{34} & z_{35} & \ddots \\ z_{41} & z_{42} & z_{43} & z_{44} & z_{45} & \ddots \\ 0 & z_{52} & z_{53} & z_{54} & z_{55} & \ddots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}$$

By applying Cholesky decomposition to matrix $\mathbf{Z}$ as $\mathbf{Z} = \mathbf{P}\mathbf{\Sigma}\mathbf{P}^\top$, we can calculate $\widehat{\mathbf{f}}$ as

$$\widehat{\mathbf{f}} = \mathbf{Z}^{-1}\mathbf{y} = (\mathbf{P}^\top)^{-1}\mathbf{\Sigma}^{-1}\mathbf{P}^{-1}\mathbf{y},$$

where $\mathbf{P} \in \mathbb{R}^{M\times M}$, $\mathbf{\Sigma} \in \mathbb{R}^{M\times M}$ has the form of

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ \ell_1 & 1 & 0 & 0 & \dots & 0 & 0 \\ \gamma_1 & \ell_2 & 1 & 0 & \dots & 0 & 0 \\ \eta_1 & \gamma_2 & \ell_3 & 1 & \dots & 0 & 0 \\ 0 & \eta_2 & \gamma_3 & \ell_4 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & 1 & 0 \\ 0 & 0 & \dots & \eta_{M-3} & \gamma_{M-2} & \ell_{M-1} & 1 \end{pmatrix}, \mathbf{\Sigma} = \begin{pmatrix} s_1 & 0 & 0 & \dots & 0 \\ 0 & s_2 & 0 & \dots & 0 \\ 0 & 0 & s_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \dots & s_M \end{pmatrix}.$$

In the remaining of the proof in this scenario, we will verify the following two issues:

1. the computational complexity to decompose $\mathbf{Z}$ into $\mathbf{P}\mathbf{\Sigma}\mathbf{P}^\top$ is $O(M)$ with $n \in \{0, \dots, N-1\}$ fixed;
2. the computational complexity to compute $(\mathbf{P}^\top)^{-1}\mathbf{\Sigma}^{-1}\mathbf{P}^{-1}\mathbf{y}$ is $O(M)$ with $n \in \{0, \dots, N-1\}$ fixed.

First of all, we verify that the computational complexity to decompose $\mathbf{Z}$ into $\mathbf{P}\mathbf{\Sigma}\mathbf{P}^\top$ is $O(M)$ when $n \in \{0, \ldots, N-1\}$ fixed By applying method of undetermined coefficients to equality $\mathbf{Z} = \mathbf{P}\mathbf{\Sigma}\mathbf{P}^\top$, we have $\mathbf{P}\mathbf{\Sigma}\mathbf{P}^\top$ as

$$\begin{bmatrix} s_1 & s_1\ell_1 & s_1\gamma_1 & \ldots & 0 \\ s_1\ell_1 & s_1\ell_1^2 + s_2 & s_1\ell_1\gamma_1 + s_2\ell_2 & \ldots & 0 \\ s_1\gamma_1 & s_1\ell_1\gamma_1 + s_2\ell_2 & s_1\gamma_1^2 + s_2\ell_2^2 + s_3 & \ldots & 0 \\ s_1\eta_1 & s_1\eta_1\ell_1 + s_2\gamma_2 & s_1\eta_1\gamma_1 + s_2\gamma_2\ell_2 + s_3\ell_3 & \ldots & 0 \\ 0 & s_2\eta_2 & s_2\eta_2\ell_2 + s_3\gamma_3 & \ldots & 0 \\ 0 & 0 & s_3\eta_3 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & s_{M-3}\eta_{M-3}^2 + s_{M-2}\gamma_{M-2}^2 \\ & & & & +s_{M-1}\gamma_{M-1}\ell_{M-1}^2 + s_M \end{bmatrix},$$

which is equivalent to matrix $\mathbf{Z}$. Through the above method of undetermined coefficients, we can solve the explicit value of all entries in matrix $\mathbf{P}, \mathbf{\Sigma}$, i.e., $\ell_1, \ldots, \ell_{M-1}$, $\gamma_1, \ldots, \gamma_{M-2}, \eta_1, \ldots, \eta_{M-3}$ in matrix $\mathbf{P}$ and $s_1, \ldots, s_M$ in matrix $\mathbf{\Sigma}$, which is summarized in algorithm 7. From algorithm 7, we can see that the computational complexity to decompose $\mathbf{Z}$ into $\mathbf{P}\mathbf{\Sigma}\mathbf{P}^\top$ is $O(M)$ with $n \in \{0, \ldots, N-1\}$ fixed.

Second, we verify the computational complexity to compute $(\mathbf{P}^\top)^{-1}\mathbf{\Sigma}^{-1}\mathbf{P}^{-1}\mathbf{y}$ is $O(M)$ with $n \in \{0, \ldots, N-1\}$ fixed and matrix $\mathbf{P}, \mathbf{\Sigma}$ available. To realize this objective, we will first verify that the computational complexity to calculate $\bar{\mathbf{y}} = \mathbf{P}^{-1}\mathbf{y}$ is $O(M)$. Then, we will first verify that the computational complexity to calculate $\bar{\bar{\mathbf{y}}} = \mathbf{\Sigma}^{-1}\bar{\mathbf{y}}$ is $O(M)$. Finally, we will first verify that the computational complexity to calculate $\bar{\bar{\bar{\mathbf{y}}}} = (\mathbf{P}^\top)^{-1}\bar{\bar{\mathbf{y}}}$ is $O(M)$. First of all, let us verify the computational complexity to compute $\bar{\mathbf{y}} = \mathbf{P}^{-1}\mathbf{y}$ is $O(M)$ with $n \in \{0, \ldots, N-1\}$ fixed. Because we

have a system of equations derived from $\mathbf{P}\bar{\mathbf{y}} = \mathbf{y}$:

$$\begin{cases} \bar{y}_1 &= y_1 \\ \bar{y}_2 &= y_2 - \ell_1\bar{y}_1 \\ \bar{y}_3 &= y_3 - \gamma_1\bar{y}_1 - \ell_2\bar{y}_2 \\ \bar{y}_4 &= y_4 - \eta_1\bar{y}_1 - \gamma_2\bar{y}_2 - \ell_3\bar{y}_3 \\ \bar{y}_5 &= y_5 - \eta_2\bar{y}_3 - \gamma_3\bar{y}_3 - \ell_4\bar{y}_4 \\ \vdots & \\ \bar{y}_M &= y_M - \eta_{M-3}\bar{y}_{M-3} - \gamma_{M-2}\bar{y}_{M-2} - \ell_{M-1}\bar{y}_{M-1} \end{cases},$$

we can solve vector $\bar{\mathbf{y}} = (\bar{y}_1, \bar{y}_2, \ldots, \bar{y}_M)^\top$ explicitly through algorithm 8, which only requires $O(M)$ computational complexity. After deriving $\bar{\mathbf{y}} = \mathbf{P}^{-1}\mathbf{y}$, we can easily verify that the computational complexity to derive $\bar{\bar{\mathbf{y}}} = \mathbf{\Sigma}^{-1}\bar{\mathbf{y}}$ is still $O(M)$ because $\boldsymbol{\sigma}$ is a diagonal matrix. Finally, after deriving $\bar{\bar{\mathbf{y}}} = \mathbf{\Sigma}^{-1}\bar{\mathbf{y}}$, we can verify that the computational complexity to derive $\bar{\bar{\bar{\mathbf{y}}}} = (\mathbf{P}^\top)^{-1}\bar{\bar{\mathbf{y}}}$ is still $O(M)$ with the similar logic as that in $\bar{\mathbf{y}} = \mathbf{P}^{-1}\mathbf{y}$.

From the above discussion, we know that the computational complexity to calculate $\widehat{\mathbf{f}} = (\widehat{u(x_0, t_n)}, \widehat{u(x_1, t_n)}, \ldots, \widehat{u(x_{M-1}, t_n)})^\top$, is $O(M)$ with $n \in \{0, 1, \ldots, N-1\}$ fixed. In other words, the computational complexity to derive $\left\{\widehat{u(x_i, t_n)}\right\}_{i=0,\ldots,M-1}$ is $O(M)$. According, the computational complexity to derive $\left\{\widehat{u(x_i, t_n)}\right\}_{i=0,\ldots,M-1,n=0,\ldots,N-1}$ is $O(MN)$.

Algorithm 7: Pseudo code to solve $\mathbf{P}, \mathbf{\Sigma}$

Input: matrix $\mathbf{Z}$
Output: matrix $\mathbf{P}, \mathbf{\Sigma}$
Initialize $s_j = \eta_j = \gamma_j = \ell_j = 0 \; \forall j \leq 0$
for $i = 1, 2, \ldots, M$ **do**
    $s_i = z_{ii} - s_{i-3}\eta_{i-3}^2 - s_{i-2}\gamma_{i-2}^2 - s_{i-1}\ell_{i-1}^2$
    $\ell_i = (z_{i,i+1} - s_{i-2}\gamma_{i-2}\eta_{i-2} - s_{i-1}\gamma_{i-1}\ell_{i-1})/s_i$
    $\eta_i = a_{i,i+3}/s_i$

Algorithm 8: Pseudo code to solve $\mathbf{P}^{-1}\mathbf{y}$

Input: matrix $\mathbf{P}, \mathbf{y}$
Output: vector $\bar{\mathbf{y}}$
Initialize $\eta_i = \gamma_i = \ell_i = 0 \ \forall i \leq 0$
for $i = 1, \ldots, M$ **do**
  $\bar{y}_i = y_i - \eta_{i-3}\bar{y}_{i-3} - \gamma_{i-2}\bar{y}_{i-2} - \ell_{i-1}\bar{y}_{i-1}$

□

Proof of Proposition 3.2.2

Proof. In subsection 3.6.1, we discuss how to use cubic spline to derive derivatives of $u(x,t)$. In this section, we discuss how to use local polynomial regression to derive derivatives, as a benchmark method.

Recall that the derivatives can be estimated by local polynomial regression includes $u(x_i,t_n), \frac{\partial}{\partial x}u(x_i,t_n), \frac{\partial^2}{\partial x^2}u(x_i,t_n), \ldots$. And here we take the derivation $\frac{\partial^l}{\partial x^l}u(x,t_n)$ as an example ($l = 0,1,2,\ldots$), and the other derivatives can be derived with the same logic flow. To derive the estimation of $\frac{\partial^l}{\partial x^l}u(x,t_n)$, we fix the temporal variable t_n for a general $n \in \{0,1,\ldots,N-1\}$. Then we locally fit a degree $\check{p}$ polynomial over the data $\{(x_i,u_i^n)\}_{i=0,\ldots,M-1}$, i.e.,

$$\begin{cases} u(x_0,t_n) &= u(x,t_n) + \frac{\partial}{\partial x}u(x,t_n)(x_0-x) + \ldots + \frac{\partial^{\check{p}}}{\partial x^{\check{p}}}u(x,t_n)(x_0-x)^{\check{p}} \\ u(x_1,t_n) &= u(x,t_n) + \frac{\partial}{\partial x}u(x,t_n)(x_1-x) + \ldots + \frac{\partial^{\check{p}}}{\partial x^{\check{p}}}u(x,t_n)(x_1-x)^{\check{p}} \\ \vdots & \vdots \\ u(x_{M-1},t_n) &= u(x,t_n) + \frac{\partial}{\partial x}u(x,t_n)(x_{M-1}-x) + \ldots + \frac{\partial^{\check{p}}}{\partial x^{\check{p}}}u(x,t_n)(x_{M-1}-x)^{\check{p}} \end{cases}.$$

For the choice of $\check{p}$, we choose $\check{p} = l+3$ to realize minmax efficiency [see 91]. If we denote $\mathbf{b}(x) = \left(u(x,t_n), \frac{\partial}{\partial x}u(x,t_n), \ldots, \frac{\partial^{\check{p}}}{\partial x^{\check{p}}}u(x,t_n)\right)^\top$, then $\frac{\partial^l}{\partial x^l}u(x,t_n)$ can be obtained as the $(l+1)$-th entry of the vector $\widehat{\mathbf{b}(x)}$, and $\widehat{\mathbf{b}(x)}$ is obtained by the following optimization

problem:

$$\widehat{\mathbf{b}(x)} = \arg\min_{\mathbf{b}(x)} \sum_{i=0}^{M-1} \left[u_i^n - \sum_{j=0}^{\check{p}} \frac{\partial^j}{\partial x^j} u(x, t_n)(x_i - x)^j \right]^2 \mathcal{K}\left(\frac{x_i - x}{h} \right), \tag{3.23}$$

where h is the bandwidth parameter, and $\mathcal{K}$ is a kernel function, and in our paper, we use the Epanechikov kernel $\mathcal{K}(x) = \frac{3}{4}\max\{0, 1 - x^2\}$ for $x \in \mathbb{R}$. Essentially, the optimization problem in Equation 3.23 is a weighted least squares model, where $\mathbf{b}(x)$ can be solved in a close form:

$$\mathbf{b}(x) = \left(\mathbf{X}_{\text{spa}}^\top \mathbf{W}_{\text{spa}} \mathbf{X}_{\text{spa}} \right)^{-1} \mathbf{X}_{\text{spa}}^\top \mathbf{W}_{\text{spa}} \mathbf{u}_{:}^n, \tag{3.24}$$

where

$$\mathbf{X}_{\text{spa}} = \begin{bmatrix} 1 & (x_0 - x) & \dots & (x_0 - x)^{\check{p}} \\ 1 & (x_1 - x) & \dots & (x_1 - x)^{\check{p}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & (x_{M-1} - x) & \dots & (x_{M-1} - x)^{\check{p}} \end{bmatrix}, \mathbf{u}_{:}^n = \begin{bmatrix} u_0^n \\ u_1^n \\ \vdots \\ u_{M-1}^n \end{bmatrix}$$

and $\mathbf{W}_{\text{spa}} = \text{diag}\left(\mathcal{K}\left(\frac{x_0 - x}{h}\right), \dots, \mathcal{K}\left(\frac{x_{M-1} - x}{h}\right) \right)$.

By implementing the local polynomial in this way, the computational complexity is much higher than our method, and we summarize its computational complexity in the following proposition.

Following please find the proof.

Similar to the proof of the computational complexity in cubic spline, the proof of the computational complexity of local polynomial regression in the funcational estimation stage lies in calculating all elements in matrix $\mathbf{X}$ and vector $\nabla_t \mathbf{u}$, including

$$\left\{ \widehat{u(x_i, t_n)}, \widehat{\frac{\partial}{\partial x} u(x_i, t_n)}, \widehat{\frac{\partial^2}{\partial x^2} u(x_i, t_n)}, \widehat{\frac{\partial}{\partial t} u(x_i, t_n)} \right\}_{i=0,\dots,M-1, n=0,\dots,N-1}.$$

We will take the estimation of $\widehat{\frac{\partial^p}{\partial x^p} u(x_i, t_n)}$ with a general $p \in \mathbb{N}$ as an example. To solve $\left\{ \widehat{\frac{\partial^p}{\partial x^p} u(x_i, t_n)} \right\}_{i=0,\dots,M-1, n=0,\dots,N-1}$, we first focus on $\left\{ \widehat{\frac{\partial^p}{\partial x^p} u(x_i, t_n)} \right\}_{i=0,\dots,M-1}$, with

$n \in \{0, \ldots, N-1\}$ fixed. To solve it, the main idea of local polynomial regression is to do Taylor expansion:

$$\begin{cases} u(x_0, t_n) &= u(x, t_n) + \frac{\partial}{\partial x} u(x, t_n)(x_0 - x) + \ldots + \frac{\partial^{\check{p}}}{\partial x^{\check{p}}} u(x, t_n)(x_0 - x)^{\check{p}} \\ u(x_1, t_n) &= u(x, t_n) + \frac{\partial}{\partial x} u(x, t_n)(x_1 - x) + \ldots + \frac{\partial^{\check{p}}}{\partial x^{\check{p}}} u(x, t_n)(x_1 - x)^{\check{p}} \\ \vdots & \vdots \\ u(x_{M-1}, t_n) &= u(x, t_n) + \frac{\partial}{\partial x} u(x, t_n)(x_{M-1} - x) + \ldots + \frac{\partial^{\check{p}}}{\partial x^{\check{p}}} u(x, t_n)(x_{M-1} - x)^{\check{p}} \end{cases},$$

where $\check{p}$ is usually set as $\check{p} = p + 3$ to obtain asymptotic minimax efficiency [see 91]. In the above system of equations, if we denote

$$\mathbf{b}(x) = \left(u(x, t_n), \frac{\partial}{\partial x} u(x, t_n), \ldots, \frac{\partial^{\check{p}}}{\partial x^{\check{p}}} u(x, t_n) \right)^\top,$$

then we can solve $\mathbf{b}(x)$ through the optimization problem in Equation 3.23 with a closed-form solution shown in Equation 3.24:

$$\mathbf{b}(x) = \left(\mathbf{X}_{\text{spa}}^\top \mathbf{W}_{\text{spa}} \mathbf{X}_{\text{spa}} \right)^{-1} \mathbf{X}_{\text{spa}}^\top \mathbf{W}_{\text{spa}} \mathbf{u}_{:}^{n}, \tag{3.25}$$

where

$$\mathbf{X}_{\text{spa}} = \begin{bmatrix} 1 & (x_0 - x) & \ldots & (x_0 - x)^{\check{p}} \\ 1 & (x_1 - x) & \ldots & (x_1 - x)^{\check{p}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & (x_{M-1} - x) & \ldots & (x_{M-1} - x)^{\check{p}} \end{bmatrix}, \mathbf{u}_{:}^{n} = \begin{bmatrix} u_0^n \\ u_1^n \\ \vdots \\ u_{M-1}^n \end{bmatrix}$$

and $\mathbf{W}_{\text{spa}} = \text{diag}\left(\mathcal{K}\left(\frac{x_0 - x}{h}\right), \ldots, \mathcal{K}\left(\frac{x_{M-1} - x}{h}\right) \right)$.

The main computational complexity to derive $\mathbf{b}(x)$ lies in the computation of inverse

of matrix $\mathbf{X}_{\text{spa}}^{\top}\mathbf{W}_{\text{spa}}\mathbf{X}_{\text{spa}} \in \mathbb{R}^{(\check{p}+1)\times(\check{p}+1)}$, where

$$\mathbf{X}_{\text{spa}}^{\top}\mathbf{W}_{\text{spa}}\mathbf{X}_{\text{spa}} = \begin{bmatrix} \sum_{i=0}^{M-1} w_i & \sum_{i=0}^{M-1} w_i(x_i-x) & \sum_{i=0}^{M-1} w_i(x_i-x)^2 & \dots & \sum_{i=0}^{M-1} w_i(x_i-x)^{\check{p}} \\ \sum_{i=0}^{M-1} w_i(x_i-x) & \sum_{i=0}^{M-1} w_i(x_i-x)^2 & \sum_{i=0}^{M-1} w_i(x_i-x)^3 & \dots & \sum_{i=0}^{M-1} w_i(x_i-x)^{\check{p}+1} \\ \sum_{i=0}^{M-1} w_i(x_i-x)^2 & \sum_{i=0}^{M-1} w_i(x_i-x)^3 & \sum_{i=0}^{M-1} w_i(x_i-x)^4 & \dots & \sum_{i=0}^{M-1} w_i(x_i-x)^{\check{p}+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum_{i=0}^{M-1} w_i(x_i-x)^{\check{p}} & \sum_{i=0}^{M-1} w_i(x_i-x)^{\check{p}+1} & \sum_{i=0}^{M-1} w_i(x_i-x)^{\check{p}+2} & \dots & \sum_{i=0}^{M-1} w_i(x_i-x)^{2\check{p}} \end{bmatrix}$$

and

$$\mathbf{X}_{\text{spa}}^{\top}\mathbf{W}_{\text{spa}}\mathbf{u}_{:}^{n} = \begin{pmatrix} \sum_{i=0}^{M-1} w_i u_i^n \\ \sum_{i=0}^{M-1} w_i(x_i-x)u_i^n \\ \sum_{i=0}^{M-1} w_i(x_i-x)^2 u_i^n \\ \sum_{i=0}^{M-1} w_i(x_i-x)^3 u_i^n \\ \sum_{i=0}^{M-1} w_i(x_i-x)^4 u_i^n \end{pmatrix},$$

we know that for a fixed $n \in \{0,\dots,N-1\}$ and $x \in \{x_0,\dots,x_{M-1}\}$, the computational complexity of computing $\mathbf{X}_{\text{spa}}^{\top}\mathbf{W}_{\text{spa}}\mathbf{X}_{\text{spa}}$ and $\mathbf{X}_{\text{spa}}^{\top}\mathbf{W}_{\text{spa}}\mathbf{u}_{:}^{n}$ is $O(\check{p}^2M)$. Besides, the computational complexity to derive $(\mathbf{X}_{\text{spa}}^{\top}\mathbf{W}_{\text{spa}}\mathbf{X}_{\text{spa}})^{-1}$ is $O(\check{p}^3)$. So we know that for a fixed $n \in \{0,\dots,N-1\}$ and $x \in \{x_0,\dots,x_{M-1}\}$, the computational complexity of computing $\frac{\partial^p}{\partial x^p}u(x_i,t_n)$ is $\max\{O(\check{p}^2M), O(\check{p}^3)\}$ with $\check{p}$ usually set as $\check{p} = p+3$. Accordingly, the computational complexity of computing $\left\{\frac{\partial^p}{\partial x^p}u(x_i,t_n)\right\}_{i=0,\dots,M-1,n=0,\dots,N-1}$ is $\max\{O(\check{p}^2M^2N), O(\check{p}^3MN)\}$. Because $p \le q_{\max}$, we know that the computational complexity of computing all derivatives with respective to x with highest order as $q_{\max}$ is $\max\{O(q_{\max}^2M^2N), O(q_{\max}^3MN)\}$. Similarly, the computational complexity of computing the first order derivatives with respective to t is $\max\{O(MN^2), O(MN)\}$. In conclusion,

the computational complexity to derive all elements in matrix $\mathbf{X}$ and vector $\nabla_t \mathbf{u}$, including

$$\left\{ \widehat{u(x_i,t_n)}, \widehat{\frac{\partial}{\partial x}u(x_i,t_n)}, \widehat{\frac{\partial^2}{\partial x^2}u(x_i,t_n)}, \widehat{\frac{\partial}{\partial t}u(x_i,t_n)} \right\}_{i=0,\ldots,M-1,n=0,\ldots,N-1}.$$

by local polynomial regression in Equation 3.4 is $\max\{O(q_{\max}^2 M^2 N), O(MN^2), O(q_{\max}^3 MN)\}$, where $q_{\max}$ is the highest order of derivatives desired in Equation 3.3. □

Proof of Lemma 3.6.1

Proof. In this proof, we take $k = 0$ as an illustration example, i.e., prove that when

$$\begin{aligned}\epsilon \;>\; \mathscr{C}(\sigma, \|u\|_{L^\infty(\Omega)}) \max \;& \left\{ \tfrac{4K_{\max}}{M^{3/7}}, 4AM^{-3/7}, 4\sqrt{2}\tfrac{d^3}{dx^3}f^*(0)M^{-3/7}, \tfrac{16(C\log M+\gamma)\log(M)}{M^{3/7}}, \right. \\ & \left. 16\sqrt{\tfrac{\omega}{7}}\widetilde{C}(\sigma, \|u\|_{L^\infty(\Omega)})\tfrac{\sqrt{\log(M)}}{M^{3/7}} \right\},\end{aligned}$$

we have

$$P\left[\sup_{x\in[0,X_{\max}]} \left| \widehat{u(x,t_n)} - u(x,t_n) \right| > \epsilon \right] < 2Me^{-\frac{M^{2/7}}{2\sigma^2}} + Qe^{-L\gamma} + 4\sqrt{2}\eta^4 M^{-\frac{2}{7}\omega}$$

for a fixed t_n with $n \in \{0, 1, \ldots, N-1\}$. For $k = 1, 2$, it can be derived with the same logic flow.

Recall in subsection 3.2.1, the fitted value of the smoothing cubic spline $s(x)$ is the minimizer of the optimization problem in Equation 3.4. From Theorem A in [89] (also mentioned by [92] in the Section 1, and equation (2.2) in [93]) that when Condition 3.3.4 - Condition 3.3.5 hold and for large M and small $\widetilde{\lambda} = \frac{1-\alpha}{\alpha}$, we have

$$\widehat{f_i} = \frac{1}{M\widetilde{\lambda}^{1/4}} \sum_{j=0}^{M-1} K\left(\frac{x_i - x_j}{\widetilde{\lambda}^{1/4}} \right) u_j^n,$$

where $\widehat{f_i} = \widehat{u(x_i,t_n)}$, $\widetilde{\lambda}$ trades off the goodness-of-fit and smoothness of the cubic spline in

Equation 3.4 and $K(\cdot)$ is a fixed kernel function defined as

$$K(x) = \frac{1}{2} e^{-|x|/\sqrt{2}} \left[\sin(|x|/\sqrt{2} + \pi/4) \right].$$

For a general spatial variable x and fixed $n \in \{0, 1, \ldots, N-1\}$, we denote

$$f^*(x) = u(x, t_n),$$

which is the ground truth of the underlying dynamic function $u(x, t_n)$ with t_n fixed. Besides, we denote $\widehat{f}(x) = \widehat{u(x, t_n)}$, which is an estimation of the ground truth of $f^*(x) = u(x, t_n)$ with t_n fixed. Accordingly to the above discussion, this estimation of $\widehat{f}(x)$ can be written as

$$\widehat{f}(x) = \frac{1}{M\widetilde{\lambda}^{1/4}} \sum_{j=0}^{M-1} K\left(\frac{x - x_j}{\widetilde{\lambda}^{1/4}} \right) u_j^n,$$

where $\widehat{f_i} = \widehat{f}(x_i)$ for $i \in \{0, 1, \ldots, M-1\}$

In order to bound $P\left(\sup |\widehat{f}(x) - f^*(x)| > \epsilon\right)$ for a general x, we decompose it as follows:

$$\begin{aligned}
& P\left(\sup |\widehat{f}(x) - f^*(x)| > \epsilon\right) \\
= \; & P\left(\sup |\widehat{f}(x) - \widehat{f}^B(x) + \widehat{f}^B(x) - \widehat{f}^*(x)| > \epsilon\right) \\
= \; & P\Big(\sup |\widehat{f}(x) - \widehat{f}^B(x) - E(\widehat{f}(x) - \widehat{f}^B(x)) + E(\widehat{f}(x) - \widehat{f}^B(x)) + \\
& \widehat{f}^B(x) - \widehat{f}^*(x)| > \epsilon\Big) \\
= \; & P\Bigg(\sup |\underbrace{\widehat{f}(x) - \widehat{f}^B(x)}_{\mathcal{A}} - \underbrace{E(\widehat{f}(x) - \widehat{f}^B(x))}_{\mathcal{B}} + \\
& \underbrace{E(\widehat{f}(x)) - f^*(x)}_{\mathcal{C}} + \underbrace{\widehat{f}^B(x) - E(\widehat{f}^B(x))}_{\mathcal{D}}| > \epsilon\Bigg) \\
\leq \; & P\left(\sup |\mathcal{A}| > \tfrac{\epsilon}{4}\right) + P\left(\sup |\mathcal{B}| > \tfrac{\epsilon}{4}\right) + P\left(\sup |\mathcal{C}| > \tfrac{\epsilon}{4}\right) + P\left(\sup |\mathcal{D}| > \tfrac{\epsilon}{4}\right)
\end{aligned} \tag{3.26}$$

where the $\widehat{f^B}(x)$ in (Equation 3.26) the truncated estimator defined as

$$\widehat{f^B}(x) = \frac{1}{M\widetilde{\lambda}^{1/4}} \sum_{j=0}^{M-1} K\left(\frac{x - x_j}{\widetilde{\lambda}^{1/4}}\right) u_j^n \mathbb{1}\{u_j^n < B_M\}.$$

Here $\{B_M\}$ is an increasing sequence and $B_M \to +\infty$ as $M \to +\infty$, i.e., $B_M = M^b$ with constant $b > 0$, and we will discuss the value of b at the end of this proof.

In the remaining of the proof, we work on the upper bound of the four decomposed terms, i.e., $P\left(\sup|\mathcal{A}| > \frac{\epsilon}{4}\right), P\left(\sup|\mathcal{B}| > \frac{\epsilon}{4}\right), P\left(\sup|\mathcal{C}| > \frac{\epsilon}{4}\right), P\left(\sup|\mathcal{D}| > \frac{\epsilon}{4}\right)$.

First, let us discuss the upper bound of $P\left(\sup|\mathcal{A}| > \frac{\epsilon}{4}\right)$.

Because

$$\begin{aligned} P\left(\sup|\mathcal{A}| > \frac{\epsilon}{4}\right) &= P\left(\sup\left|\widehat{f}(x) - \widehat{f^B}(x)\right| > \frac{\epsilon}{4}\right) \\ &= P\left(\sup\left|\frac{1}{M\widetilde{\lambda}^{1/4}} \sum_{j=0}^{M-1} K\left(\frac{x - x_j}{\widetilde{\lambda}^{1/4}}\right) u_j^n \mathbb{1}\{u_j^n \geq B_M\}\right| > \frac{\epsilon}{4}\right) \\ &\leq P\left(\sup\left|\frac{K_{\max}}{M\widetilde{\lambda}^{1/4}} \sum_{j=0}^{M-1} u_j^n \mathbb{1}\{u_j^n \geq B_M\}\right| > \frac{\epsilon}{4}\right), \end{aligned}$$

where $K_{\max} = \max_{x \in [0, X_{\max}] \cup [0, T_{\max}]} K(x)$. If we let $\frac{\epsilon}{4} > \frac{K_{\max}}{M\widetilde{\lambda}^{1/4}} B_M$, then we have

$$\begin{aligned} P\left(\sup|\mathcal{A}| > \frac{\epsilon}{4}\right) &\leq P\left(\exists\, i = 0, \ldots, M-1, s.t.\ |u_i^n| \geq B_M\right) \\ &= P\left(\max_{i=0,\ldots,M-1} |u_i^n| \geq B_M\right) \end{aligned}$$

Let $C_M = B_M - \|U\|_{L^\infty(\Omega)}$, where U is the random variable generated from the unknown

dynamic system, i.e., $U = u(x,t) + \epsilon$ with $\epsilon \sim N(0, \sigma^2)$. Then we have

$$\begin{aligned} P\left(\sup|\mathcal{A}| > \frac{\epsilon}{4}\right) &= P\left(\sup\left|\widehat{f}(x) - \widehat{f}^B(x)\right| > \frac{\epsilon}{4}\right) \\ &\leq P\left(\max_{i=0,\ldots,M-1}|U_i^n - u_i^n| \geq C_M\right) \\ &\leq 2Me^{-C_M^2/(2\sigma^2)} \end{aligned}$$

Next, let us discuss the upper bound of $P\left(\sup|\mathcal{B}| > \frac{\epsilon}{4}\right)$.

$$\begin{aligned} \mathcal{B} &= E\left(|\widehat{f}(x) - \widehat{f}^B(x)|\right) \\ &= E\left(\left|\frac{1}{M\widetilde{\lambda}^{1/4}}\sum_{j=0}^{M-1} K\left(\frac{x-x_j}{\widetilde{\lambda}^{1/4}}\right) u_j^n \mathbb{1}\{u_j^n \geq B_M\}\right|\right) \\ &\leq E\left(\frac{1}{M\widetilde{\lambda}^{1/4}}\sum_{j=0}^{M-1}\left|K\left(\frac{x-x_j}{\widetilde{\lambda}^{1/4}}\right)\right| |u_j^n| \mathbb{1}\{u_j^n \geq B_M\}\right) \\ &= \frac{1}{\widetilde{\lambda}^{1/4}}\int\int_{|u|\geq B_M}\left|K\left(\frac{x-a}{\widetilde{\lambda}^{1/4}}\right)\right| |u| dF_M(a,u) \qquad (3.27) \\ &\leq \int |K(\xi)|d\xi \times \underbrace{\sup_\alpha \int_{|u|\geq B_M} |u| f_M(\alpha,u)du}_{\mathcal{V}} \qquad (3.28) \end{aligned}$$

Here in Equation 3.27, $F_M(\cdot,\cdot)$ is the empirical c.d.f. of (x,u)'s, and in Equation 3.28, $f_M(\cdot,\cdot)$ is the empirical p.d.f. of (x,u)'s.

Now let us take a look at the upper bound of $\mathcal{V}$. For any $s > 0$, we have

$$\begin{aligned} \sup_\alpha \int_{|u|\geq B_M} \frac{|u|}{B_M} f_M(\alpha,u)du &\leq \sup_\alpha \int_{|u|\geq B_M}\left(\frac{|u|}{B_M}\right)^s f_M(\alpha,u)du \\ &\leq \sup_\alpha \int \left(\frac{|u|}{B_M}\right)^s f_M(\alpha,u)du, \end{aligned}$$

which gives

$$\mathcal{V} := \sup_{\alpha} \int_{|u| \geq B_M} |u| f_M(\alpha, u) du \leq B_M^{1-s} \underbrace{\sup_{\alpha} \int |u|^s f_M(\alpha, u) du}_{\pi_s}.$$

From the lemma statement we know that when $s = 2$, we have $\pi_s := \sup_\alpha \int |u|^s f_M(\alpha, u) du < +\infty$. If we set $A = \pi_s \int |K(\xi)| d\xi$, then we have

$$\mathcal{B} \leq A B_M^{1-s}$$

So when $\frac{\epsilon}{4} > A B_M^{1-s}$, we have

$$P\left(\sup |\mathcal{B}| > \frac{\epsilon}{4}\right) = P\left(E\left(|\widehat{f}(x) - \widehat{f}^B(x)|\right) \geq \frac{\epsilon}{4}\right) = 0.$$

Then, let us discuss the upper bound of $P\left(\sup |\mathcal{C}| > \frac{\epsilon}{4}\right)$. According to Lemma 5 in [94], when

$f^*(x) \in C^4$, $\frac{d^2}{dx^2} f^*(x_0) = \frac{d^2}{dx^2} f^*(x_{M-1}) = 0$ and $\frac{d^3}{dx^3} f^*(x_0) \neq 0$, $\frac{d^3}{dx^3} f^*(x_{M-1}) = 0$, we have

$$\begin{aligned} & E(\widehat{f}(x)) - f^*(x) \\ = \; & \sqrt{2} \frac{d^3}{dx^3} f^*(0) \widetilde{\lambda}^{3/4} \exp\left(\frac{-x}{\sqrt{2}} \widetilde{\lambda}^{-1/4}\right) \cos\left(\frac{x}{\sqrt{2}} \widetilde{\lambda}^{-1/4}\right) + \ell(x), \end{aligned}$$

where the error term $\ell(x)$ satisfies

$$\int [\ell(x)]^2 dx = o\left(\int \left[E(\widehat{f}(x)) - f^*(x)\right]^2 dx\right).$$

So when $\frac{\epsilon}{4} > \sqrt{2}\frac{d^3}{dx^3}f^*(0)\widetilde{\lambda}^{3/4}$ and M is sufficiently large then we have

$$P\left(\sup|\mathcal{C}| > \frac{\epsilon}{4}\right) = 0$$

Finally, let us discuss the upper bound of $P\left(\sup|\mathcal{D}| > \frac{\epsilon}{4}\right)$.

In order to bound $P\left(\sup|D| > \frac{\epsilon}{4}\right)$, we further decompose $\mathcal{D}$ into two components, i.e.,

$$\mathcal{D} := \widehat{f}^B(x) - E(\widehat{f}^B(x)) = e_M(x,t_n) + \frac{1}{\sqrt{M}}\rho_M(x,t_n).$$

The decomposition procedure and the definition of $e_M(x,t_n), \rho_M(x,t_n)$ are described in the following system of equations [see 95, Proposition 2]:

$$\begin{aligned}
\mathcal{D} &= \widehat{f}^B(x) - E(\widehat{f}^B(x)) \\
&= \frac{1}{M\widetilde{\lambda}^{1/4}}\sum_{j=0}^{M-1} K\left(\frac{x-x_j}{\widetilde{\lambda}^{1/4}}\right) u_j^n \mathbb{1}\{u_j^n < B_M\} - \\
&\quad E\left(\frac{1}{M\widetilde{\lambda}^{1/4}}\sum_{j=0}^{M-1} K\left(\frac{x-x_j}{\widetilde{\lambda}^{1/4}}\right) u_j^n \mathbb{1}\{u_j^n < B_M\}\right) \\
&= \frac{1}{\sqrt{M}\widetilde{\lambda}^{1/4}}\int_{a\in\mathbb{R}}\int_{|u|<B_M} K\left(\frac{x-a}{\widetilde{\lambda}^{1/4}}\right) u \; d\underbrace{\left(\sqrt{M}(F_M(a,u) - F(a,u))\right)}_{Z_M(a,u)} \quad (3.29) \\
&= \frac{1}{\sqrt{M}\widetilde{\lambda}^{1/4}}\int_{a\in\mathbb{R}} K\left(\frac{x-a}{\widetilde{\lambda}^{1/4}}\right)\int_{|u|<B_M} u \; d(Z_M(a,u)) \\
&= \frac{1}{\sqrt{M}\widetilde{\lambda}^{1/4}}\int_{a\in\mathbb{R}} K\left(\frac{x-a}{\widetilde{\lambda}^{1/4}}\right)\left[\int_{|u|<B_M} u \; d(Z_M(a,u) - B_0(T(a,u))) + \right. \\
&\quad \left.\int_{|u|<B_M} u \; dB_0(T(a,u))\right] \quad (3.30) \\
&= \underbrace{\frac{1}{\sqrt{M}\widetilde{\lambda}^{1/4}}\int_{a\in\mathbb{R}}\int_{|u|<B_M} K\left(\frac{x-a}{\widetilde{\lambda}^{1/4}}\right) u \; d(Z_M(a,u) - B_0(T(a,u)))}_{e_M(x,t_n)} \\
&\quad + \frac{1}{\sqrt{M}}\underbrace{\frac{1}{\widetilde{\lambda}^{1/4}}\int_{a\in\mathbb{R}}\int_{|u|<B_M} K\left(\frac{x-a}{\widetilde{\lambda}^{1/4}}\right) u \; dB_0(T(a,u))}_{\rho_M(x,t_n)}.
\end{aligned}$$

In Equation 3.29, $F_M(\cdot,\cdot) := F_M(\cdot,\cdot|t_n)$ is the empirical c.d.f of (x,u) with a fixed t_n, and $Z_M(a,u) = \sqrt{M}(F_M(a,u) - F(a,u))$ is a two-dimensional empirical process [see 96, 95]. In Equation 3.30, $B_0(T(a,u))$ is a sample path of two-dimensional Brownian bride. And $T(a,u) : \mathbb{R}^2 \to [0,1]^2$ is the transformation defined by [97], i.e., $T(a,u) = (F_A(x), F_{U|A}(u|a))$, where F_A is the marginal c.d.f of A and $F_{U|A}$ is the conditional c.d.f of U given A [see 95, Proposition 2].

Through the above decomposition of $\mathcal{D}$, we have

$$P\left(\sup|D| > \frac{\epsilon}{4}\right) \leq P\left(\sup|e_M(x,t_n)| > \frac{\epsilon}{8}\right) + P\left(\sup\frac{1}{\sqrt{M}}|\rho_M(x,t_n)| > \frac{\epsilon}{8}\right).$$

For $e_M(x,t_n)$, we have

$$\begin{aligned}
& P\left(\sup|e_M(x,t_n)| > \frac{\epsilon}{8}\right) \\
= \; & P\left(\sup\left|\frac{1}{\sqrt{M}\widetilde{\lambda}^{1/4}}\int_{a\in\mathbb{R}}\int_{|u|<B_M} K\left(\frac{x-a}{\widetilde{\lambda}^{1/4}}\right) u \; d(Z_M(a,u) - B_0(T(a,u)))\right| > \frac{\epsilon}{8}\right) \\
\leq \; & P\left(\frac{2B_M K_{\max}}{\sqrt{M}\widetilde{\lambda}^{1/4}}\sup_{a,u}|Z_M(a,u) - B_0(T(a,u))| > \frac{\epsilon}{8}\right).
\end{aligned}$$

Proved by Theorem 1 in [96], we know that, for any γ, we have

$$P\left(\sup_{a,u}|Z_M(a,u) - B_0(T(a,u))| > \frac{(C\log M + \gamma)\log M}{\sqrt{M}}\right) \leq Qe^{-L\gamma},$$

where C, Q, L are absolute positive constants which is independent of temporal resolution N and spatial resolution M. Thus, when $\frac{\epsilon}{8} \geq \frac{2B_M K_{\max}}{\sqrt{M}\widetilde{\lambda}^{1/4}}\frac{(C\log M+\gamma)\log M}{\sqrt{M}}$, we have

$$P\left(\sup|e_M(x,t_n)| > \frac{\epsilon}{8}\right) < Qe^{-L\gamma}.$$

For $\rho_M(x,t_n)$, by equation (7) in [95], we have

$$\begin{aligned}\frac{\widetilde{\lambda}^{1/8}\sup|\rho_M(x,t_n)|}{\sqrt{\log(1/\widetilde{\lambda}^{1/4})}} \leq & \underbrace{16(\log V)^{1/2}S^{1/2}\left(\log\left(\frac{1}{\widetilde{\lambda}^{1/4}}\right)\right)^{-1/2}\int|\xi|^{1/2}|dK(\xi)|}_{\mathcal{W}_{1,M}}+ \\ & \underbrace{16\sqrt{2}\widetilde{\lambda}^{-1/8}\left(\log\left(\frac{1}{\widetilde{\lambda}^{1/4}}\right)\right)^{-1/2}\int q(S\widetilde{\lambda}^{1/4}|\tau|)|d(K(\tau))|}_{\mathcal{W}_{2,M}},\end{aligned}$$

where V is a random variable satisfying $E(V)\leq 4\sqrt{2}\eta^4$, with $\eta^2=\max_{i=0,\ldots,M-1,n=0,\ldots,N-1}$ $E(U_i^n)^2$, $S=\sup_x\int u^2 f(x,u)du$ with $f(\cdot,\cdot)$ as the distribution function of (x_i,u_i^n), and $q(z)=\int_0^z\frac{1}{2}\sqrt{\frac{1}{y}\log\left(\frac{1}{y}\right)}dy$. So we have the following system of equations:

$$\begin{aligned}& P\left(\sup\frac{1}{\sqrt{M}}|\rho_M(x,t_n)|>\frac{\epsilon}{8}\right) \\ = & P\left(\frac{\widetilde{\lambda}^{1/8}\sup|\rho_M(x,t_n)|}{\sqrt{\log(1/\widetilde{\lambda}^{1/4})}}>\frac{\sqrt{M}\widetilde{\lambda}^{1/8}\epsilon}{8\sqrt{\log(1/\widetilde{\lambda}^{1/4})}}\right) \\ \leq & P\left(\mathcal{W}_{1,M}+\mathcal{W}_{2,M}>\frac{\sqrt{M}\widetilde{\lambda}^{1/8}\epsilon}{8\sqrt{\log(1/\widetilde{\lambda}^{1/4})}}\right) \\ \leq & P\left(\mathcal{W}_{1,M}\geq\frac{\sqrt{M}\widetilde{\lambda}^{1/8}\epsilon}{16\sqrt{\log(1/\widetilde{\lambda}^{1/4})}}\right)+P\left(\mathcal{W}_{2,M}\geq\frac{\sqrt{M}\widetilde{\lambda}^{1/8}\epsilon}{16\sqrt{\log(1/\widetilde{\lambda}^{1/4})}}\right) \quad (3.31)\end{aligned}$$

Now let us bound $P\left(\mathcal{W}_{1,M}\geq\frac{\sqrt{M}\widetilde{\lambda}^{1/8}\epsilon}{16\sqrt{\log(1/\widetilde{\lambda}^{1/4})}}\right), P\left(\mathcal{W}_{2,M}\geq\frac{\sqrt{M}\widetilde{\lambda}^{1/8}\epsilon}{16\sqrt{\log(1/\widetilde{\lambda}^{1/4})}}\right)$ in Equation 3.31 separately.

1. For the first term in Equation 3.31, we have

$$
\begin{aligned}
& P\left(\mathcal{W}_{1,M} \geq \frac{\sqrt{M}\widetilde{\lambda}^{1/8}\epsilon}{16\sqrt{\log(1/\widetilde{\lambda}^{1/4})}}\right) \\
= \; & P\left(16(\log V)^{1/2} S^{1/2}\left(\log\left(\frac{1}{\widetilde{\lambda}^{1/4}}\right)\right)^{-1/2}\int |\xi|^{1/2}|dK(\xi)| \geq \frac{\sqrt{M}\widetilde{\lambda}^{1/8}\epsilon}{16\sqrt{\log(1/\widetilde{\lambda}^{1/4})}}\right) \\
= \; & P\left((\log V)^{1/2} \geq \frac{\sqrt{M}\widetilde{\lambda}^{1/8}\epsilon}{16^2 S^{1/2}\int |\xi|^{1/2}|dK(\xi)|}\right) \\
= \; & P\left(\log V \geq \left(\frac{\sqrt{M}\widetilde{\lambda}^{1/8}\epsilon}{16^2 S^{1/2}\int |\xi|^{1/2}|dK(\xi)|}\right)^2\right) \\
= \; & P\left(V \geq \exp\left[\left(\frac{\sqrt{M}\widetilde{\lambda}^{1/8}\epsilon}{16^2 S^{1/2}\int |\xi|^{1/2}|dK(\xi)|}\right)^2\right]\right) \\
\leq \; & \frac{E(V)}{\exp\left[\left(\frac{\sqrt{M}\widetilde{\lambda}^{1/8}\epsilon}{16^2 S^{1/2}\int |\xi|^{1/2}|dK(\xi)|}\right)^2\right]} \qquad (3.32) \\
\leq \; & \frac{4\sqrt{2}\eta^4}{\exp\left[\left(\frac{\sqrt{M}\widetilde{\lambda}^{1/8}\epsilon}{16^2 S^{1/2}\int |\xi|^{1/2}|dK(\xi)|}\right)^2\right]} \qquad (3.33) \\
= \; & 4\sqrt{2}\eta^4\widetilde{\lambda}^{\omega/4} \qquad (3.34)
\end{aligned}
$$

Here inequality Equation 3.32 is due to Markov's inequality, and inequality Equation 3.33 is due to the fact that $E(V) \leq 4\sqrt{2}\eta^4$. Equality Equation 3.34 is because we set $\frac{\sqrt{M}\widetilde{\lambda}^{1/8}\epsilon}{16\sqrt{\log(1/\widetilde{\lambda}^{1/4})}} = \sqrt{\omega}\widetilde{C}(t_n, \sigma, \|u\|_{L^\infty}(\Omega))$, where

$$\widetilde{C}(t_n, \sigma, \|u\|_{L^\infty}(\Omega)) := 16\sqrt{S}\int |\xi|^{1/2}|dK(\xi)|$$

and $\omega > 1$ is an arbitrary scaler.

2. For the second term of Equation 3.31, it converges to $\widetilde{C}(t_n, \sigma, \|u\|_{L^\infty}(\Omega))$ by using arguments similar to [98] (page. 180-181) under the condition in Lemma 3.6.1 that $\int \sqrt{|x\log(|x|)|}|dK(x)| < +\infty$. Here we add $(t_n, \sigma, \|u\|_{L^\infty(\Omega)})$ after $\bar{C}$ to emphasize

that the constant $\bar{C}(t_n, \sigma, \|u\|_{L^\infty(\Omega)})$ is dependent on $t_n, \sigma, \|u\|_{L^\infty(\Omega)}$.

It should be noted that

$$\widetilde{C}(t_n, \sigma, \|u\|_{L^\infty(\Omega)}) < +\infty, \tag{3.35}$$

given the reasons listed as follows. First, it can be easily verified that the term $\int |\xi|^{1/2} |dK(\xi)|$ in $\widetilde{C}(t_n, \sigma, \|u\|_{L^\infty}(\Omega))$ is bounded. Second, for $S = \sup_x \int u^2 f(x,u) du$, it is also bounded. The reasons are described as follows. For a general $\varrho > 0$, we have

$$\begin{aligned}
& \sup_{x\in[0,X_{\max}]} \int |u|^\varrho f(x,u) du \\
= & \sup_{x\in[0,X_{\max}]} \int |u|^\varrho \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(u-u(x,t_n))^2}{2\sigma^2}\right) du \\
= & \sup_{x\in[0,X_{\max}]} \frac{1}{\sqrt{2}} \sigma^2 2^{\varrho/2} \Gamma\left(\frac{1+\varrho}{2}\right) G\left(-\frac{\varrho}{2}, \frac{1}{2}, -\frac{1}{2}\left(\frac{u(x,t_n)}{\sigma}\right)^2\right),
\end{aligned}$$

where $G(a,b,z)$ is Kummer's confluent hypergeometric function of $z \in \mathbb{C}$ with parameters $a, b \in \mathbb{C}$ [see 99]. Because $G\left(-\frac{\varrho}{2}, \frac{1}{2}, \cdot\right)$ is an entire function for fixed parameters, we have

$$\begin{aligned}
& \sup_{x\in[0,X_{\max}]} \int |u|^\varrho f(x,u) du \\
\leq & \sup_{x\in[0,X_{\max}]} \frac{1}{\sqrt{2}} \sigma^2 2^{\varrho/2} \Gamma\left(\frac{1+\varrho}{2}\right) \sup_{z\in\left[-\frac{\max_{t\in\Omega} u^2(x,t)}{2\sigma^2}, -\frac{\min_{t\in\Omega} u^2(x,t)}{2\sigma^2}\right]} G\left(-\frac{\varrho}{2}, \frac{1}{2}, z\right) \\
< & +\infty.
\end{aligned}$$

So we can bound $\sup_{x\in[0,X_{\max}]} \int |u|^\varrho f(x,u) du$ by a constant. If we take $\varrho = 2$, we can obtain $S = \sup_x \int u^2 f(x,u) du$ bounded by a constant. So we can declare the statement in Equation 3.35.

We would also like to declare that there exist a constant $\widetilde{C}(\sigma, \|u\|_{L^\infty(\Omega)}) > 0$ such

that for any $N \geq 1$, we have

$$\max_{n=0,\ldots,N-1} \widetilde{C}(t_n, \sigma, \|u\|_{L^\infty(\Omega)}) \leq \widetilde{C}(\sigma, \|u\|_{L^\infty(\Omega)}),$$

where $\bar{C}(\sigma, \|u\|_{L^\infty(\Omega)})$ is independent of t_n, x_i, M, N, and only depends on the noisy data $\mathcal{D}$ itself.

From the above discussion, we learn that $\mathcal{W}_{2,M}$ converges to $\widetilde{C}(t_n, \sigma, \|u\|_{L^\infty(\Omega)})$, which can be bounded by $\widetilde{C}(\sigma, \|u\|_{L^\infty(\Omega)})$. If we set $\frac{\sqrt{M}\widetilde{\lambda}^{1/8}\epsilon}{16\sqrt{\log(1/\widetilde{\lambda}^{1/4})}} > \sqrt{\omega}\widetilde{C}(\sigma, \|u\|_{L^\infty(\Omega)})$ with $\omega > 1$, then there exists a positive integer $\dot{M}(\omega)$ such that as long as $M > \dot{M}(\omega)$, we have $P\left(\mathcal{W}_{2,M} \geq \frac{\sqrt{M}\widetilde{\lambda}^{1/8}\epsilon}{16\sqrt{\log(1/\widetilde{\lambda}^{1/4})}}\right) = 0$.

For the value of ω, we set it as $\omega = M^{2r}$ with $r > 0$. And we will discuss the value of r later.

By combining $P\left(\mathcal{W}_{1,M} \geq \frac{\sqrt{M}\widetilde{\lambda}^{1/8}\epsilon}{16\sqrt{\log(1/\widetilde{\lambda}^{1/4})}}\right), P\left(\mathcal{W}_{2,M} \geq \frac{\sqrt{M}\widetilde{\lambda}^{1/8}\epsilon}{16\sqrt{\log(1/\widetilde{\lambda}^{1/4})}}\right)$ together, we have when $\frac{\epsilon}{16} > \sqrt{\omega}\widetilde{C}(\sigma, \|u\|_{L^\infty(\Omega)})\sqrt{\frac{\log(1/\widetilde{\lambda}^{1/4})}{M\widetilde{\lambda}^{1/4}}}$ and $M > \dot{M}(\omega)$, we have

$$P\left(\sup\left|\frac{1}{\sqrt{M}}\rho_M(x, t_n)\right| > \frac{\epsilon}{8}\right) < 4\sqrt{2}\eta^4\widetilde{\lambda}^{\omega/4}.$$

By combining the discussion on $P\left(\sup|\mathcal{A}| > \frac{\epsilon}{4}\right), P\left(\sup|\mathcal{B}| > \frac{\epsilon}{4}\right), P\left(\sup|\mathcal{C}| > \frac{\epsilon}{4}\right)$, and $P\left(\sup|\mathcal{D}| > \frac{\epsilon}{4}\right)$, we can conclude that when

- $\frac{\epsilon}{4} > \frac{K_{\max}}{M\widetilde{\lambda}^{1/4}}B_M$
- $\frac{\epsilon}{4} > AB_M^{1-s} \ \ (s = 2)$
- $\frac{\epsilon}{4} > \sqrt{2}\frac{d^3}{dx^3}f^*(0)\widetilde{\lambda}^{3/4}$
- $\frac{\epsilon}{8} > \frac{2B_M K_{\max}(C\log M + \gamma)\log M}{\widetilde{\lambda}^{1/4}M}$
- $\frac{\epsilon}{16} > \sqrt{\omega}\widetilde{C}(\sigma, \|u\|_{L^\infty(\Omega)})\sqrt{\frac{\log(1/\widetilde{\lambda}^{1/4})}{M\widetilde{\lambda}^{1/4}}}$

we have

$$P\left(\sup|\mathcal{A}+\mathcal{B}+\mathcal{C}+\mathcal{D}|>\epsilon\right)<\underbrace{2Me^{-\frac{C_M^2}{2\sigma^2}}}_{\mathcal{Z}_1}+\underbrace{Qe^{-L\gamma}}_{\mathcal{Z}_2}+\underbrace{4\sqrt{2}\eta^4\widetilde{\lambda}^{\omega/4}}_{\mathcal{Z}_3}. \tag{3.36}$$

Let

$$\begin{cases} E_1 &= \frac{4K_{\max}}{M\widetilde{\lambda}^{1/4}}B_M \\ E_2 &= 4AB_M^{1-s} \\ E_3 &= 4\sqrt{2}\frac{d^3}{dx^3}f^*(0)\widetilde{\lambda}^{3/4} \\ E_4 &= \frac{16B_MK_{\max}(C\log M+\gamma)\log M}{\widetilde{\lambda}^{1/4}M} \\ E_5 &= 16\sqrt{\omega}\widetilde{C}(\sigma,\|u\|_{L^\infty(\Omega)})\sqrt{\frac{\log(1/\widetilde{\lambda}^{1/4})}{M\widetilde{\lambda}^{1/4}}} \end{cases},$$

by setting $\widetilde{\lambda}=M^{-a}, B_M=M^b$ with $a,b>0$, we have

$$\begin{cases} E_1 &= \frac{4K_{\max}}{M\widetilde{\lambda}^{1/4}}B_M=\frac{4K_{\max}}{M^{1-a/4-b}} \\ E_2 &= 4AB_M^{1-s}=4A\frac{1}{M^{b(s-1)}} \\ E_3 &= 4\sqrt{2}\frac{d}{dx}f^*(0)\widetilde{\lambda}^{3/4}=4\sqrt{2}\frac{d}{dx}f^*(0)M^{-3a/4} \\ E_4 &= \frac{16B_MK_{\max}(C\log M+\gamma)\log M}{\widetilde{\lambda}^{1/4}M}=\frac{16K_{\max}(C\log M+\gamma)\log(M)}{M^{1-a/4-b}} \\ E_5 &= 16\sqrt{\omega}\widetilde{C}(\sigma,\|u\|_{L^\infty(\Omega)})\sqrt{\frac{\log(1/\widetilde{\lambda}^{1/4})}{M\widetilde{\lambda}^{1/4}}}=8\sqrt{a\omega}\widetilde{C}(\sigma,\|u\|_{L^\infty(\Omega)})\sqrt{\frac{\log(M)}{M^{1-a/4}}}. \end{cases}$$

To guarantee that $E_1,E_2,E_3,E_4,E_5\to 0$ as $M\to+\infty$, we can set

$$\begin{cases} 1-a/4-b=3a/4 \\ b(s-1)>0 \\ \frac{1}{2}(1-a/4)=3a/4 \\ a,b>0 \\ s=2 \end{cases}.$$

then we have

$$\begin{cases} a = 4/7 \\ b = 3/7 \\ s = 2 \end{cases} .$$

Accordingly, we have

$$\begin{cases} E_1 = \frac{4K_{\max}}{M^{3/7}} \\ E_2 = 4AM^{-3/7} \\ E_3 = 4\sqrt{2}\frac{d^3}{dx^3}f^*(0)M^{-3/7} \\ E_4 = \frac{16K_{\max}(C\log M+\gamma)\log(M)}{M^{3/7}} \\ E_5 = 16\sqrt{\frac{\omega}{7}}\widetilde{C}(\sigma, \|u\|_{L^\infty(\Omega)})\frac{\sqrt{\log(M)}}{M^{3/7}} \end{cases} ,$$

where

$$E_1, E_2, E_3, E_5 \lesssim E_4$$

as $M \to +\infty$. Here, the operator $\lesssim$ means that when $M \to +\infty$, the order of the left side hand of $\lesssim$ will be much smaller than that on the right side hand. So we can declare that when M is sufficiently large and

$$\begin{aligned} \epsilon \quad > \quad \max \quad & \left\{ \frac{4K_{\max}}{M^{3/7}}, 4AM^{-3/7}, 4\sqrt{2}\frac{d^3}{dx^3}f^*(0)M^{-3/7}, \frac{16K_{\max}(C\log M+\gamma)\log(M)}{M^{3/7}}, \right. \\ & \left. 16\sqrt{\frac{\omega}{7}}\widetilde{C}(\sigma, \|u\|_{L^\infty(\Omega)})\frac{\sqrt{\log(M)}}{M^{3/7}} \right\} \end{aligned}$$

we have

$$\begin{aligned} P\left(\sup|\mathcal{A}+\mathcal{B}+\mathcal{C}+\mathcal{D}| > \epsilon\right) &\leq 2Me^{-\frac{C_M^2}{2\sigma^2}} + Qe^{-L\gamma} + 4\sqrt{2}\eta^4\widetilde{\lambda}^{\omega/4} \\ &= 2Me^{-\frac{\left(M^{3/7}-\|U\|_{L^\infty(\Omega)}\right)^2}{2\sigma^2}} + Qe^{-L\gamma} + 4\sqrt{2}\eta^4 M^{-\omega/7} \end{aligned}$$

□

Proof of Lemma 3.6.2

Proof. For the estimation error $\|\nabla_t \mathbf{u} - \mathbf{X}\boldsymbol{\beta}^*\|_\infty$, we have

$$\begin{aligned}
\|\nabla_t \mathbf{u} - \mathbf{X}\boldsymbol{\beta}^*\|_\infty &= \|\nabla_t \mathbf{u} - \nabla_t \mathbf{u}^* + \nabla_t \mathbf{u}^* - \mathbf{X}\boldsymbol{\beta}^*\|_\infty \\
&= \|\nabla_t \mathbf{u} - \nabla_t \mathbf{u}^* + \mathbf{X}^*\boldsymbol{\beta}^* - \mathbf{X}\boldsymbol{\beta}^*\|_\infty \\
&\leq \|\nabla_t \mathbf{u} - \nabla_t \mathbf{u}^*\|_\infty + \|(\mathbf{X}^* - \mathbf{X})\boldsymbol{\beta}^*\|_\infty. \qquad (3.37)
\end{aligned}$$

So accordingly, we have

$$P\left(\|\nabla_t \mathbf{u} - \mathbf{X}\boldsymbol{\beta}^*\|_\infty > \epsilon\right) \leq P\left(\|\nabla_t \mathbf{u} - \nabla_t \mathbf{u}^*\|_\infty > \frac{\epsilon}{2}\right) + P\left(\|(\mathbf{X}^* - \mathbf{X})\boldsymbol{\beta}^*\|_\infty\right).$$

In the remaining of the proof, we will discuss the bound of $P\left(\|\nabla_t \mathbf{u} - \nabla_t \mathbf{u}^*\|_\infty > \frac{\epsilon}{2}\right)$ and $P\left(\|(\mathbf{X}^* - \mathbf{X})\boldsymbol{\beta}^*\|_\infty\right)$ separately.

- First let us discuss the bound of $P\left(\|\nabla_t \mathbf{u} - \nabla_t \mathbf{u}^*\|_\infty > \frac{\epsilon}{2}\right)$. Because

$$\begin{aligned}
& P\left(\|\nabla_t \mathbf{u} - \nabla_t \mathbf{u}^*\|_\infty > \frac{\epsilon}{2}\right) \\
\leq\ & P\left(\max_{i=0,\ldots,M-1} \sup_{t\in[0,T_{\max}]} \left|\widehat{\frac{\partial}{\partial t}u(x_i,t)} - \frac{\partial}{\partial t}u(x_i,t)\right| > \frac{\epsilon}{2}\right) \\
\leq\ & \sum_{i=0}^{M-1} P\left(\sup_{t\in[0,T_{\max}]} \left|\widehat{\frac{\partial}{\partial t}u(x_i,t)} - \frac{\partial}{\partial t}u(x_i,t)\right| > \frac{\epsilon}{2}\right),
\end{aligned}$$

  if we set

$$\begin{aligned}
\tfrac{\epsilon}{2} > \mathscr{C}_{(\sigma,\|u\|_{L^\infty(\Omega)})} \max\ & \Big\{4K_{\max}N^{-3/7}, 4\bar{A}N^{-3/7}, 4\sqrt{2}\tfrac{d^3}{dx^3}\bar{f}^*(0)N^{-3/7}, \\
& \tfrac{16K_{\max}\left[C_{(\sigma,\|u\|_{L^\infty(\Omega)})}\log(N)+\gamma_{(N)}\right]\log(N)}{N^{3/7}}, \\
& 16\sqrt{\tfrac{\omega_{(N)}}{7}}\widetilde{C}_{(\sigma,\|u\|_{L^\infty(\Omega)})}\tfrac{\sqrt{\log(N)}}{N^{3/7}}\Big\},
\end{aligned} \qquad (3.38)$$

then we have

$$\begin{aligned} & P\left(\|\nabla_t \mathbf{u} - \nabla_t \mathbf{u}^*\|_\infty > \frac{\epsilon}{2}\right) \\ \leq\ & M\left[2Ne^{-\frac{(N^{3/7}-\|U\|_{L^\infty(\Omega)})^2}{2\sigma^2}} + Q_{(\sigma,\|u\|_{L^\infty(\Omega)})}e^{-L\gamma_{(N)}}\right. \\ & \left.+4\sqrt{2}\eta^4 N^{-\omega_{(N)}/7}\right], \end{aligned} \tag{3.39}$$

where inequity Equation 3.39 is derived according to Corollary 3.6.1.

- Second, let us discuss the bound of $P\left(\|(\mathbf{X}^* - \mathbf{X})\boldsymbol{\beta}^*\|_\infty\right)$. Because

$$\begin{aligned} & P\left(\|(\mathbf{X}^* - \mathbf{X})\boldsymbol{\beta}^*\|_\infty > \frac{\epsilon}{2}\right) \\ \leq\ & P\left(\|\boldsymbol{\beta}^*\|_\infty \max_{n=0,\ldots,N-1}\sup_{x\in[0,X_{\max}]}\sum_{k=1}^{K}\|(\mathbf{X}_k^*(x,t_n) - \mathbf{X}_k(x,t_n))\|_\infty > \frac{\epsilon}{2}\right) \\ =\ & P\left(\max_{n=0,\ldots,N-1}\sup_{x\in[0,X_{\max}]}\sum_{k=1}^{K}\|(\mathbf{X}_k^*(x,t_n) - \mathbf{X}_k(x,t_n))\|_\infty > \frac{\epsilon}{2\|\boldsymbol{\beta}^*\|_\infty}\right) \\ \leq\ & \sum_{n=0}^{N-1}\sum_{k=1}^{K} P\left(\sup_{x\in[0,X_{\max}]}\|(\mathbf{X}_k^*(x,t_n) - \mathbf{X}_k(x,t_n))\|_\infty > \frac{\epsilon}{2K\|\boldsymbol{\beta}^*\|_\infty}\right), \end{aligned}$$

if we set

$$\begin{aligned} \tfrac{\epsilon}{2K\|\boldsymbol{\beta}^*\|_\infty} \ >\ & \mathscr{C}_{(\sigma,\|u\|_{L^\infty(\Omega)})}\max\ \left\{4K_{\max}M^{-3/7}, 4AM^{-3/7}, 4\sqrt{2}\tfrac{d^3}{dx^3}f^*(0)M^{-3/7},\right. \\ & \tfrac{16\left[C_{(\sigma,\|u\|_{L^\infty(\Omega)})}\log M+\gamma_{(M)}\right]\log(M)}{M^{3/7}}, \\ & \left.16\sqrt{\tfrac{\omega_{(M)}}{7}}\widetilde{C}(\sigma,\|u\|_{L^\infty(\Omega_{(M)})})\tfrac{\sqrt{\log(M)}}{M^{3/7}}\right\}, \end{aligned} \tag{3.40}$$

then we have

$$\begin{aligned} & P\left(\|(\mathbf{X}^* - \mathbf{X})\boldsymbol{\beta}^*\|_\infty > \frac{\epsilon}{2}\right) \\ \leq\ & NK\left[2Me^{-\frac{(M^{3/7}-\|U\|_{L^\infty(\Omega)})^2}{2\sigma^2}} + Q_{(\sigma,\|u\|_{L^\infty(\Omega)})}e^{-L\gamma_{(M)}}\right. \\ & \left.+4\sqrt{2}\eta^4 M^{-\omega_{(M)}/7}\right]. \end{aligned} \tag{3.41}$$

Inequality Equation 3.41 is derived by Lemma 3.6.1.

By combining the results in Equation 3.38, Equation 3.39, Equation 3.40, Equation 3.41, we have that when

$$\begin{aligned}\frac{\epsilon}{2} \;>\; \mathscr{C}_{(\sigma,\|u\|_{L^\infty(\Omega)})} \max \;& \Big\{4K_{\max}M^{-3/7}, 4KK_{\max}\|\boldsymbol{\beta}^*\|_\infty N^{-3/7},\\ & 4AM^{-3/7}, 4K\|\boldsymbol{\beta}^*\|_\infty \bar{A}N^{-3/7},\\ & 4\sqrt{2}\tfrac{d^3}{dx^3}f^*(0)M^{-3/7}, 4\sqrt{2}K\|\boldsymbol{\beta}^*\|_\infty \tfrac{d^3}{dx^3}\bar{f}^*(0)N^{-3/7},\\ & \tfrac{16KK_{\max}\|\boldsymbol{\beta}^*\|_\infty\left[C_{(\sigma,\|u\|_{L^\infty(\Omega)})}\log(M)+\gamma_{(M)}\right]\log(M)}{M^{3/7}},\\ & \tfrac{16K_{\max}\left[C_{(\sigma,\|u\|_{L^\infty(\Omega)})}\log(N)+\gamma_{(N)}\right]\log(N)}{N^{3/7}},\\ & 16\sqrt{\tfrac{\omega_{(M)}}{7}}\widetilde{C}_{(\sigma,\|u\|_{L^\infty(\Omega)})}\tfrac{\sqrt{\log(M)}}{M^{3/7}},\\ & 16K\|\boldsymbol{\beta}^*\|_\infty\sqrt{\tfrac{\omega_{(N)}}{7}}\widetilde{C}_{(\sigma,\|u\|_{L^\infty(\Omega)})}\tfrac{\sqrt{\log(N)}}{N^{3/7}}\Big\},\end{aligned}$$

we have

$$\begin{aligned}& P\left(\|\nabla_t\mathbf{u}-\mathbf{X}\boldsymbol{\beta}^*\|_\infty>\epsilon\right)\\ \leq\;& M\left[2Ne^{-\frac{(N^{3/7}-\|U\|_{L^\infty(\Omega)})^2}{2\sigma^2}}+Q_{(\sigma,\|u\|_{L^\infty(\Omega)})}e^{-L\gamma_{(N)}}+4\sqrt{2}\eta^4N^{-\omega_{(N)}/7}\right]+\\ & NK\left[2Me^{-\frac{(M^{3/7}-\|U\|_{L^\infty(\Omega)})^2}{2\sigma^2}}+Q_{(\sigma,\|u\|_{L^\infty(\Omega)})}e^{-L\gamma_{(M)}}+4\sqrt{2}\eta^4M^{-\omega_{(M)}/7}\right]\end{aligned}$$

Now, let us do some simplification of the above results. Let $M=N^\kappa, \gamma_{(M)}=\gamma_{(N)}=$

$\frac{1}{L}N^r, \omega_{(M)} = \omega_{(N)} = N^{2r}$, and

$$\begin{cases} \mathcal{J}_1 &= 4KK_{\max}\|\boldsymbol{\beta}^*\|_\infty N^{-3\kappa/7} \\ \mathcal{J}_1' &= 4K_{\max}N^{-3/7} \\ \mathcal{J}_2 &= 4AK\|\boldsymbol{\beta}^*\|_\infty N^{-3\kappa/7} \\ \mathcal{J}_2' &= 4\bar{A}N^{-3/7} \\ \mathcal{J}_3 &= 4\sqrt{2}K\|\boldsymbol{\beta}^*\|_\infty \frac{d^3}{dx^3}f^*(0)N^{-3\kappa/7} \\ \mathcal{J}_3' &= 4\sqrt{2}\frac{d^3}{dx^3}\bar{f}^*(0)N^{-3/7} \\ \mathcal{J}_4 &= \frac{16KK_{\max}\|\boldsymbol{\beta}^*\|_\infty \left[C_{(\sigma,\|u\|_{L^\infty(\Omega)})}(\log(\kappa)+\log(N))+N^r/L\right](\log(\kappa)+\log(N))}{N^{3\kappa/7}} \\ \mathcal{J}_4' &= \frac{16K_{\max}\left[C_{(\sigma,\|u\|_{L^\infty(\Omega)})}\log(N)+N^r\right]\log(N)}{N^{3/7}} \\ \mathcal{J}_5 &= 16K\|\boldsymbol{\beta}^*\|_\infty\sqrt{\frac{N^{2r}}{7}}\widetilde{C}_{(\sigma,\|u\|_{L^\infty(\Omega)})}\frac{\sqrt{\log(\kappa)+\log(N)}}{N^{3\kappa/7}} \\ \mathcal{J}_5' &= 16\sqrt{\frac{N^{2r}}{7}}\widetilde{C}_{(\sigma,\|u\|_{L^\infty(\Omega)})}\frac{\sqrt{\log(N)}}{N^{3/7}} \end{cases}.$$

To guarantee that $\mathcal{J}_1, \mathcal{J}_1', \mathcal{J}_2, \mathcal{J}_2', \mathcal{J}_3, \mathcal{J}_3', \mathcal{J}_4, \mathcal{J}_4', \mathcal{J}_5, \mathcal{J}_5' \to 0$, as $N \to +\infty$, we need

$$\begin{cases} 3\kappa/7 - r > 0 \\ 3/7 - r > 0 \end{cases},$$

where the optimal κ is $\kappa = 1$. Accordingly, we have

$$\mathcal{J}_1, \mathcal{J}_1', \mathcal{J}_2, \mathcal{J}_2', \mathcal{J}_3, \mathcal{J}_3', \mathcal{J}_5, \mathcal{J}_5' \lesssim \mathcal{J}_4, \mathcal{J}_4'.$$

Based on the above discussion, we can declare that when N is sufficiently large, with

$$\epsilon > \mathscr{C}_{(\sigma,\|u\|_{L^\infty(\Omega)})}\frac{\log(N)}{N^{3/7-r}}$$

for any $r \in \left(0, \frac{3}{7}\right)$ and $M = O(N)$, we have

$$
\begin{aligned}
& P\left\|\nabla_t \mathbf{u} - \mathbf{X}\boldsymbol{\beta}^*\right\|_\infty > \epsilon) \\
\leq\ & M\left[2Ne^{-\frac{(N^{3/7}-\|U\|_{L^\infty(\Omega)})^2}{2\sigma^2}} + Q_{(\sigma,\|u\|_{L^\infty(\Omega)})}e^{-L\gamma_{(N)}} + 4\sqrt{2}\eta^4 N^{-\omega_{(N)}/7}\right] + \\
& NK\left[2Me^{-\frac{(M^{3/7}-\|U\|_{L^\infty(\Omega)})^2}{2\sigma^2}} + Q_{(\sigma,\|u\|_{L^\infty(\Omega)})}e^{-L\gamma_{(M)}} + 4\sqrt{2}\eta^4 M^{-\omega_{(M)}/7}\right] \\
=\ & M\left[2Ne^{-\frac{(N^{3/7}-\|U\|_{L^\infty(\Omega)})^2}{2\sigma^2}} + Q_{(\sigma,\|u\|_{L^\infty(\Omega)})}e^{-N^r} + 4\sqrt{2}\eta^4 N^{-N^{2r}/7}\right] + \\
& NK\left[2Me^{-\frac{(M^{3/7}-\|U\|_{L^\infty(\Omega)})^2}{2\sigma^2}} + Q_{(\sigma,\|u\|_{L^\infty(\Omega)})}e^{-N^r} + 4\sqrt{2}\eta^4 M^{-N^{2r}/7}\right] \\
=\ & O(Ne^{-N^r})
\end{aligned}
$$

Thus, we finish the proof of the theorem. □

Proof of Theorem 3.3.1

Proof. By KKT-condition, any minimizer $\boldsymbol{\beta}$ of Equation 3.10 must satisfies:

$$-\frac{1}{MN}\mathbf{X}^\top(\nabla_t \mathbf{u} - \mathbf{X}\boldsymbol{\beta}) + \lambda \mathbf{z} = 0 \ \text{ for } \ \mathbf{z} \in \partial\|\boldsymbol{\beta}\|_1,$$

where $\partial\|\boldsymbol{\beta}\|_1$ is the sub-differential of $\|\boldsymbol{\beta}\|_1$. The above equation can be equivalently transformed into

$$\mathbf{X}^\top\mathbf{X}(\boldsymbol{\beta} - \boldsymbol{\beta}^*) + \mathbf{X}^\top\left[(\mathbf{X} - \mathbf{X}^*)\boldsymbol{\beta}^* - (\nabla_t \mathbf{u} - \nabla_t \mathbf{u}^*)\right] + \lambda MN\mathbf{z} = 0. \tag{3.42}$$

Here matrix $\mathbf{X} \in \mathbb{R}^{MN\times K}$ is defined in Equation 3.9, and matrix $\mathbf{X}^* \in \mathbb{R}^{MN\times K}$ is defined as

$$\mathbf{X}^* \ = \ (\ \mathbf{x}_0^0 \ \ \mathbf{x}_1^0 \ \ \ldots \ \ \mathbf{x}_{M-1}^0 \ \ \mathbf{x}_1^0 \ \ \ldots \ \ \mathbf{x}_{M-1}^{N-1}\ )^\top,$$

with

$$\mathbf{x}_i^n = \left(1, \; u(x_i,t_n), \; \frac{\partial u(x_i,t_n)}{\partial x}, \; \frac{\partial^2 u(x_i,t_n)}{\partial x^2}, \; \left(\widehat{u(x_i,t_n)}\right)^2, \; \ldots, \left(\frac{\partial^2 u(x_i,t_n)}{\partial x^2}\right)^{p_{\max}} \right)^\top.$$

And vector $\boldsymbol{\beta}^* = (\beta_1, \ldots, \beta_K) \in \mathbb{R}^K$ is the ground truth coefficients. Besides, vector $\nabla_t \mathbf{u} \in \mathbb{R}^{MN}$ is defined in Equation 3.8, and vector $\nabla_t \mathbf{u}^* \in \mathbb{R}^K$ is the ground truth, i.e.,

$$\nabla_t \mathbf{u}^* = \left(\frac{\partial u(x_0,t_0)}{\partial t}, \; \frac{\partial u(x_1,t_0)}{\partial t}, \; \ldots, \; \frac{\partial u(x_{M-1},t_0)}{\partial t}, \; \frac{\partial u(x_0,t_1)}{\partial t}, \; \ldots, \; \frac{\partial u(x_{M-1},t_{N-1})}{\partial t} \right)^\top.$$

Let us denote $\mathcal{S} = \{i : \boldsymbol{\beta}_i^* \neq 0 \; \forall \; i = 0, 1, \ldots, K\}$, then we can decompose $\mathbf{X}$ into $\mathbf{X}_\mathcal{S}$ and $\mathbf{X}_{\mathcal{S}^c}$, where $\mathbf{X}_\mathcal{S}$ is the columns of $\mathbf{X}$ whose indices are in $\mathcal{S}$ and $\mathbf{X}_{\mathcal{S}^c}$ is the complement of $\mathbf{X}_\mathcal{S}$. And we can also decompose $\boldsymbol{\beta}$ into $\boldsymbol{\beta}_\mathcal{S}$ and $\boldsymbol{\beta}_{\mathcal{S}^c}$, where $\boldsymbol{\beta}_\mathcal{S}$ is the subvector of $\boldsymbol{\beta}$ only contains elements whose indices are in $\mathcal{S}$ and $\boldsymbol{\beta}_{\mathcal{S}^c}$ is the complement of $\boldsymbol{\beta}_\mathcal{S}$.

By using the decomposition, we can rewrite Equation 3.42 as

$$\begin{aligned} \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix} &= \begin{pmatrix} \mathbf{X}_\mathcal{S}^\top \mathbf{X}_\mathcal{S} & \mathbf{X}_\mathcal{S}^\top \mathbf{X}_{\mathcal{S}^c} \\ \mathbf{X}_{\mathcal{S}^c}^\top \mathbf{X}_\mathcal{S} & \mathbf{X}_{\mathcal{S}^c}^\top \mathbf{X}_{\mathcal{S}^c} \end{pmatrix} \begin{pmatrix} \boldsymbol{\beta}_\mathcal{S} - \boldsymbol{\beta}_\mathcal{S}^* \\ \boldsymbol{\beta}_{\mathcal{S}^c} \end{pmatrix} + \\ & \begin{pmatrix} \mathbf{X}_\mathcal{S}^\top \\ \mathbf{X}_{\mathcal{S}^c}^\top \end{pmatrix} [(\mathbf{X} - \mathbf{X}^*)_\mathcal{S} \boldsymbol{\beta}_\mathcal{S}^* - (\nabla_t \mathbf{u} - \nabla_t \mathbf{u}^*)] + \lambda MN \begin{pmatrix} \mathbf{z}_\mathcal{S} \\ \mathbf{z}_{\mathcal{S}^c} \end{pmatrix} \end{aligned} \tag{3.43}$$

Suppose the primal-dual witness (PDW) construction gives us an solution $(\check{\boldsymbol{\beta}}, \check{\mathbf{z}}) \in \mathbb{R}^K \times \mathbb{R}^K$, where $\check{\boldsymbol{\beta}}_{\mathcal{S}^c} = 0$ and $\check{\mathbf{z}} \in \partial \|\check{\boldsymbol{\beta}}\|_1$. By plugging $(\check{\boldsymbol{\beta}}, \check{\mathbf{z}})$ into the above equation, we have

$$\begin{aligned} \check{\mathbf{z}}_{\mathcal{S}^c} &= \mathbf{X}_{\mathcal{S}^c}^\top \mathbf{X}_\mathcal{S} (\mathbf{X}_\mathcal{S}^\top \mathbf{X}_\mathcal{S})^{-1} \mathbf{z}_\mathcal{S} - \\ & \quad \mathbf{X}_{\mathcal{S}^c}^\top \underbrace{(\mathbf{I} - \mathbf{X}_\mathcal{S} (\mathbf{X}_\mathcal{S}^\top \mathbf{X}_\mathcal{S})^{-1} \mathbf{X}_\mathcal{S}^\top)}_{\mathbf{H}_{X_s}} \frac{[(\mathbf{X} - \mathbf{X}^*)_\mathcal{S} \boldsymbol{\beta}_\mathcal{S}^* - (\nabla_t \mathbf{u} - \nabla_t \mathbf{u}^*)]}{\lambda MN} \\ &= \mathbf{X}_{\mathcal{S}^c}^\top \mathbf{X}_\mathcal{S} (\mathbf{X}_\mathcal{S}^\top \mathbf{X}_\mathcal{S})^{-1} \mathbf{z}_\mathcal{S} - \frac{1}{\lambda MN} \mathbf{X}_{\mathcal{S}^c}^\top \mathbf{H}_{X_s} \underbrace{(\mathbf{X}_\mathcal{S} \boldsymbol{\beta}_\mathcal{S}^* - \nabla_t \mathbf{u})}_{\boldsymbol{\tau}} \end{aligned} \tag{3.44}$$

From Equation 3.44, we have

$$\begin{aligned} P(\|\check{\mathbf{z}}_{\mathcal{S}^c}\|_\infty \geq 1) &= P\left(\left\|\mathbf{X}_{\mathcal{S}^c}^\top \mathbf{X}_{\mathcal{S}}(\mathbf{X}_{\mathcal{S}}^\top \mathbf{X}_{\mathcal{S}})^{-1}\mathbf{z}_{\mathcal{S}} - \frac{1}{\lambda MN}\mathbf{X}_{\mathcal{S}^c}^\top \mathbf{H}_{X_s}\boldsymbol{\tau}\right\|_\infty > 1\right) \\ &\leq P\left(\left\|\mathbf{X}_{\mathcal{S}^c}^\top \mathbf{X}_{\mathcal{S}}(\mathbf{X}_{\mathcal{S}}^\top \mathbf{X}_{\mathcal{S}})^{-1}\mathbf{z}_{\mathcal{S}}\right\|_\infty > 1-\mu\right) + \\ &\quad P\left(\left\|\frac{1}{\lambda MN}\mathbf{X}_{\mathcal{S}^c}^\top \mathbf{H}_{X_s}\boldsymbol{\tau}\right\|_\infty > \mu\right). \end{aligned}$$

If we denote $\widetilde{Z_j} = \frac{1}{\lambda MN}(\mathbf{X}_{\mathcal{S}^c})_j^\top \mathbf{H}_{X_s}\boldsymbol{\tau}$, where $(\mathbf{X}_{\mathcal{S}^c})_j$ is the j-th column of $\mathbf{X}_{\mathcal{S}^c}$, then we have

$$P(\|\check{\mathbf{z}}_{\mathcal{S}^c}\|_\infty \geq 1) \leq P\left(\left\|\mathbf{X}_{\mathcal{S}^c}^\top \mathbf{X}_{\mathcal{S}}(\mathbf{X}_{\mathcal{S}}^\top \mathbf{X}_{\mathcal{S}})^{-1}\right\|_\infty > 1-\mu\right) + P\left(\max_{j\in\mathcal{S}^c}|\widetilde{Z_j}| > \mu\right). \quad (3.45)$$

Now let us discuss the upper bound of the second term, i.e., $P\left(\max_{j\in\mathcal{S}^c}|\widetilde{Z_j}| > \mu\right)$. Because

$$\begin{aligned} P\left(\max_{j\in\mathcal{S}^c}|\widetilde{Z_j}| > \mu\right) &= P\left(\left\|\frac{1}{\lambda MN}\mathbf{X}_{\mathcal{S}^c}^\top \mathbf{H}_{X_s}\boldsymbol{\tau}\right\|_\infty > \mu\right) \\ &\leq P\left(\left\|\frac{1}{\lambda MN}\mathbf{X}_{\mathcal{S}^c}^\top \mathbf{H}_{X_s}\boldsymbol{\tau}\right\|_2 > \mu\right) \\ &\leq P\left(\left\|\frac{1}{\lambda MN}\mathbf{X}^\top \mathbf{H}_{X_s}\boldsymbol{\tau}\right\|_2 > \mu\right) \\ &\leq P\left(\frac{1}{\lambda MN}\|\mathbf{X}\|_2\|\boldsymbol{\tau}\|_2 > \mu\right) \\ &\leq P\left(\|\boldsymbol{\tau}\|_2 > \lambda\mu\sqrt{\frac{MN}{K}}\right) \\ &\leq P\left(\|\boldsymbol{\tau}\|_\infty > \lambda\mu\frac{1}{\sqrt{K}}\right) \end{aligned} \quad (3.46)$$

By Lemma 3.6.2, we know when

$$\lambda\mu\frac{1}{\sqrt{K}} > \mathscr{C}(\sigma, \|u\|_{L^\infty(\Omega)})\frac{\log(N)}{N^{3/7-r}},$$

we have

$$P\left(\|\nabla_t \mathbf{u} - \mathbf{X}\boldsymbol{\beta}^*\|_\infty > \epsilon\right) < Ne^{-N^r}.$$

So we know that

$$\begin{aligned} P\left(\|\boldsymbol{\tau}\|_\infty > \lambda\mu\frac{1}{\sqrt{K}}\right) &= P\left(\|\nabla_t \mathbf{u} - \mathbf{X}_{\mathcal{S}}\boldsymbol{\beta}^*_{\mathcal{S}}\|_\infty > \lambda\mu\frac{1}{\sqrt{K}}\right) \\ &\leq P\left(\|\nabla_t \mathbf{u} - \mathbf{X}\boldsymbol{\beta}^*\|_\infty > \lambda\mu\frac{1}{\sqrt{K}}\right) \\ &< Ne^{-N^r} \end{aligned} \tag{3.47}$$

By plugging the results in Equation 3.46 and Equation 3.47 into Equation 3.45, we have

$$\begin{aligned} P(\|\check{\mathbf{z}}_{\mathcal{S}^c}\|_\infty \geq 1) &\leq P\left(\left\|\mathbf{X}_{\mathcal{S}^c}^\top \mathbf{X}_{\mathcal{S}}(\mathbf{X}_{\mathcal{S}}^\top \mathbf{X}_{\mathcal{S}})^{-1}\right\|_\infty > 1-\mu\right) + P\left(\max_{j\in\mathcal{S}^c}|\widetilde{Z_j}| > \mu\right) \\ &\leq P\left(\left\|\mathbf{X}_{\mathcal{S}^c}^\top \mathbf{X}_{\mathcal{S}}(\mathbf{X}_{\mathcal{S}}^\top \mathbf{X}_{\mathcal{S}})^{-1}\right\|_\infty > 1-\mu\right) + P\left(\|\boldsymbol{\tau}\|_\infty > \lambda\mu\frac{1}{\sqrt{K}}\right) \\ &\leq P\left(\left\|\mathbf{X}_{\mathcal{S}^c}^\top \mathbf{X}_{\mathcal{S}}(\mathbf{X}_{\mathcal{S}}^\top \mathbf{X}_{\mathcal{S}})^{-1}\right\|_\infty > 1-\mu\right) + Ne^{-N^r} \end{aligned}$$

The probability for proper support set recovery is

$$\begin{aligned} P(\|\check{\mathbf{z}}_{\mathcal{S}^c}\|_\infty < 1) &= 1 - P(\|\check{\mathbf{z}}_{\mathcal{S}^c}\|_\infty \geq 1) \\ &\geq 1 - \left[P\left(\left\|\mathbf{X}_{\mathcal{S}^c}^\top \mathbf{X}_{\mathcal{S}}(\mathbf{X}_{\mathcal{S}}^\top \mathbf{X}_{\mathcal{S}})^{-1}\right\|_\infty > 1-\mu\right) + Ne^{-N^r}\right] \\ &= P\left(\left\|\mathbf{X}_{\mathcal{S}^c}^\top \mathbf{X}_{\mathcal{S}}(\mathbf{X}_{\mathcal{S}}^\top \mathbf{X}_{\mathcal{S}})^{-1}\right\|_\infty \leq 1-\mu\right) - Ne^{-N^r} \\ &\leq P_\mu - Ne^{-N^r}. \end{aligned}$$

Thus, we finish the proof. □

Proof of Theorem 3.3.2

Proof. By Equation 3.43, we can solve $\boldsymbol{\beta}_{\mathcal{S}} - \boldsymbol{\beta}^*_{\mathcal{S}}$ as

$$\boldsymbol{\beta}_{\mathcal{S}} - \boldsymbol{\beta}^*_{\mathcal{S}} = (\mathbf{X}_{\mathcal{S}}^\top \mathbf{X}_{\mathcal{S}})^{-1}\left[-\mathbf{X}_{\mathcal{S}}^\top(\mathbf{X}_{\mathcal{S}} - \mathbf{X}^*_{\mathcal{S}})\boldsymbol{\beta}^*_{\mathcal{S}} + \mathbf{X}_{\mathcal{S}}^\top(\nabla_t \mathbf{u} - \nabla_t \mathbf{u}^*) - \lambda MN\mathbf{z}_{\mathcal{S}}\right].$$

Thus, we have the following series of equations:

$$
\begin{aligned}
& \max_{k\in\mathcal{S}} |\boldsymbol{\beta}_k - \boldsymbol{\beta}^*_k| \\
\leq\ & \left\|(\mathbf{X}_\mathcal{S}^\top \mathbf{X}_\mathcal{S})^{-1}\right\|_\infty \left\|\mathbf{X}_\mathcal{S}^\top \left[\nabla_t \mathbf{u} - \nabla_t \mathbf{u}^* - (\mathbf{X}_\mathcal{S} - \mathbf{X}^*_\mathcal{S})\boldsymbol{\beta}^*_\mathcal{S}\right] - \lambda MN \mathbf{z}_\mathcal{S}\right\|_\infty \\
\leq\ & \left\|(\mathbf{X}_\mathcal{S}^\top \mathbf{X}_\mathcal{S})^{-1}\right\|_\infty \left[\left\|\mathbf{X}_\mathcal{S}^\top \left[\nabla_t \mathbf{u} - \nabla_t \mathbf{u}^* - (\mathbf{X}_\mathcal{S} - \mathbf{X}^*_\mathcal{S})\boldsymbol{\beta}^*_\mathcal{S}\right]\right\|_\infty + \lambda MN \left\|\mathbf{z}_\mathcal{S}\right\|_\infty\right] \\
=\ & \left\|(\mathbf{X}_\mathcal{S}^\top \mathbf{X}_\mathcal{S})^{-1}\right\|_\infty \left[\left\|\mathbf{X}_\mathcal{S}^\top \left(\nabla_t \mathbf{u} - \mathbf{X}_\mathcal{S}\boldsymbol{\beta}^*_\mathcal{S}\right)\right\|_\infty + \lambda MN \left\|\mathbf{z}_\mathcal{S}\right\|_\infty\right] && (3.48) \\
\leq\ & \left\|\left(\frac{\mathbf{X}_\mathcal{S}^\top \mathbf{X}_\mathcal{S}}{MN}\right)^{-1}\right\|_\infty \left(\frac{\left\|\mathbf{X}_\mathcal{S}^\top \left(\nabla_t \mathbf{u} - \mathbf{X}_\mathcal{S}\boldsymbol{\beta}^*\right)\right\|_\infty}{MN} + \lambda\right) && (3.49) \\
\leq\ & \sqrt{K} C_{\min} \left(\frac{\left\|\mathbf{X}_\mathcal{S}^\top \left(\nabla_t \mathbf{u} - \mathbf{X}_\mathcal{S}\boldsymbol{\beta}^*\right)\right\|_\infty}{MN} + \lambda\right) && (3.50) \\
\leq\ & \sqrt{K} C_{\min} \left(\frac{\left\|\mathbf{X}_\mathcal{S}\right\|_{\infty,\infty} \left\|\nabla_t \mathbf{u} - \mathbf{X}_\mathcal{S}\boldsymbol{\beta}^*\right\|_\infty}{MN} + \lambda\right) && (3.51) \\
\leq\ & \sqrt{K} C_{\min} \left(\frac{\left\|\mathbf{X}_\mathcal{S}\right\|_F \left\|\nabla_t \mathbf{u} - \mathbf{X}_\mathcal{S}\boldsymbol{\beta}^*\right\|_\infty}{\sqrt{MN}} + \lambda\right) \\
\leq\ & \sqrt{K} C_{\min} \left(\frac{\sqrt{MNK} \left\|\nabla_t \mathbf{u} - \mathbf{X}_\mathcal{S}\boldsymbol{\beta}^*\right\|_\infty}{\sqrt{MN}} + \lambda\right) && (3.52) \\
=\ & \sqrt{K} C_{\min} \left(\sqrt{K} \left\|\nabla_t \mathbf{u} - \mathbf{X}_\mathcal{S}\boldsymbol{\beta}^*\right\|_\infty + \lambda\right) \\
\leq\ & \sqrt{K} C_{\min} \left(\sqrt{K} \mathscr{C}_{(\sigma, \|u\|_{L^\infty(\Omega)})} \frac{\log(N)}{N^{3/7-r}} + \lambda\right) && (3.53)
\end{aligned}
$$

Equation 3.48 is because $\nabla_t \mathbf{u}^* = \mathbf{X}_\mathcal{S}\boldsymbol{\beta}_\mathcal{S}$. Equation 3.49 is because $\left\|\mathbf{z}_\mathcal{S}\right\|_\infty = 1$. Equation 3.50 is because of Condition 3.3.3. Inequality Equation 3.51 is because for a matrix $\mathbf{A}$ and a vector $\mathbf{x}$, we have $\|\mathbf{A}\mathbf{x}\|_q \leq \|\mathbf{A}\|_{p,q}\|\mathbf{x}\|_p$. Here the matrix norm for matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ in $\|\mathbf{A}\|_{\infty,\infty} = \|\text{vector}(\mathbf{A})\|_\infty$. In inequality Equation 3.52, the norm of matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ is that $\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n |A_{ij}|^2}$, and the norm of vector $\mathbf{a} \in \mathbb{R}^d$ is $\|\mathbf{a}\|_\infty = \max_{1\leq i\leq d} |a_i|$. Inequality Equation 3.52 is because we normalized columns of $\mathbf{X}$. Inequality Equation 3.53 is due to Lemma Lemma 3.6.2 under probability $1 - O(Ne^{-N^r}) \to 1$. □

CHAPTER 4

A HOMOTOPIC METHOD TO SOLVE THE LASSO PROBLEMS WITH AN IMPROVED UPPER BOUND OF CONVERGENCE RATE

4.1 Introduction

Lasso [see 7] has demonstrated to be an effective methods in model estimation and selection in the past decades. Lasso aims to enable sparsity during the estimation process when the dimension becomes increasingly large. We review the formulation of Lasso-type estimator in the following. Let $\mathbf{y} \in \mathbb{R}^n$ denote a response vector and $\mathbf{X} \in \mathbb{R}^{n \times p}$ be a model matrix (of predictors), and vector $\boldsymbol{\beta}^*$ is the true regression coefficients. We want to estimate $\boldsymbol{\beta}$. Vector $\mathbf{w}$ contains white-noise entries, which are independently and identically distributed following the Normal distribution $N(0, \sigma^2)$. Accordingly, the data generation mechanism in the classical linear regression model can be written as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta}^* + \mathbf{w}.$$

The Lasso estimator $\widehat{\boldsymbol{\beta}}$ is commonly written as

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{2n} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1 \right\}, \tag{4.1}$$

where parameter λ controls the trade-off between the sparsity and model's goodness of fit. Essentially, solving Equation 4.1 is an optimization problem, where many research in operations research and optimization have devoted to. For simplicity, we denote the objective function as follows:

$$F(\boldsymbol{\beta}) = \frac{1}{2n} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1. \tag{4.2}$$

In this paper, we don't consider the selection of parameter λ, which by itself has a large literature; consequently, we don't include λ in the notation $F(\boldsymbol{\beta})$. *Lasso-algorithms* are these algorithms that aim to solve Lasso problems with an objective function as in Equation 4.2. This paper considers a Lasso-algorithm that has a better provable upper bounds in its convergence rate. The convergence rate will be derived in terms of order of computational complexity. Because there is no closed form solution of the minimizer $\widehat{\boldsymbol{\beta}}$ (unless subgradient is utilized), most Lasso-algorithms are iterative with an iteration index k. For an iterative estimator/minimizer, their distance to the optimal estimator/minimizer can be measured by ϵ-precision, whose definition is listed below.

Defination 4.1.1. Suppose $\boldsymbol{\beta}^{(k)}$ is the kth iterative estimator in a certain Lasso-algorithm and $\widehat{\boldsymbol{\beta}}$ is the global minimizer that is defined as $\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} F(\boldsymbol{\beta})$. Recall that $F(\boldsymbol{\beta})$ is defined in Equation 4.2. For any pre-fixed $\epsilon > 0$, if we have

$$F(\boldsymbol{\beta}^{(k)}) - F(\widehat{\boldsymbol{\beta}}) \leq \epsilon, \tag{4.3}$$

then we declare that $\boldsymbol{\beta}^{(k)}$ achieves the ϵ-*precision*.

The *order of complexity* will be utilized to measure how fast an iterative algorithm converges to the global minimum. Recall that our optimization problem is to minimize the function $F(\boldsymbol{\beta})$. We obtain the iterative estimator $\boldsymbol{\beta}^{(k)}$ at the kth iteration. Recall that $\widehat{\boldsymbol{\beta}}$ denotes the global minimum as in Definition 4.1.1. The order of complexity measures the number of operations needed to achieve the ϵ-precision defined in Definition 4.1.1. More specifically, we adopt the big O notation as follows. The order of complexity of a Lasso-algorithm is $O\left(np\frac{1}{\epsilon}\right)$, if in order to achieve the ϵ-precision, the number of all needed numeric operations can be upper bounded by a constant multiplies $np\frac{1}{\epsilon}$. Notice that the order of complexity gives an upper bound of the number of numerical operations in order to achieve certain precision. It does not say anything on the average performance of the algorithm. It is possible that an algorithm with larger upper bounds performs better

in some cases than an algorithm with lower upper bounds. In this paper, we consider a numerical strategy that can lead to an iterative algorithm that can achieve the aforementioned ϵ-precision with a lower order of complexity. Recall that $\epsilon > 0$ is typically small. In theory, an $O\left(np\frac{1}{\sqrt{\epsilon}}\right)$ Lasso-algorithm has a lower order of complexity than an $O\left(np\frac{1}{\epsilon}\right)$ Lasso-algorithm. Moreover, an $O\left(np\log(\frac{1}{\epsilon})\right)$ Lasso-algorithm has an even lower order of complexity. We will then use numerical simulations to compare with some representative algorithms in some well-studied cases.

Due to the nature of the objective function in the Lasso problem, which is $F(\boldsymbol{\beta}) = \frac{1}{2n}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2+\lambda\|\boldsymbol{\beta}\|_1$, the first-order method is often used. The first term in $F(\boldsymbol{\beta})$ is a nice quadratic function, which is numerically amenable. The challenge is rooted in the second term of $F(\boldsymbol{\beta})$, the ℓ_1 regularization term $\|\boldsymbol{\beta}\|_1$, which is not differentiable at the origin. We review some state-of-the-art Lasso-algorithms, which will serve as the benchmarks of our algorithm. [55] proposes a Lasso-algorithm using the first-gradient and the Hessian matrix of the first term in $F(\boldsymbol{\beta})$. [55] approximate $\frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2$ by its second-order Taylor expansion. It is computationally expensive to directly calculate the Hessian matrix of $\frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2$, i.e., $\mathbf{X}^\top\mathbf{X}/n$, especially when p is large. To avoid the time-consuming calculation of the Hessian matrix, a key idea in [55] is to approximate the Hessian matrix by a diagonal matrix, whose diagonal entries are the maximal eigenvalue of $\mathbf{X}^\top\mathbf{X}/n$. After the quadratic approximate of the objective function, a proximal mapping is formed, where the soft-thresholding operator can be easily applied.

Proximal gradient descent is adopted in [55]. See additional mathematical review in subsection 4.6.1. Early foundational work on proximal gradient descent can be found in [100, 101, 102]. As the techniques matures, they became widely used in different fields. As a result, they have been referred to by a diverse set of names, including proximal algorithm, proximal point, and so on. In the survey of [103], it can be seen that, many other widely-known statistical methods – including, the Majorization-Minimization (MM) [see 104, 105], and the Alternating Direction Method of Multipliers (ADMM) [see 106] – fall

into the proximal framework. In the review of the present paper, many Lasso-algorithms follow the proximal gradient descent as well.

The classical first-order method use the gradient at the immediate previous solution. To learn from the "history," [24] proposes a Lasso-algorithm, which takes advantage of the gradients at previous two solutions. Since it uses the historic information, it is also referred as the *momentum* algorithm. Essentially, [24] falls into the framework of the Accelerate Gradient Descent (AGD), which is proposed by [107], and later widely applied into many optimization problems to speed up the convergence rate, seeing examples in [24, 108, 109, 110, 111], and many more. The mathematical details of the aforementioned two Lasso-algorithms (as well as the coming ones) are provided in subsection 4.6.1.

The above two Lasso-algorithms update their estimates globally. On the contrary, the third Lasso-algorithm utilizes coordinate descent to update the estimate. This method is widely used and an corresponding R package named *glmnet* [see 56] has fueled its adoption. The coordinate descent method has been proposed for the Lasso problem for a number of times, but only after [56], was its power fully appreciated. Early research work on the coordinate descent include the discovery by [112] and [113], and the convergence analysis by [114]. There are research work done on the applications of coordinate descent on Lasso problems, such as [115, 116, 117, 118], and so on. We choose [56] as a method to compare, since its implementation in the R package, *glmnet*, is very well-known by statisticians.

The aforementioned three Lasso-algorithms do not use a surrogate for the ℓ_1 regularization term. Different from them, the fourth Lasso-algorithm aims to find a surrogate of the ℓ_1 penalty term. Recall that the non-differentiability of the ℓ_1 penalty at the origin makes it hard to enable fast convergence rate when applying the gradient descent method. [119] proposes a surrogate function of the ℓ_1 penalty by taking advantage of the non-negative projection operator (seeing equation (2) and (3) in [119] for more details), where the surrogate function is twice differentiable. Consequently, the expectation?maximization (EM) algorithms [see 120] are used for the optimization. Among this type of Lasso-algorithms,

we select Smooth Lasso (SL) [see 121] as a benchmark, because it is developed recently and is an improved version of [119]. The difference from our proposed algorithm is that, we design a homotopic path (i.e., a sequence of surrogate functions) to make the surrogate functions closer and closer to the ℓ_1 penalty.

Another famous Lasso-algorithm utilizes the *path-following* idea [see 122, 123, 124]. The main idea of path-following Lasso-algorithm is described as follows. It begins with a large penalty parameter λ, which leads all the estimated coefficients to 0. Then it tries to identify a sequence of decreasing penalty parameter λ, such that when λ is between two kink point, support set (the set of non-zero entries of estimated $\boldsymbol{\beta}$) remains unchanged. Moreover, the estimated $\boldsymbol{\beta}$ elementwisely is a linear function of λ. However, when one is over the kink point, the support is changed. This type of algorithms have two major drawbacks. First, it is not guaranteed to work in general cases. As of now, the work in the current literature only establishes the path following Lasso-algorithm in special situations (see subsubsection 4.6.2 for a concrete counter example where the path following Lasso-algorithm fails). Second, the contemporary literature indicates that determining the number of iterations in a path following algorithm is an open question [see 122, 123]. This indicates that there is no theoretical guarantee that the order of complexity of a path following approach is low, considering that the maximum number of iterations can be as large as 2^p, where p is the number of predictors (see subsubsection 4.6.2 for a detailed discussion). Given the above two drawbacks, we think that our paper offers significant value to the advancement of the existing literature. We will design a two-layer iteration algorithm to achieve a provable faster convergence rate. To the best of our knowledge, such a faster rate has not appeared in the literature.

4.1.1 Contribution

We propose a new Lasso-algorithm that has a better provable upper bounds in its convergence rate, in terms of the order of complexity. The state-of-the-art Lasso-algorithms we

Table 4.1: The available orders of complexity of four existing Lasso-algorithms and ours (in the last column) for achieving the ϵ-precision. The common factor that involves n (the sample size) and p (the dimensionality of the parameter) is omitted for simplicity.

method	ISTA	FISTA	CD	SL	Ours
Order of complexity	$O(1/\epsilon)$	$O(1/\sqrt{\epsilon})$	$O(1/\epsilon)$	$O(1/\epsilon)$	$O\left([\log(1/\epsilon)]^2\right)$

compare include

1. the Iterative Shrinkage-Thresholding Algorithm (ISTA) [see 55],

2. the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [see 24],

3. a Coordinate Descent (CD) algorithm[see 56] and

4. the Smooth Lasso [see 121].

These algorithms are representative in the literature. See subsection 4.6.1 for technical details on the above four benchmark algorithms. Since the path-following Lasso-algorithm doesn't work for general cases and there is no theoretical guarantee of its computational complexity is low, we exclude it from our benchmark algorithms. To show the advantage of our proposed method, in Table 4.1, we list the provable upper bounds in convergence rate of four benchmark algorithms and our algorithm.

We see that our algorithm achieves an order of complexity of *log-polynomial* of $1/\epsilon$, while other benchmarks have order of complexity of *polynomial* of $1/\epsilon$. The order of complexity of our proposed algorithm is established in Theorem 4.3.3.

4.1.2 Organization of this Paper

The organization of the rest of the paper is as follows. We develop our Lasso-algorithm in section 4.2. The related main theory is established in section 4.3. Numerical examples are shown in section 4.4. Some discussion are presented in section 4.5. In subsection 4.6.1, we summarize some necessary technical details of these benchmark algorithms. A useful theorem on the accelerated gradient descents is restated in subsection 4.6.3. All the technical

proofs are relegated to subsection 4.6.4.

For the notations throughout the paper, scalars are denoted by lowercase letters (e.g., β). Vectors are denoted by lowercase boldface letters ($\boldsymbol{\beta}$) and its ith element is noted as β_i. Matrices are denoted by uppercase boldface letter ($\mathbf{B}$) and its (i, j)th element is noted as B_i.

4.2 The Proposed Algorithm

To circumvent the undesirable behavior of the ℓ_1 penalty function ($\|\boldsymbol{\beta}\|_1$) at the origin, we design an algorithm that solves a sequence of optimization problem: in each subproblem, the ℓ_1 penalty function is replaced by a surrogate function. The surrogate functions ultimately converge to the ℓ_1 penalty function. Our approach falls into the general framework of homotopic methods, therefore, we name our algorithm a *Homotopy-Shrinkage* (HS) algorithm. Its connection to *shrinkage* becomes evident when we describe the algorithm in details.

This section is organized as follows. A general description of the proposed HS algorithm is shown in subsection 4.2.1. In subsection 4.2.2, we describe how to choose the initial value of the hyper-parameter t, as well as its updating scheme. In subsection 4.2.3, we present our design of early stopping in the inner loop, which is critical to achieve the lower order of complexity. The design of the surrogate functions (to approximate the ℓ_1 penalty) is provided in subsection 4.2.4.

4.2.1 Overview of the Proposed Algorithm

We design our Homotopy-Shrinkage algorithm that has two layer of loops: an outer-loop and an inner-loop. In an outer-iteration (in the outer-loop), the following objective function is minimized

$$F_t(\boldsymbol{\beta}) = \frac{1}{2n} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda f_t(\boldsymbol{\beta})$$

where $f_t(\boldsymbol{\beta})$ is a surrogate function of the target function $\|\boldsymbol{\beta}\|_1$. Here $t > 0$ is a parameter in the function, which controls the closeness between $\|\boldsymbol{\beta}\|_1$ and $f_t(\boldsymbol{\beta})$. More specifically, if t decreases to zero, the function $f_t(\boldsymbol{\beta})$ converges to $\|\boldsymbol{\beta}\|_1$. As mentioned earlier, the design of function $f_t(\boldsymbol{\beta})$ is postponed to subsection 4.2.4. Each outer-iteration takes the previous stopping position (from the previous outer-iteration) as the starting point of this iteration. In the outer-loop, we start with a large t initially, and then gradually decrease the value of t until the desired accuracy ϵ is reached. The way to decrease the value of t is a non-trivial task to make the designed algorithm has a provable lower order of complexity. The details on how to decrease the value of t are given later.

In an inner-loop, for a fixed t, we employ the accelerated gradient descent (AGD) to minimize the current surrogate objective function $F_t(\boldsymbol{\beta})$. Note that by design, the surrogate function $F_t(\boldsymbol{\beta})$ will be strongly convex and well conditioned, consequently a lower order of complexity becomes achievable in the inner-loop. In particular, one can prove an log-polynomial computational complexity for this algorithm. To summarize the above key idea of our proposed algorithm, we present the pseudo code in algorithm 9. Details of the

Algorithm 9: Pseudo code of the proposed Homotopy-Shrinkage algorithm

Input: A response vector $\mathbf{y}$, a model matrix $\mathbf{X}$, a parameter λ that relates to the Lasso.

Output: an estimator of $\boldsymbol{\beta}$, which satisfies the ϵ-precision.

1 **Hypter-parameter initialization (See subsection 4.2.2)**

2 ► ***Outer-Iteration:*** ◄ **while** *the precision ϵ is not achieved* **do**

3 shrink t**;** /* *For detailed shrinkage procedure, please refer to line 13 in algorithm 10.* */

4 ► ***Inner-Iteration:*** ◄ **use AGD to minimize $F_t(\boldsymbol{\beta})$ until the precision of the inner-iteration is achieved**

/* $F_t(\boldsymbol{\beta}) = \frac{1}{2n} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda f_t(\boldsymbol{\beta})$ */

/* *For detailed procedure of AGD, please refer to the inner-iteration in algorithm 10.* */

proposed algorithm are discussed in the remainder of this section.

4.2.2 Value of the Hyper-parameter in the Proposed Algorithm

The first detail we would like to discuss is the choice of the initial value of the hyper-parameter t, which is denoted as t_0. A well-designed initial point t_0 is important, because starting with an unnecessarily large t_0 would end up with more shrinkage steps (i.e., the outer-iterations), which in turns costs more numerical operations. We derive a minimal value of t_0 in Equation 4.4 in Lemma 4.2.1.

Lemma 4.2.1. Suppose in a Lasso problem, we have the response vector $\mathbf{y} \in \mathbb{R}^n$ and a model matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$. For our proposed algorithm, there exist a value t_0 that satisfies the following:

$$t_0 \in \left\{ t : \left| \sum_{j=1}^{p} \mathbf{M}(t)_{ij} (\mathbf{X}^\top \mathbf{y}/n)_j \right| \leq t \right\}, \quad \forall i = 1, \ldots, p, \tag{4.4}$$

where $\mathbf{M}(t) = \left(\frac{\mathbf{X}^\top \mathbf{X}}{n} + \frac{\lambda}{3t^3} \left[\log(1+t)\right]^2 \mathbf{I} \right)^{-1}$. Here $\mathbf{X}^\top$ represents the transpose of matrix $\mathbf{X}$, and we use this notation in the remaining of this chapter. When one chooses the aforementioned t_0 as the initial point in the proposed algorithm, we have $\left|\beta_i^{(0)}\right| \leq t_0$ for any $1 \leq i \leq p$, where $\beta_i^{(0)}$ denotes the ith entry in the vector $\boldsymbol{\beta}^{(0)} = \mathbf{M}(t_0)\mathbf{X}^\top \mathbf{y}/n$.

Proof. See subsubsection 4.6.4. □

The motivation of the above lemma is to ensure that when $t = t_0$ in our proposed algorithm, the initial estimator $\boldsymbol{\beta}^{(0)}$ is going to be bounded by t_0 entry-wise.

The second detail we would like to discuss is the design of the shrinkage path of t in line 3 in algorithm 9. This is designed as follows. First, we start with a relative large t_0, which has already been discussed above. Then, in the kth outer-iteration ($k \geq 0$), we shrink the t_k to $t_{k+1} = t_k(1-h)$, where h is set to be a predetermined values. In our proofs, we will show that this can lead to a provable lower bound in the order of complexity.

4.2.3 Early Stopping in the Inner-Loop and the Complete Algorithm

To achieve a better order of complexity, it is critical to stop the inner iteration early. Specifically speaking, in the kth outer-iteration, the inner-iteration is stopped when

$$F_{t_k}(\boldsymbol{\beta}^{(k)[s]}) - F_{\min,k} < \widetilde{\epsilon}_k,$$

where $\boldsymbol{\beta}^{(k)[s]}$ denotes the iterative estimator in the sth inner-iteration of the kth outer iteration, and we have

$$F_{\min,k} = \min_{\boldsymbol{\beta}} F_{t_k}(\boldsymbol{\beta}).$$

Here, we set $\widetilde{\epsilon}_k = \frac{\lambda p}{3B}\left[\log(1+t_k)\right]^2$, where B is the upper bound of $\left|\beta_i^{(k)}\right|$ for all $i = 1, 2, \ldots, p$ and $k = 1, 2, \ldots$. The justification of our choice of $\widetilde{\epsilon}_k$ is elaborated in our proof; its detailed derivation can be found in subsubsection 4.6.4. It is worth noting that, theoretically, our algorithm can achieve the order of complexity of $O\left((\log(1/\epsilon))^2\right)$, yet, in practice, it may not be implementable. We may have to use some alternative, such as stopping the inner-iteration after a fixed number of steps. The matter of fact is that, the stoping rule in the inner-iteration of our algorithm requires knowing the value of $F_{\min,k}$, which is not possible. In simulations, it seems that we can get around it by setting a fixed number of inner iterations.

We re-describe algorithm 9 in algorithm 10, including some of the additional details that are discussed above. In particular, we elaborate the detailed steps of the inner-iteration, which used to be the line 4 of algorithm 9; recall that this is an implementation of an accelerated gradient descent algorithm.

4.2.4 Design of the replacement function $f_t(\boldsymbol{\beta})$

This section discuss the design of the surrogate function $f_t(\boldsymbol{\beta})$, which is to replace the ℓ_1 penalty in the original objective function that is in Equation 4.2. It is widely acknowledge

Algorithm 10: A detailed version of our proposed algorithm

Input: $\mathbf{y} \in \mathbb{R}^n, \mathbf{X} \in \mathbb{R}^{n \times p}$, λ, t_0, h, ϵ, B

Output: an estimator of $\boldsymbol{\beta}$, noted as $\boldsymbol{\beta}^{(k)}$, which achieves the ϵ-precision.

1 **initialization** t_0, h, $k = 1$, $\boldsymbol{\beta}^{(0)} = \left[\mathbf{X}^\top \mathbf{X} + \frac{2n\lambda[\log(1+t_0)]^2}{3t_0^2}\mathbf{I}\right]^{-1} \mathbf{X}^\top \mathbf{y}$

2 ▶ ***Outer-Iteration:*** ◀ **while** $F(\boldsymbol{\beta}^{(k-1)}) - F_{\min} > \epsilon$ **do**

3 $\quad s = 1$

4 $\quad \boldsymbol{\beta}^{(k)[0]} = \boldsymbol{\beta}^{(k-1)}$

5 $\quad \bar{\boldsymbol{\beta}}^{(k)[0]} = \boldsymbol{\beta}^{(k-1)}$

6 $\quad \widetilde{\epsilon}_k = \frac{\lambda p}{3B}\left[\log(1+t_k)\right]^2$

7 $\quad$ ▶ ***Inner-Iteration:*** ◀ **while** $F_{t_k}(\bar{\boldsymbol{\beta}}^{(k)[s-1]}) - F_{\min,k} > \widetilde{\epsilon}_k$ **do**

8 $\quad\quad \underline{\boldsymbol{\beta}}^{(k)[s]} = (1-q_s)\bar{\boldsymbol{\beta}}^{(k)[s-1]} + q_s \boldsymbol{\beta}^{(k)[s-1]}$

9 $\quad\quad \overline{\boldsymbol{\beta}}^{(k)[s]} =$
$\quad\quad\quad \arg\min_{\boldsymbol{\beta}} \left\{ \gamma_s \left[\boldsymbol{\beta}^\top \nabla F_{t_k}(\underline{\boldsymbol{\beta}}^{(k)[s]}) + \mu_k V(\underline{\boldsymbol{\beta}}^{(k)[s]}, \boldsymbol{\beta}) \right] + V(\boldsymbol{\beta}^{(k)[s-1]}, \boldsymbol{\beta}) \right\}$

10 $\quad\quad \bar{\boldsymbol{\beta}}^{(k)[s]} = (1-\alpha_s)\bar{\boldsymbol{\beta}}^{(k)[s-1]} + \alpha_s \boldsymbol{\beta}^{(k)[s]}$

11 $\quad\quad s = s + 1$

12 $\quad \boldsymbol{\beta}^{(k)} = \bar{\boldsymbol{\beta}}^{(k)[s]}$

13 $\quad t_k = t_{k-1}(1-h)$

14 $\quad k = k + 1$

In line 2, $F_{\min} = \min_{\boldsymbol{\beta}} F(\boldsymbol{\beta})$.

In line 4 and the rest of this paper, we use parenthesis (k) to denote the kth outer-iteration, and we use bracket $[s]$ to denote the sth inner-iteration.

In line 7, $F_{\min,k} = \min_{\boldsymbol{\beta}} F_{t_k}(\boldsymbol{\beta})$.

In line 8, in this paper, we choose q_s as $q_s = q = \frac{\alpha_k - \mu_k/L_k}{1-\mu_k/L_k}$ for $s = 1, 2, \ldots$, where $\alpha_k = \sqrt{\frac{\mu_k}{L_k}}$. And L_k, μ_k is defined as $\|\nabla F_{t_k}(\mathbf{x}) - \nabla F_{t_k}(\mathbf{y})\|_2^2 \leq L_k \|\mathbf{x} - \mathbf{y}\|_2$, $F_{t_k}(y) \geq F_{t_k}(\mathbf{x}) + \nabla F_{t_k}(\mathbf{x})(\mathbf{y} - \mathbf{x}) + \frac{\mu_k}{2}\|\mathbf{y} - \mathbf{x}\|_2^2$.

In line 9, we choose γ_s as $\gamma_s = \gamma = \frac{\alpha}{\mu_k(1-\alpha_k)}$ for $s = 1, 2, \ldots$. Here $V(\mathbf{x}, \mathbf{z})$ is defined as $V(\mathbf{x}, \mathbf{z}) = v(\mathbf{z}) - \left[v(\mathbf{x}) + \nabla v(\mathbf{x})^\top (\mathbf{z} - \mathbf{x})\right]$, with $v(\mathbf{x}) = \|\mathbf{x}\|_2^2 / 2$.

that, if the objective function is strongly convex and well conditioned, then the gradient descent method can achieve very fast convergence rate. It is also known that the ℓ_1 norm ($\|\boldsymbol{\beta}\|_1$ in our paper) is not strongly convex, while the quadratic function (such as $\|\boldsymbol{\beta}\|_2^2$) can be easily proved to be strongly convex. Motivated by these facts, we try to replace the ℓ_1 penalty ($\|\boldsymbol{\beta}\|_1$ in $F(\boldsymbol{\beta}) = \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1$) by $f_t(\boldsymbol{\beta})$, which is quadratic near 0 and almost linear outside. By making this replacement, the surrogate objective function $F_t(\boldsymbol{\beta}) = \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda f_t(\boldsymbol{\beta})$ can achieve strongly convex. It is nontrivial to find a good surrogate function $f_t(\boldsymbol{\beta})$. We list the requirements of $f_t(\boldsymbol{\beta})$ in Condition 4.2.1.

Condition 4.2.1. Assume function $f_t(x)$ satisfies the following conditions.

1. When $t \to 0$, we have $f_0(x) = |x|$, where $|x|$ is the absolute value function.
2. For fixed $t > 0$, function $x \mapsto f_t(x)$ is quadratic on $[-t, t]$, here $\mapsto$ indicates that the left hand side (i.e., x) is the variable in the function in the right hand side (i.e., $f_t(x)$). We following this convention in the rest of this paper.
3. Function $x \mapsto f_t(x)$ is C^1. Here C^1 is the set of all continuously differentiable functions.
4. Function $f_t(x)$ has the second derivative with respective to x.

Following the requirements in Condition 4.2.1, we design $f_t(x)$ in the following equation, where the input variable x is a scalar.

$$f_t(x) = \begin{cases} \frac{1}{3t^3}\left[\log(1+t)\right]^2 x^2, & \text{if } |x| \le t, \\ \left[\frac{\log(1+t)}{t}\right]^2 |x| + \frac{1}{3|x|}\left[\log(1+t)\right]^2 - \frac{1}{t}\left[\log(1+t)\right]^2, & \text{otherwise.} \end{cases} \tag{4.5}$$

Figure 4.1 displays this surrogate function $f_t(x)$ and its derivatives when the parameter t takes different values. The first row in Figure 4.1 shows the closeness between $f_t(x)$ and $|x|$ when t changes, and the second and third rows present their first and second derivatives,

respectively. It can be seen that, when $t \to 0$, both $f_t(x)$, its first and second derivative become closer to the counterparts of the function $|x|$.

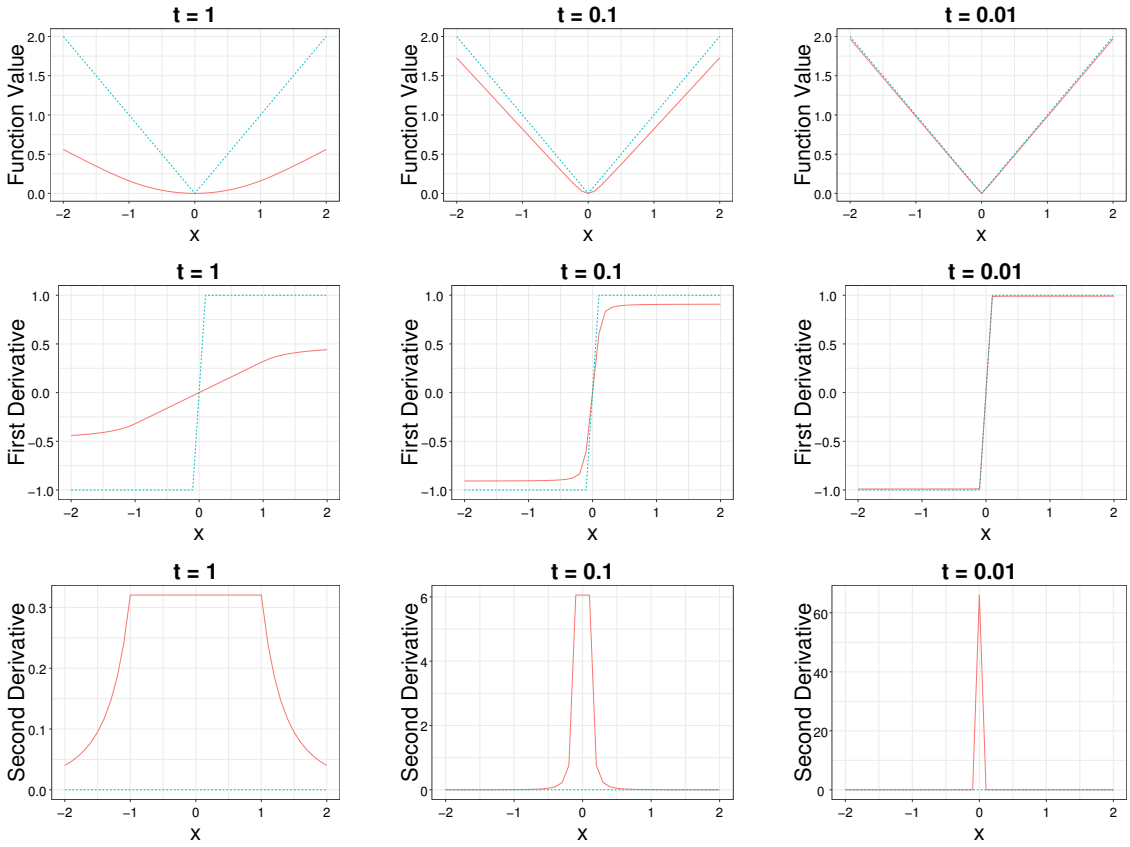


Figure 4.1: The red solid line in the first, second, third row represents the function $f_t(x)$, its first derivative, and its second derivatives, respectively, under the scenario when $t = 1, t = 0.1$ and $t = 0.01$. The blue dashed line in the first, the second, and the third row represents $|x|$, its first derivative, and its second derivatives, respectively, under the same scenarios. For function $|x|$, the first and second derivatives are not defined at the origin. This figure shows the closeness between $f_t(x)$ and $|x|$ when t converges to zero.

It is also worth noting that when the input variable is a vector instead of a scaler, for example if we have $\boldsymbol{\beta} = (\beta_1 \cdots \beta_p)^\top$, then $f_t(\boldsymbol{\beta})$ can be defined accordingly: $f_t(\boldsymbol{\beta}) = \sum_{i=1}^{p} f_t(\beta_i)$.

Remark 4.2.1. The design of $f_t(x)$ in Equation 4.5 is not unique but needs to satisfy some special requirements. Generally speaking, we can assume that $f_t(x)$ has the following

format:

$$f_t(x) = \begin{cases} d(t)x^2, & \text{if } |x| \le t, \\ a(t)|x| + b(t)g(x) + c(t), & \text{otherwise.} \end{cases} \tag{4.6}$$

The requirement in Condition 4.2.1 is equivalently transformed into :

1. both $x \mapsto f_t(x)$ and $t \mapsto f_t(x)$ are C^1.
2. $a(0) = 1$, $b(0) = 0$, $c(0) = 0$, so that $f_0(x) = |x|$.

Besides, we wish the second derivative of $f_t(x)$ has the format of $f_t''(x) = h(t)\max\{t, |x|\}^{\upsilon}$, where $h(t)$ is a function of t and υ is a constant. Accordingly, it is reasonable to suppose that $g(x) = \frac{1}{(1-\upsilon)(2-\upsilon)}x^{2-\upsilon}$. Combining all the requests above, one has

$$a(t) = \frac{\upsilon}{1+\upsilon}t^{1+\upsilon}b(t).$$

Since $a(0) = 1$, we choose $b(t) = \frac{1+\upsilon}{\upsilon}\left[\log(1+t)\right]^{1+\upsilon}$. Other choice of $b(t)$ can be $\sin(\cdot)$ function or other functions, which makes $t^{1-\upsilon}b(t)$ as an constant when $t = 0$.

The purpose of designing $f_t(\boldsymbol{\beta})$ is to replace the ℓ_1 penalty ($\|\boldsymbol{\beta}\|_1$) and then shrink t in each iteration k, i.e., $t_k = t_{k-1}(1-h)$, where h is a parameter controlling the shrinking degree of t. Because of the first statement in Condition 4.2.1, the replacement $f_t(\boldsymbol{\beta})$ gets more and more close to $\|\boldsymbol{\beta}\|_1$ as $t \to 0$.

Remark 4.2.2. Our idea is similar to [125] in appearance, however, the differences are as follows.

1. [125] aims at ℓ_p penalty, where $p \notin \{1, 2, +\infty\}$, while we focus on the $p = 1$, which is not discussed in [125] and the theory in [125] is not easily-extendable to the situation when $p = 1$.
2. [125] minimizes a linear function instead of the quadratic residual $\frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2$, where the Hessian matrix of the objective function needs different treatment.

Suppose that in the kth iteration, by replacing $\|\boldsymbol{\beta}\|_1$ with $f_{t_k}(\boldsymbol{\beta})$, the objective function of the Lasso problem — $F(\boldsymbol{\beta}) = \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1$ — is transformed into

$$F_{t_k}(\boldsymbol{\beta}) = \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda f_{t_k}(\boldsymbol{\beta}). \tag{4.7}$$

The surrogate function $f_{t_k}(x)$ in Equation 4.5 has a property in the following lemma.

Lemma 4.2.2. Suppose that from the beginning of our algorithm to the end of our algorithm, we have that $\beta_i^{(k)} \leq B$ for any $i \in \{1, 2 \dots p\}$ and $k \in \{1, 2, \dots\}$. Then for any $k \in \{1, 2, \dots\}$, the surrogate function defined in Equation 4.5, i.e., $f_{t_k}(x)$, has the following property:

$$f_{t_k}(B) - B \leq f_{t_k}(x) - |x| \leq 0. \tag{4.8}$$

Proof. See subsubsection 4.6.4. □

4.3 Order of complexity of the HS Algorithm

This section deals with the order of complexity of the HS algorithm, i.e., how many number of operations needed to achieve the ϵ-precision that is defined in Definition 4.1.1. Since our HS algorithm involves two layers of iterations — one is for the shrinkage of t (we call it an *outer-loop*), and the other is the AGD optimization (we call it the *inner-loop*)— the order of complexity is mainly effected by: (i) the number of inner-iterations, (ii) the number of outer-iterations, (iii) the number of operations in each inner-iteration. To solve these components respectively, we discuss (i) in subsection 4.3.1, and (ii) in subsection 4.3.2. And because (iii) is very similar to that in algorithm 11, algorithm 12, algorithm 13, and algorithm 14, we will not discuss it separately in a section and will only discuss it briefly in subsection 4.3.3.

4.3.1 Number of Inner-Iteration

Recall the line 7 - line 11 in algorithm 10, for a fixed t_k, the inner-loop aims at finding an estimator $\boldsymbol{\beta}^{(k)}$, whose precision is shown in the following equation

$$F_{t_k}(\boldsymbol{\beta}^{(k)}) - F_{\min,k} \leq \widetilde{\epsilon}_k, \tag{4.9}$$

where $\widetilde{\epsilon}_k$ is the precision we set for the AGD algorithm on optimizing function $F_{t_k}(\boldsymbol{\beta}) = \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda f_{t_k}(\boldsymbol{\beta})$ and the value of $\widetilde{\epsilon}_k$ will be specified later. We denote $\widehat{\boldsymbol{\beta}}^{(k)} = \arg\min_{\boldsymbol{\beta}}\{\frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda f_{t_k}(\boldsymbol{\beta})\}$, and we have $F_{\min,k} = F(\widehat{\boldsymbol{\beta}}^{(k)})$. In the kth outer-iteration, an upper bound on the number of inner-iterations that are needed to achieve the $\widetilde{\epsilon}_k$ is shown in the following theorem.

Theorem 4.3.1 (Inner-Loop)**.** Recall that a Lasso problem has a response vector $\mathbf{y} \in \mathbb{R}^n$ and a model matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$. To minimize the Lasso objective function $F(\boldsymbol{\beta}) = \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1$, we design a homotopic approach, i.e., in the kth outer-iteration of our proposed algorithm, we use AGD algorithm to minimize a surrogate function $F_{t_k}(\boldsymbol{\beta}) = \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda f_{t_k}(\boldsymbol{\beta})$. Instead of converging to the minimizer of $F_{t_k}(\boldsymbol{\beta})$, we do an early stopping to control the total number of numerical operations. And we denote the early stopping estimation as $\boldsymbol{\beta}^{(k)[s]}$, where s is the number of AGD-iterations (inner-iterations). We assume that for any $k = 1, 2, \ldots, s = 1, 2, \ldots$, we have $\left|\beta_i^{(k)[s]}\right| \leq B$, where $\beta_i^{(k)[s]}$ is the ith entry of vector $\boldsymbol{\beta}^{(k)[s]}$ ($i = 1, 2, \ldots, p$), and B is a constant. And we further assume that our proposed algorithm stops when $t_k < \tau$. Under the above assumption, we know that in the kth outer-iteration, the condition number of function $F_{t_k}(\cdot)$ can be bounded by $\frac{3B^3\lambda_{\max}\left(\frac{\mathbf{x}^\top\mathbf{x}}{n}\right)}{2\lambda[\log(1+\tau)]^2} + \left(\frac{B}{\tau}\right)^3$. Accordingly, after $C_1\log(1/\widetilde{\epsilon}_k)$ inner-iterations, one is guaranteed to achieve the following precision

$$F_{t_k}(\boldsymbol{\beta}^{(k)}) - F_{\min,k} \leq \widetilde{\epsilon}_k,$$

where $F_{\min,k} = \min_{\boldsymbol{\beta}} F_{t_k}(\boldsymbol{\beta})$, $\widetilde{\epsilon}_k = \frac{\lambda p}{3B}\left[\log(1+t_k)\right]^2$ and C_1 is a constant that does not depend on the value of t_k.

Proof. See subsubsection 4.6.4. □

Remark 4.3.1. Our assumption that the entries of $\boldsymbol{\beta}$ are bounded by a constant B is reasonable, as such a condition has appeared widely in the literature. A popular way to justify is that the values have to be manageable in a modern computer, which are restricted by the largest value that can be stored in the corresponding computer platform.

Remark 4.3.2. In the above theorem, we need a condition: $t_k > \tau$, where $\tau > 0$ is a predetermined constant. This condition prevents function $f_{t_k}(\boldsymbol{\beta})$ from converging to the function $\|\boldsymbol{\beta}\|$. In subsection 4.5.1, we will argue that our result applies in the "warm-up" stage of a Lasso-algorithm. That is, when t_k is small enough, under some conditions, the stopping point of our algorithm provides an estimator that is close enough to the ultimate Lasso estimator; therefore from our estimator, we may reliably estimate the support of the ultimate Lasso estimator, and simply run an ordinary regression on this support set. In this sense, our result finds a "warm start" for solving the Lasso problems, at the same time, achieves a provable faster convergence rate.

4.3.2 Number of Outer-Iteration

This section discusses the minimal number of outer-iterations needed to achieve the ϵ-precision defined in Definition 4.1.1, which explains the line 2 in algorithm 10.

Theorem 4.3.2 (Number of outer-iteration)**.** With the conditions in Theorem 4.3.1 being satisfied, and suppose the following conditions are also satisfied:

1. For $k = 1, 2, \ldots$, we have $t_k = t_0(1-h)^k$ where t_0, h are pre-specified.
2. The precision of AGD in minimizing function $F_{t_k}(\boldsymbol{\beta})$ is set as $\widetilde{\epsilon}_k = \frac{\lambda p}{3B}[\log(1+t_k)]^2$, i.e., we run the AGD until the following inequality is achieved: $F_{t_k}(\boldsymbol{\beta}^{(k)[s]}) - F_{k,\min} <$

$\widetilde{\epsilon}_k$, where quantity $\boldsymbol{\beta}^{(k)[s]}$ is the iterative estimator in the sth inner-iteration at the kth outer-iteration, and recall that $F_{k,\min} = \min_{\boldsymbol{\beta}} F_{t_k}(\boldsymbol{\beta})$.

Then when $k \geq \frac{-1}{\log(1-h)} \log\left(\frac{\lambda p t_0(2B+1)}{\epsilon}\right)$, our proposed algorithm finds a point $\boldsymbol{\beta}^{(k)}$ such that

$$F(\boldsymbol{\beta}^{(k)}) - F_{\min} \leq \epsilon,$$

where $F_{\min} = \min_{\boldsymbol{\beta}} F(\boldsymbol{\beta})$ with $F(\boldsymbol{\beta}) = \frac{1}{2n} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1$, which is defined in Equation 4.2.

Proof. The proof is shown in subsubsection 4.6.4. □

4.3.3 Order of complexity for HS Algorithm

With all the above blocks, we develop the main theory, i.e., the order of complexity, in this section. Recall that, the definition of order of complexity is the total number of operations needed to achieve the ϵ-precision. The reason for us to adopt the order of complexity instead of the running time is that the order of complexity is independent of (different) computer platforms, while running time possibly depends on different platforms. Consequently, the order of complexity provides a more reliable way for us to compare different algorithms.

Theorem 4.3.3 (Main Theory)**.** Under the conditions that are listed in Theorem 4.3.1 and Theorem 4.3.2, we can find $\boldsymbol{\beta}^{(k)}$ such that

$$F(\boldsymbol{\beta}^{(k)}) - F_{\min} \leq \epsilon$$

with the number of numerical operations has the order of complexity

$$p^2 O\left(\left[\frac{-1}{\log(1-h)} \log\left(\frac{\lambda p t_0(2B+1)}{\epsilon}\right)\right]^2\right).$$

Proof. See subsubsection 4.6.4. □

4.4 Numerical Examples

In this section, we compare the performance of HS algorithm with other state-of-the-art algorithms through numerical experiments. As we mentioned in section 4.1, there are many Lasso-algorithms. Proximal mapping appears to be a major tool in developing Lasso-algorithms. We take ISTA [see 55] as a representative. Starting from the proximal mapping, some Lasso-algorithms utilize the accelerated gradient descent, and develop a faster approach in proximal mapping. For this type of Lasso-algorithms, we select FISTA [see 24] as a representative. For the third type of Lasso-algorithms, we adopt coordinate descent. And the last type of Lasso-algorithms apply surrogate functions to approximate the ℓ_1 penalty; here we select SL [see 121] as an representative. Based on the theoretically analysis in section 4.1 and subsection 4.6.1, we see that ISTA [see 55], CD [see 56] and SL [see 121] share the same order of complexity, so we will only pick ISTA as a representative of these three. FISTA [see 24] is also selected as a benchmark since it has the best order of complexity among the existing Lasso-algorithms.

In this section, two numerical examples are shown. The difference between the two simulations lies on the setting of the true parameter $\boldsymbol{\beta}$. In first simulation, the ith entry of the true parameter $\boldsymbol{\beta} \in \mathbb{R}^p$ is generated by $\beta_i = (-1)^i \exp(-2(i-1)/20)$ for $i = 1, \ldots, p$. This style of parameter follows an traditional fashion, which is similar to [56]. In our second simulation, we set $\beta_i = (-1)^i \exp(-2(i-1)/20) \mathbb{1}\{i \leq 10\}$. This parameter setting assumes that most of the entries in $\boldsymbol{\beta}$ is zero, which renders a case with sparse truth.

4.4.1 Simulation 1

The objective in this numerical example is to explore whether HS has better performance than the benchmarks when estimating the true parameter under the sparse linear regression model. For a fair comparison, the simulation setting and data generation mechanism is similar to [56]. The difference is that in [56], the running time is compared under different

simulation setting, while we adopt the number of numeric operations here. (Recall that running time may depend on the platforms, while the number of numerical operations does not.)

The data generation mechanism is as follows. We generate Gaussian data with n observations and p covariates, with each predictor is associated with a random vector $\mathbf{x}_j \in \mathbb{R}^n$, and the model matrix is $\mathbf{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_j, \cdots, \mathbf{x}_p)$. Here we assume that the random vector $\mathbf{x}_j$ follows the multivariate normal distribution with zero mean, variances being equal to 1, and identical population correlation ρ, that is, the covariance matrix of $\mathbf{x}_j$ is of the following form:

$$\begin{pmatrix} 1 & \rho & \dots & \rho \\ \rho & 1 & \dots & \rho \\ \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \dots & 1 \end{pmatrix}.$$

In this simulation, we set $\rho = 0.1$. The response values were generated by

$$\mathbf{y} = \sum_{j=1}^{p} \mathbf{x}_j \beta_j + q\mathbf{z}, \tag{4.10}$$

where the ith ($1 \leq i \leq p$) entry in vector $\boldsymbol{\beta} = (\beta_1 \cdots \beta_p)^\top$ is generated by

$$\beta_i = (-1)^i \exp\left(-2(i-1)/20\right),$$

which are constructed to have alternating signs and to be exponentially decreasing. Besides, $\mathbf{z} = (z_1 \cdots z_p)^\top$ is the white noise with z_i satisfying the standard normal distribution $N(0, 1)$. quantity q is chosen so that the signal-to-noise ratio is 3.0. The turning parameter λ is set to be 10^{-3}. And in our simulation, two scenarios are discussed, where the first scenarios is $n = 50, p = 20$ and the second scenario is $n = 50, p = 80$.

Table 4.2 summarizes the numerical results of the number of operations for ISTA, FISTA, and our algorithm to achieve the different ϵ-precision. And Figure 4.3 visualizes

the numerical results in Table 4.2, where the blue line, red line, and yellow line represent the number of numerical operations of ISTA, FISTA, and our method, respectively. The x-axis is the $\log(1/\epsilon)$ (Recall ϵ in Equation 4.3). And y-axis is the logarithms of the number of numerical operations to achieve the corresponding ϵ-precision.

In both two scenarios, i.e., $n = 50, p = 20$ and $n = 50, p = 80$, there are some common properties of these three methods (ISTA, FISTA, and Ours). Generally speaking, as the precision ϵ approaches to 0, it costs more number of numerical operations for the designed algorithms to get the optimizer that achieves the desired precision. Therefore, no matter ISTA, FISTA or our method, the common characteristic of Figure 4.3 is that, both three methods have an increasing tendency.

In both two scenarios, i.e., $n = 50, p = 20$ and $n = 50, p = 80$, there are also some differences among these three methods (ISTA, FISTA, and Ours). Generally speaking, both ISTA and FISTA requires larger number of numerical operations than that of our method. For example, in the second scenario, when ϵ is 0.005, our method only requires $36,457$ operations, however, ISTA and FISTA need $190,131$ and $55,133$ operations, respectively. So, it is obvious that our method, compared with the state-of-the-art Lasso-algorithms (where FISTA is the most efficient one), requires less number of numerical operations to achieve the same ϵ-precision. In the first scenario ($n = 50, p = 20$) with large ϵ, the number of numerical operations of the three types of algorithms are very similar, because when ϵ and p are very small, the hidden constant before the complexity ($O(p^2/\epsilon)$ for ISTA, $O(p^2/\sqrt{\epsilon})$ for FISTA, and $O(p^2(\log(1/\epsilon))^2$ for HS) are dominated.

The pattern in Figure 4.3 matches our theoretical results. As we have shown in section 4.1, the number of numerical operations of ISTA and FISTA are $O(1/\epsilon)$ and $O(1/\sqrt{\epsilon})$, respectively. If we take the logarithm of these two number of numerical operations, then they ought to be $O\left(\log(1/\epsilon)\right)$ and $O\left(\frac{1}{2}\log(1/\epsilon)\right)$. Therefore, in principle, the slop of ISTA and FISTA in Figure 4.3 should be 1 and $\frac{1}{2}$, respectively. To verify this conjecture, we perform a linear regression and find that the slop of the ISTA curve in the right panel of

Figure 4.3 is 0.71411, and the slop of the FISTA curve in the left panel of Figure 4.3 is 0.25211. It should be acknowledged that there is some deviation of the slops from the theoretically predicted values, when comparing the numerical results with the theoretical analysis. The bias in both two scenarios is because ϵ is not small enough. If we decrease ϵ to 0, then the bias would be reduced, because the term related to ϵ in the complexity, i.e., $O(p^2/\epsilon)$ for ISTA, $O(p^2/\sqrt{\epsilon})$ for FISTA, and $O(p^2(\log(1/\epsilon))^2)$ for HS, will be dominated. For our proposed algorithm, whose computational complexity is $O\left([\log(1/\epsilon)]^2\right)$, its shape in Figure 4.3 should be similar to $\log(2\log(x))$ theoretically. Yet, in real practice, it is difficult to achieve this ideal computational complexity, because in each outer-iteration, it is difficult to know exactly when the inner-iteration should stop (see line 7 in algorithm 10). However, through the optimal computational complexity is hard to achieve in real practice, the computational complexity of our propose algorithm is still lower than that of ISTA and FISTA.

Table 4.2: Numerical complexity of ISTA, FISTA, HS in the first simulation

	Precision ϵ								
method	0.05	0.03	0.02	0.01	0.009	0.008	0.007	0.006	0.005
	$n = 50,\ p = 20$								
ISTA	$5,070$	$6,016$	$7,005$	$9,585$	$10,101$	$10,703$	$11,434$	$12,294$	$13,369$
FISTA	$4,781$	$5,117$	$5,453$	$6,461$	$6,685$	$6,797$	$7,021$	$7,133$	$7,357$
Ours	$5,478$	$5,478$	$5,479$	$5,479$	$5,479$	$5,479$	$5,479$	$5,479$	$6,005$
	$n = 50,\ p = 80$								
ISTA	$37,400$	$50,277$	$65,273$	$109,772$	$119,226$	$130,799$	$145,306$	$164,377$	$190,131$
FISTA	$31,237$	$34,533$	$37,417$	$45,657$	$47,305$	$48,541$	$50,189$	$52,249$	$55,133$
Ours	$30,919$	$30,919$	$32,765$	$34,611$	$34,611$	$34,611$	$36,457$	$36,457$	$36,457$

[1] There is the parameters settings of our HS algorithm: $t_0 = 3, h = 0.1, \lambda = 1e-3, \boldsymbol{\beta}^{(0)} = \mathbf{1}_{p\times 1}$.

4.4.2 Simulation 2

In this section, we discuss another simulation setting different from that one in subsection 4.4.1. Here, we still focus on the regression, where data is generated by following the formulation in Equation 4.10. The $\mathbf{y}, \mathbf{X}, q, \mathbf{z}$ are generated the same way as in subsection 4.4.1, i.e., the jth column in $\mathbf{X}$ has the same population correlation $\rho = 0.1$, and $\mathbf{z} = (z_1 \cdots z_p)^\top$ is the white noise with normal distribution of z_i as $N(0, 1)$ and

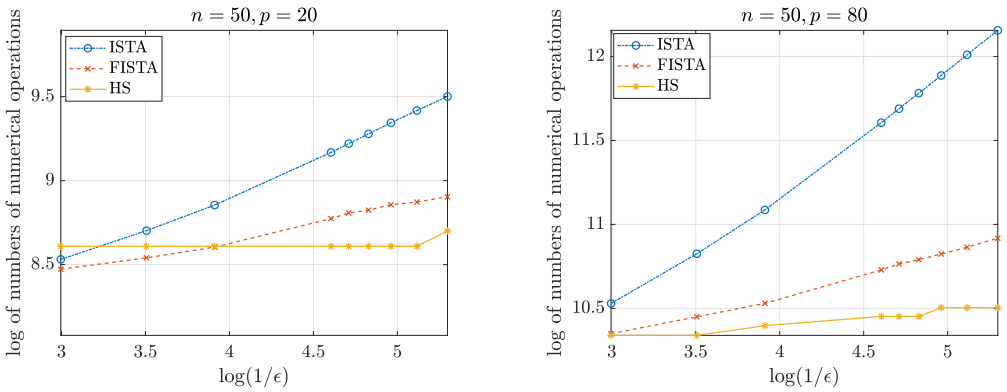


Figure 4.2: Number of Operations of ISTA, FISTA, and our algorithm under different ϵ in the first simulation

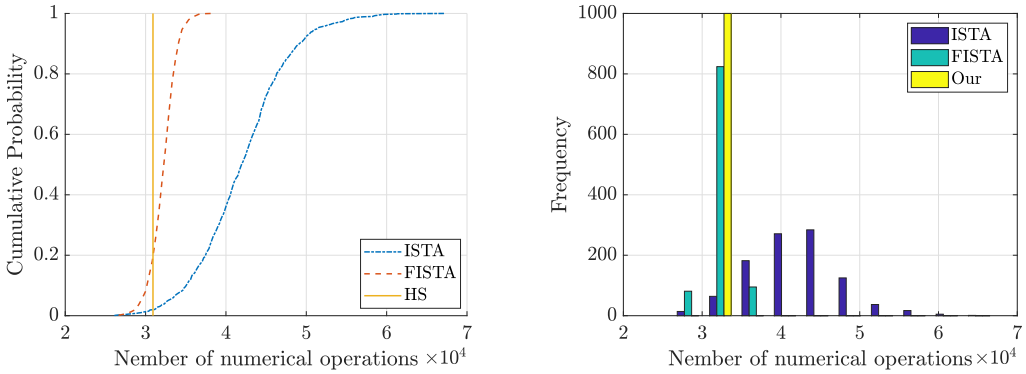


Figure 4.3: Empirical cumulative distribution function (left) and histogram (right) of the 1000 simulations in the first numerical example.

q is chosen so that the signal-to-noise ratio is 3.0. The only difference between this simulation to the one in subsection 4.4.1 lies in the setting of $\boldsymbol{\beta}$. Here, we only set the first 10 entries in $\boldsymbol{\beta}$ as non-zero, and the remaining entries in $\boldsymbol{\beta}$ is set as 0, i.e., $\beta_i = (-1)^i \exp\left[-2(i-1)/20\right] \mathbb{1}\{i \leq 10\}$.

Table 4.3 summarizes the numerical results of the number of operations for ISTA, FISTA, and our algorithm to achieve the different ϵ-precisions. And Figure 4.4 visualizes the numerical results in Figure 4.4, where the blue line, red line, and yellow line represent the number of numerical operations of ISTA, FISTA, and our method, respectively. The x-axis is the $\log(1/\epsilon)$ (Recall ϵ in Equation 4.3). And y-axis is the logarithms of the number of numerical operations to achieve the corresponding ϵ-precision.

In both scenarios, i.e., $n = 50, p = 20$ and $n = 50, p = 80$, we can see that, generally speaking, both ISTA and FISTA requires more number of numerical operations than that of our method. For example, for the second scenario, when ϵ is 0.005, our method only requires $32,763$ operations, however, ISTA and FISTA need $179,373$ and $53,485$ operations, respectively. So, it is obvious that our method, compared with the state-of-the-art Lasso-algorithms (where FISTA is the most efficient one), requires less number of numerical operations to achieve the same ϵ-precision. For the first scencario ($n = 50, p = 20$) with large ϵ, the number of numerical operations of the three types of algorithms are very similar, because when ϵ and p are very small, the hidden constant before the complexity ($O(p^2/\epsilon)$ for ISTA, $O(p^2/\sqrt{\epsilon})$ for FISTA, and $O(p^2/(\log(1/\epsilon))^2)$ for HS) are dominated.

Table 4.3: Numerical complexity of ISTA, FISTA, HS in the second simulation

	Precision ϵ								
method	0.05	0.03	0.02	0.01	0.009	0.008	0.007	0.006	0.005
	$n = 50,\ p = 20$								
ISTA	$5,242$	$6,274$	$7,263$	$9,370$	$9,757$	$10,187$	$10,703$	$11,305$	$12,122$
FISTA	$4,781$	$5,229$	$5,565$	$6,125$	$6,349$	$6,461$	$6,573$	$6,797$	$7,021$
Ours	$5,479$	$5,479$	$5,479$	$6,005$	$6,005$	$6,005$	$6,005$	$6,005$	$6,005$
	$n = 50,\ p = 80$								
ISTA	$39,519$	$55,330$	$72,119$	$112,869$	$120,693$	$130,473$	$142,698$	$158,346$	$179,373$
FISTA	$31,649$	$35,769$	$39,065$	$45,657$	$46,893$	$48,129$	$49,365$	$51,013$	$53,485$
Ours	$30,918$	$32,763$	$32,763$	$32,763$	$32,763$	$32,763$	$32,763$	$32,763$	$32,763$

[1] The parameters settings of our HS algorithm: $t_0 = 3, h = 0.1, \lambda = 1e-3, \boldsymbol{\beta}^{(0)} = 0.1 \times \mathbf{1}_{p\times 1}$

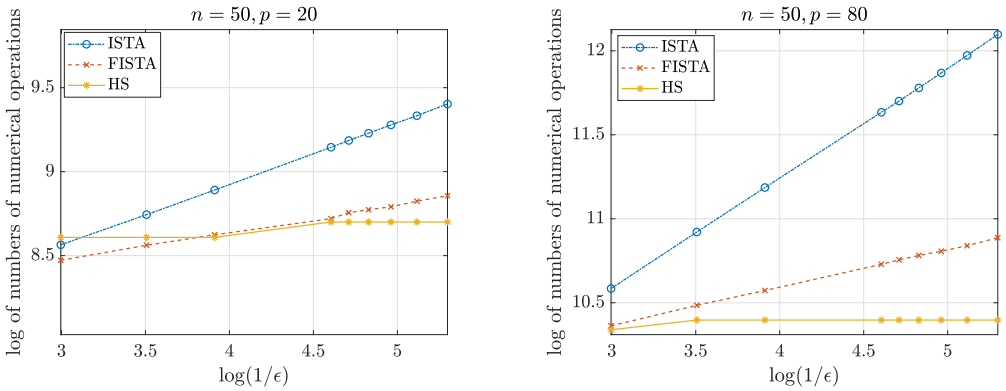


Figure 4.4: Number of Operations of ISTA, FISTA, and our algorithm under different ϵ in the second simulation.

4.5 Discussion

In our theoretical result, we required the presence of a constant $\tau > 0$, such that $t_k \geq \tau$ for all k. Such a condition prevents the hyper-parameter t from converge to zero. In subsection 4.5.1, we show that when the τ is chosen to be small enough, an early-stopped homotopic approach will find the support of the global solution, therefore, one can simply run the ordinary regression on this support set, without losing anything.

In subsection 4.5.2, we discuss other seemingly similar homotopic ideas, and articulate the differences between theirs and the work that is presented in this paper.

4.5.1 Support Recovery and the Need for Hyper-parameter t to Converge to Zero

In Theorem 4.3.1, we assume that there is a constant $\tau > 0$, such that $t_k \geq \tau$ for all k. Such a condition prevents the hyper-parameter t from converge to zero. Therefore, our result just applies to the warm-up stage of a homotopic approach in solving the Lasso problem. In this subsection, we show that under some standard conditions that have appeared in the literature, as long as we set τ to be small enough, the associated algorithm will find a solution that both has small "prediction error" and "estimation error". The mathematical

meaning of "prediction error" is

$$\frac{1}{n}\left\|\mathbf{X}\left(\widetilde{\boldsymbol{\beta}}-\widehat{\boldsymbol{\beta}}\right)\right\|_2^2,$$

where $\widetilde{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2n}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda f_t(\boldsymbol{\beta})$ for a general t and $f_t(\boldsymbol{\beta})$ defined in Equation 4.5, and $\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2n}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1$. And the mathematical meaning of "estimation error" in our paper is

$$\left\|\widetilde{\boldsymbol{\beta}}-\widehat{\boldsymbol{\beta}}\right\|_2^2.$$

In the remaining of this section, we will give two propositions, where we develop the conditions where we would have small prediction error and estimation error, respectively.

We begin with the prediction error, i.e., $\frac{1}{n}\left\|\mathbf{X}\left(\widetilde{\boldsymbol{\beta}}-\widehat{\boldsymbol{\beta}}\right)\right\|_2^2$. In the Proposition 4.5.1, we declare that there is no additional conditions needed to guarantee the small prediction error. That is, as long as we converge $t \to 0$, our proposed algorithm can guarantee the prediction error goes to zero as well.

Proposition 4.5.1. For our proposed algorithm, when $t \to 0$, we have the perdition error $\frac{1}{n}\left\|\mathbf{X}\left(\widetilde{\boldsymbol{\beta}}-\widehat{\boldsymbol{\beta}}\right)\right\|_2^2 \to 0$, where $\widetilde{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2n}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda f_t(\boldsymbol{\beta})$ for a general t and $f_t(\boldsymbol{\beta})$ defined in Equation 4.5. And $\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2n}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1$.

Proof. See subsubsection 4.6.4. □

After developing the prediction error, we now discuss the estimation error. A nice property of Lasso is that, it can potentially achieve the sparse estimation when $n < p$, i.e., most of the entries in the Lasso estimator $\widehat{\boldsymbol{\beta}}$ are zero, and only few of them are non-zero. The index set of these non-zero entries are called *support set*, i.e., $\mathcal{S} = \{i : \text{if } \widehat{\beta}_i \neq 0; \forall\, i = 1, 2, \ldots, p\}$. To show how Lasso can realize the sparse estimation, we take $\mathbf{X} = \mathbf{I}$ as an illustration example, where $\mathbf{I}$ is the identity matrix. (More complicated model matrix $\mathbf{X}$ can also be used, but here we use $\mathbf{X} = \mathbf{I}$ to create an example.) Then we have the linear

regression model as

$$\mathbf{y} = \boldsymbol{\beta} + \mathbf{w},$$

where $\mathbf{y}$ is the response vector, and $\mathbf{w}$ is the white noise. The Lasso estimator of the above linear regression model is

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2n} \|\mathbf{y} - \boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1 .$$

It can be verified that

$$\widehat{\beta}_i = \begin{cases} \text{sign}(y_i)(|y_i| - n\lambda), & \text{if } |y_i| > n\lambda; \\ 0, & \text{otherwise,} \end{cases}$$

is the solution of the Lasso problem. Note that $\widehat{\boldsymbol{\beta}}$ is sparse if $\mathbf{y}$ has many components with small magnitudes. However, if we consider $f_t(\boldsymbol{\beta})$, instead of the ℓ_1 penalty $\|\boldsymbol{\beta}\|_1$, we have

$$\widetilde{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2n} \|\mathbf{y} - \boldsymbol{\beta}\|_2^2 + \lambda f_t(\boldsymbol{\beta}).$$

We can show that $\widetilde{\beta}_i = 0$ if and only if $y_i = 0$. This shows that $\widetilde{\boldsymbol{\beta}}$ is not guaranteed to be sparse.

Although $\widetilde{\boldsymbol{\beta}}$ is not sparse, we can still verify that $\widetilde{\boldsymbol{\beta}}$ has very small estimation error under some specific assumptions of the model matrix $\mathbf{X}$.

Proposition 4.5.2. Suppose the model matrix $\mathbf{X}$ in the Lasso problem has the following three properties:

1. $\left\| \left(\mathbf{X}_{\mathcal{S}}^\top \mathbf{X}_{\mathcal{S}}\right)^{-1} \mathbf{X}_{\mathcal{S}}^\top \right\|_F$ can be bounded by a constant, where $\mathcal{S} = \{i : \widehat{\beta}_i \neq 0, \forall i = 1, 2, \ldots, p\}$ with $\widehat{\boldsymbol{\beta}} = \frac{1}{2n} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1$. And $\|\cdot\|_F$ is the Frobenius norm defined as $\|\mathbf{A}_{m\times n}\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n |A_{ij}|^2}$, where A_{ij} is the (i,j)th entry in matrix $\mathbf{A}$.

2. $\left\|\mathbf{X}_{\mathcal{S}^c}^{\dagger}\right\|_F$ can be bounded by a constant, where $\mathcal{S}^c$ is the complement set of $\mathcal{S}$. And $\mathbf{X}_{\mathcal{S}^c}^{\dagger}$ is the pseudo-inverse of matrix $\mathbf{X}_{\mathcal{S}^c}$. The mathematical meaning of pseudo-inverse is that, suppose $\mathbf{X}_{\mathcal{S}^c} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}$, which is the singular value decomposition (SVD) of $\mathbf{X}_{\mathcal{S}^c}$. Then $\mathbf{X}_{\mathcal{S}^c}^{\dagger} = \mathbf{V}^{\top}\mathbf{\Sigma}^{\dagger}\mathbf{U}^{\top}$. For the rectangular diagonal matrix $\mathbf{\Sigma}$, we get $\mathbf{\Sigma}^{\dagger}$ by taking the reciprocal of each non-zero elements on the diagonal, leaving the zeros in place, and then transposing the matrix.

3. $\sigma_{\max}(\mathbf{\Sigma}_1) < \min\{2, 2\sigma_{\min}(\mathbf{\Sigma}_2)\}$, where $\sigma_{\max}(\mathbf{\Sigma}_1)$ returns the maximal absolute diagonal values of matrix $\mathbf{\Sigma}_1$, and $\sigma_{\min}(\mathbf{\Sigma}_2)$ returns the minimal absolute diagonal values of matrix $\mathbf{\Sigma}_2$. Matrix $\mathbf{\Sigma}_1$ is the diagonal matrix in the SVD of matrix $\left(\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}}\right)^{-1}\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}^c} + \left(\mathbf{X}_{\mathcal{S}^c}^{\dagger}\mathbf{X}_{\mathcal{S}}\right)^{\top}$, i.e.,

$$\left(\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}}\right)^{-1}\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}^c} + \left(\mathbf{X}_{\mathcal{S}^c}^{\dagger}\mathbf{X}_{\mathcal{S}}\right)^{\top} = \mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1.$$

Matrix $\mathbf{\Sigma}_2$ is the diagonal matrix of the SVD of matrix $\frac{1}{2}\mathbf{X}_{\mathcal{S}^c}^{\dagger}\mathbf{X}_{\mathcal{S}^c} + \frac{1}{2}\left(\mathbf{X}_{\mathcal{S}^c}^{\dagger}\mathbf{X}_{\mathcal{S}^c}\right)^{\top}$, i.e.,

$$\frac{1}{2}\mathbf{X}_{\mathcal{S}^c}^{\dagger}\mathbf{X}_{\mathcal{S}^c} + \frac{1}{2}\left(\mathbf{X}_{\mathcal{S}^c}^{\dagger}\mathbf{X}_{\mathcal{S}^c}\right)^{\top} = \mathbf{U}_2\mathbf{\Sigma}_2\mathbf{V}_2.$$

Then we have $\left\|\widetilde{\boldsymbol{\beta}} - \widehat{\boldsymbol{\beta}}\right\|_2^2 \to 0$ when $t \to 0$.

Proof. See subsubsection 4.6.4. □

We notice that the above proposition requires a strong condition on the model matrix $\mathbf{X}$ in order to achieve the support recovery. Releasing the conditions in the above proposition is an interesting future research topic.

4.5.2 Other Related Homotopic Ideas

It is worth noting that, in the recent research, some researchers also realize the log-polynomial order of complexity [see 126, 127, 128, 129, 130] in a framework similar to Lasso-algorithms. However, we would like to clarify that, there are some essential differences between our

paper and these papers. First, the problem formulation in these papers is totally different from ours. The problem formulation these papers solve is that, they start at some initial objective problem, which is a Lasso-type objective function:

$$\frac{1}{2n}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2+\lambda^{(0)}\|\boldsymbol{\beta}\|_1, \tag{4.11}$$

and then they gradually decrease the large $\lambda^{(0)}$ until the target regularization $\lambda^{(\text{target})}$ is reached. When the $\lambda^{(\text{target})}$ is reached, the algorithm is stopped. However, this algorithmic solution is not the optimal in Equation 4.11. In other words, the solution of these papers is not exactly the Lasso solution. While in our paper, our objective function stays the same as Equation 4.2 from the beginning to the end of our algorithm. Therefore, the solution we iteratively calculated is the minimizer of the Lasso problem in Equation 4.2. In addition to the difference of the objective function, the assumptions between our algorithm and these papers are also different. Specifically, these papers require more additional assumptions than us, such as the *restricted isometry property (RIP)*, which is used to ensure that the all solution path is sparse. Finally, through both our paper and these papers are called "homotopic" method, the definition of the "homotopic" is different. Specifically, these papers use the homotopic path in the penalty parameter λ: they start from a very large λ and then shrinkage to the target λ. This type of method is also called "path following" in other papers, such as [123, 131, 132], and etc, instead of "homotopic path". However, our paper use the homotopic path in the ℓ_1 penalty $\lambda\|\boldsymbol{\beta}\|_1$: we replace the ℓ_1 regularization term with a surrogate function, and then by adjusting the parameters in the surrogates, to get the surrogate approximates closer to the original ℓ_1 regularization term.

4.6 Supplementary Material

4.6.1 Review of Some State-of-the-art Algorithms

In this section, we will show the algorithm mechanism of these four representative we select, namely ISTA [see 55], FISTA [24, see], CD [see 56], and SL [see 121]. For each algorithm, we show (i) their number of operations in an iteration, (ii) the number of iterations to meet the ϵ-precision in Equation 4.3, (iii) and their according order of complexity.

Iterative Shrinkage-Thresholding Algorithms (ISTA)

ISTA aims at the minimization of a summation of two functions, $g + f$, where the first function $g : \mathbb{R}^p \to \mathbb{R}$ is continuous convex and the other function $f : \mathbb{R}^p \to \mathbb{R}$ is smooth convex with a Lipschitz continuous gradient. Recall the definition of Lipschitz continuous gradient as follows:

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_2 \leq L\|\mathbf{x} - \mathbf{y}\|_2.$$

If we let $g(\boldsymbol{\beta}) = \lambda\|\boldsymbol{\beta}\|_1$ and $f(\boldsymbol{\beta}) = \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2$ with the Lipschitz continuous gradient L taking the largest eigenvalue of matrix $\mathbf{X}^\top\mathbf{X}/n$, noted as $\sigma_{\max}(\mathbf{X}^\top\mathbf{X}/n)$, then Lasso is a special case of ISTA.

The key point of ISTA lies in the updating rule from $\boldsymbol{\beta}^{(k)}$ to $\boldsymbol{\beta}^{(k+1)}$, i.e., $\boldsymbol{\beta}^{(k)} \to \boldsymbol{\beta}^{(k+1)}$. It is realized by updating $\boldsymbol{\beta}^{(k+1)}$ through the quadratic approximation function of $f(\boldsymbol{\beta})$ at value $\boldsymbol{\beta}^{(k)}$:

$$\boldsymbol{\beta}^{(k+1)} = \arg\min_{\boldsymbol{\beta}} f(\boldsymbol{\beta}^{(k)}) + \langle(\boldsymbol{\beta} - \boldsymbol{\beta}^{(k)}), \nabla f(\boldsymbol{\beta}^{(k)})\rangle + \frac{\sigma_{\max}(\mathbf{X}^\top\mathbf{X}/n)}{2}\|\boldsymbol{\beta} - \boldsymbol{\beta}^{(k)}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1. \tag{4.12}$$

Simple algebra shows that (ignoring constant terms in $\boldsymbol{\beta}$), minimization of equation Equa-

tion 4.12 is equivalent to the minimization problem in the following equation:

$$\boldsymbol{\beta}^{(k+1)} = \arg\min_{\boldsymbol{\beta}} \frac{\sigma_{\max}(\mathbf{X}^\top\mathbf{X}/n)}{2} \left\| \boldsymbol{\beta} - \left(\boldsymbol{\beta}^{(k)} - \frac{\frac{1}{n}(\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta}^{(k)} - \mathbf{X}^\top\mathbf{y})}{\sigma_{\max}(\mathbf{X}^\top\mathbf{X}/n)} \right) \right\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1, \tag{4.13}$$

where the soft-thresholding function in Equation 4.14 can be used to solve the problem in Equation 4.13:

$$S(x,\alpha) \quad = \quad \begin{cases} x-\alpha, & \text{if } x \geq \alpha, \\ x+\alpha, & \text{if } x \leq -\alpha, \\ 0, & \text{otherwise.} \end{cases} \tag{4.14}$$

The summary of ISTA algorithm is presented in algorithm 11.

Algorithm 11: Iterative Shrinkage-Thresholding Algorithms (ISTA)

Input: $\mathbf{y} \in \mathbb{R}^n, \mathbf{X} \in \mathbb{R}^{n\times p}$, $L = \sigma_{\max}(\mathbf{X}^\top\mathbf{X}/n)$
Output: an estimator of $\boldsymbol{\beta}$ satisfies the ϵ-precision, noted as $\boldsymbol{\beta}^{(k)}$
1 **initialization;**
2 $\boldsymbol{\beta}^{(0)}, k = 0$
3 **while** $F(\boldsymbol{\beta}^{(k)}) - F(\widehat{\boldsymbol{\beta}}) > \epsilon$ **do**
4 $\quad \boldsymbol{\beta}^{(k+1)} = S(\boldsymbol{\beta}^{(k)} - \frac{1}{nL}(\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta}^{(k)} - \mathbf{X}^\top\mathbf{y}), \lambda/L)$
5 $\quad k = k+1$

It can be seen from line 4 in algorithm 11 that the number of operations in one iteration of ISTA is $O(p^2)$. This is because that the main computation of each iteration in ISTA is the matrix multiplication in $\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta}^{(k)}$. Note that the matrix $\mathbf{X}^\top\mathbf{X}$ can be pre-calculated and saved, therefore, the order of computational complexity is $p(2p-1)$ [see 133].

In addition to the operations in each iteration, we also develop the convergence analysis of ISTA in the following equation [see 24, Theorem 3.1]. To make it more clear, we list Theorem 3.1 in [24, Theorem 3.1] below with several changes of notation. The notations are changed to be consistent with the terminology that are used in this paper.

Theorem 4.6.1. Let $\left\{\boldsymbol{\beta}^{(k)}\right\}$ be the sequence generated by line 4 in algorithm 11. Then for

any $k \geq 1$, we have

$$F(\boldsymbol{\beta}^{(k)}) - F(\widehat{\boldsymbol{\beta}}) \leq \frac{\sigma_{\max}(\mathbf{X}^\top \mathbf{X}/n)\|\boldsymbol{\beta}^{(0)} - \widehat{\boldsymbol{\beta}}\|_2^2}{2k}. \tag{4.15}$$

Therefore, to achieve the ϵ-precision, i.e., $F(\boldsymbol{\beta}^{(k)}) - F(\widehat{\boldsymbol{\beta}}) \leq \epsilon$, at least $\frac{\sigma_{\max}(\mathbf{X}^\top \mathbf{X}/n)\|\boldsymbol{\beta}^{(0)} - \widehat{\boldsymbol{\beta}}\|_2^2}{2\epsilon}$ iterations are required, which leads to the order of complexity $O(\frac{\sigma_{\max}(\mathbf{X}^\top \mathbf{X}/n)\|\boldsymbol{\beta}^{(0)} - \widehat{\boldsymbol{\beta}}\|_2^2}{2\epsilon} p^2) = O(p^2/\epsilon)$.

Fast Iterative Shrinkage-Thresholding Algorithms (FISTA)

Motivated by ISTA, [24] developed another algorithm called Fast Iterative Shrinkage-Thresholding Algorithms (FISTA). The main difference of ISTA and FISTA is that FISTA employs an auxiliary variable $\boldsymbol{\alpha}^{(k)}$ to update from $\boldsymbol{\beta}^{(k)}$ to $\boldsymbol{\beta}^{(k+1)}$ in the second-order Taylor expansion step (i.e., the one in Equation 4.12); More specifically, they have

$$\boldsymbol{\beta}^{(k+1)} = \arg\min_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}^{(k)}) + \langle (\boldsymbol{\alpha} - \boldsymbol{\alpha}^{(k)}), \nabla f(\boldsymbol{\alpha}^{(k)}) \rangle + \frac{\sigma_{\max}(\mathbf{X}^\top \mathbf{X}/n)}{2} \|\boldsymbol{\alpha} - \boldsymbol{\alpha}^{(k)}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_1, \tag{4.16}$$

where $\boldsymbol{\alpha}^{(k)}$ is a specific linear combination of the previous two estimator $\boldsymbol{\beta}^{(k-1)}, \boldsymbol{\beta}^{(k-2)}$, in particular, we have $\boldsymbol{\alpha}^{(k)} = \boldsymbol{\beta}^{(k-1)} + \frac{t_{k-1}-1}{t_k}(\boldsymbol{\beta}^{(k-1)} - \boldsymbol{\beta}^{(k-2)})$. FISTA falls in the framework of Accelerate Gradient Descent (AGD), as it takes additional past information to utilize an extra gradient step via the auxiliary sequence $\boldsymbol{\alpha}^{(k)}$, which is constructed by adding a "momentum" term $\boldsymbol{\beta}^{(k-1)} - \boldsymbol{\beta}^{(k-2)}$ that incorporates the effect of second-order changes. For completeness, the FISTA is shown in algorithm 12.

Obviously, the main computational effort in both ISTA and FISTA remains the same, namely, in the soft-thresholding operation of line 4 in algorithm 11 and algorithm 12. The number of operations in each iterations of FISTA is still $O(p^2)$. Although for both ISTA and FISTA, they have the same number of operation in one iteration, FISTA has improved convergence rate than ISTA, which is shown in the following theorem [see 24, Theorem 4.4].

Algorithm 12: Fast Iterative Shrinkage-Thresholding Algorithms (FISTA)

Input: $\mathbf{y} \in \mathbb{R}^n, \mathbf{X} \in \mathbb{R}^{n\times p}$, $L = \sigma_{\max}(\mathbf{X}^\top \mathbf{X}/n)$

Output: an estimator of $\boldsymbol{\beta}$, noted as $\boldsymbol{\beta}^{(k)}$, which satisfies the ϵ-precision.

1 **initialization;**

2 $\boldsymbol{\beta}^{(0)}$ **,** $t_1 = 1$**,** $k = 0$

3 **while** $F(\boldsymbol{\beta}^{(k)}) - F(\widehat{\boldsymbol{\beta}}) > \epsilon$ **do**

4 $\quad \boldsymbol{\beta}^{(k)} = S(\boldsymbol{\alpha}^{(k)} - \frac{1}{nL}(\mathbf{X}^\top \mathbf{X}\boldsymbol{\alpha}^{(k)} - \mathbf{X}^\top \mathbf{y}), \lambda/L)$

5 $\quad t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$

6 $\quad \boldsymbol{\alpha}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \frac{t_k - 1}{t_{k+1}}(\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}^{(k-1)})$

7 $\quad k = k + 1$

Theorem 4.6.2. Let $\left\{\boldsymbol{\alpha}^{(k)}\right\}, \left\{\boldsymbol{\beta}^{(k)}\right\}$ be a sequence generated by line 6 and line 4 in algorithm 12, respectively. Then for any $k \geq 1$, we have that

$$F(\boldsymbol{\beta}^{(k)}) - F(\widehat{\boldsymbol{\beta}}) \leq \frac{2\sigma_{\max}(\mathbf{X}^\top \mathbf{X}/n)\|\boldsymbol{\beta}^{(0)} - \widehat{\boldsymbol{\beta}}\|_2^2}{(k+1)^2}. \tag{4.17}$$

Consequently, FISTA has a faster convergence rate than ISTA, which improves from $O(1/k)$ to $O(1/k^2)$. This is because that, to update from $\boldsymbol{\beta}^{(k-1)}$ to $\boldsymbol{\beta}^{(k)}$, ISTA only considers $\boldsymbol{\beta}^{(k-1)}$, however, FISTA takes both $\boldsymbol{\beta}^{(k-1)}$ and $\boldsymbol{\beta}^{(k-2)}$ into account. To achieve the precision $F(\boldsymbol{\beta}^{(k)}) - F(\widehat{\boldsymbol{\beta}}) \leq \epsilon$, at least $\frac{2\sigma_{\max}(\mathbf{X}^\top \mathbf{X}/n)\|\boldsymbol{\beta}^{(0)} - \widehat{\boldsymbol{\beta}}\|_2^2}{\sqrt{\epsilon}}$ iterations are required, which leads to an order of complexity of $O(\frac{2\sigma_{\max}(\mathbf{X}^\top \mathbf{X}/n)\|\boldsymbol{\beta}^{(0)} - \widehat{\boldsymbol{\beta}}\|_2^2}{\sqrt{\epsilon}} p^2) = O(p^2/\sqrt{\epsilon})$.

Coordinate Descent (CD)

The updating rules in both ISTA and FISTA involve all coordinates simultaneously. In contrast, [56] proposes a Lasso-algorithm that cyclically chooses one coordinate at a time and performs a simple analytical update. Such an approach is called coordinate gradient descent.

The updating rule (from $\boldsymbol{\beta}^{(k)}$ to $\boldsymbol{\beta}^{(k+1)}$) in CD is that, it optimizes with respect to only the jth entry of $\boldsymbol{\beta}^{(k+1)}$ $(j = 1, \cdots, p)$ where the gradient at $\beta_j^{(k)}$ in the following equation

is used for the updating process:

$$\frac{\partial}{\partial \beta_j} F(\boldsymbol{\beta}^{(k)}) = \frac{1}{n}\left(\mathbf{e}_j^\top \mathbf{X}^\top \mathbf{X}\boldsymbol{\beta}^{(k)} - \mathbf{y}^\top \mathbf{X}\mathbf{e}_j\right) + \lambda \text{sign}(\beta_j) \tag{4.18}$$

where $\mathbf{e}_j$ is a vector of length p, whose entries are all zero expect that the jth entry is equal to 1. Imposing the gradient in Equation 4.18 to be 0, we can solve for $\beta_j^{(k+1)}$ as follows:

$$\beta_j^{(k+1)} = S\left(\mathbf{y}^\top \mathbf{X}\mathbf{e}_j - \sum_{l \neq j} \left(\mathbf{X}^\top \mathbf{X}\right)_{jl} \beta_l^{(k)}, n\lambda\right) \Big/ \left(\mathbf{X}^\top \mathbf{X}\right)_{jj},$$

where $S(\cdot)$ is the soft-thresholding function defined in Equation 4.14. This algorithm has been implemented into the a R package, *glmnet*, and we summarize it in algorithm 13.

Algorithm 13: Coordinate Descent(CD)

Input: $\mathbf{y} \in \mathbb{R}^n, \mathbf{X} \in \mathbb{R}^{n \times p}$, λ

Output: an estimator of $\boldsymbol{\beta}$, noted as $\boldsymbol{\beta}^{(k)}$, which satisfies the ϵ-precision.

1 **initialization;**

2 $\boldsymbol{\beta}^{(0)}, k = 0$

3 **while** $F(\boldsymbol{\beta}^{(k)}) - F(\widehat{\boldsymbol{\beta}}) > \epsilon$ **do**

4 **for** $j = 1 \cdots p$ **do**

5 $\beta_j^{(k+1)} = S\left(\mathbf{y}^\top \mathbf{X}\mathbf{e}_j - \sum_{l \neq j} \left(\mathbf{X}^\top \mathbf{X}\right)_{jl} \beta_l^{(k)}, n\lambda\right) \Big/ \left(\mathbf{X}^\top \mathbf{X}\right)_{jj}$

After reviewing the algorithm of CD, we develop the order of complexity of CD. Firstly, the number of operations in each iteration of CD is $O(p^2)$. It can be explained by the following two reasons. (i) While updating $\beta_j^{(k+1)}$ (line 5 in algorithm 13), it costs $O(p)$ operations because of $\sum_{l \neq j} \left(\mathbf{X}^\top \mathbf{X}\right)_{jl} \beta_j^{(k)}$. (ii) From line 4 in algorithm 13, we can see that all p entries of $\boldsymbol{\beta}^{(k+1)}$ are updated one by one. Combining (i) and (ii), we can see that the number of operations need in one iteration of CD is of the order $O(p^2)$.

The convergence rate of CD is derived as a corollary in [134, Corollary 3.8] and here we list the corollary as a theorem below. We changed several notations to adopt the terminology in this paper:

Theorem 4.6.3. Let $\left\{\boldsymbol{\beta}^{(k)}\right\}$ be the sequence generated by the line 5 in algorithm 13. Then we have that

$$F(\boldsymbol{\beta}^{(k)}) - F(\widehat{\boldsymbol{\beta}}) \leq \frac{4\sigma_{\max}(\mathbf{X}^\top\mathbf{X}/n)(1+p)\|\boldsymbol{\beta}^{(0)} - \widehat{\boldsymbol{\beta}}\|_2^2}{k + (8/p)}. \tag{4.19}$$

The above equation shows that, to achieve the precision ϵ-precision, at least

$$\frac{4\sigma_{\max}(\mathbf{X}^\top\mathbf{X}/n)(1+p)\|\boldsymbol{\beta}^{(0)} - \widehat{\boldsymbol{\beta}}\|_2^2}{\epsilon} - \frac{8}{p}$$

iterations are required, which leads to an order of complexity of

$$O\left(\left[\frac{4\sigma_{\max}(\mathbf{X}^\top\mathbf{X}/n)(1+p)\|\boldsymbol{\beta}^{(0)} - \widehat{\boldsymbol{\beta}}\|_2^2}{\epsilon} - \frac{8}{p}\right]p^2\right) = O(p^2/\epsilon - 8p) = O(p^2/\epsilon).$$

Smooth Lasso (SL)

The aforementioned Lasso-algorithms all aim exactly at minimizing the function $F(\boldsymbol{\beta})$. On the contrary, [121] uses an approximate objective function to solve the Lasso. Their method is called a Smooth-Lasso (SL) algorithm. The main idea of SL is that it use a smooth function—$\phi_\alpha(u) = \frac{2}{u}\log(1 + e^{\alpha u}) - u$— to approximate the ℓ_1 penalty, and the Accelerated Gradient Descent (AGD) algorithm is applied after the replacement. Therefore, the objective function of SL becomes $F_\alpha(\boldsymbol{\beta}) = \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\sum_{i=1}^p \phi_\alpha(\beta_i)$. The pseudo code of SL is displayed in algorithm 14.

For the computational effort, it mainly lies in the calculation of $\nabla F_\alpha(\mathbf{w}) = \frac{\mathbf{X}^\top\mathbf{X}}{n}\mathbf{w} - \frac{\mathbf{X}^\top\mathbf{y}}{n} + \mathbf{v}$, where the $\mathbf{v}$ is a vector of length p, whose ith entry is $\frac{-2}{w_i^2}\log(1 + e^{\alpha w_i}) + \frac{2\alpha e^{\alpha w_i}}{w_i(1+e^{\alpha w_i})} - 1$. Accordingly, the main computational effort of each iteration of SL is the matrix multiplication in $\mathbf{X}^\top\mathbf{X}\mathbf{w}^{(k)}$, which cost $O(p^2)$ operations. On the other side, proved by [121], the approximation error of $\boldsymbol{\beta}^{(k)}$ in SL is shown in Equation 4.20.

Theorem 4.6.4. Let $\left\{\boldsymbol{\beta}^{(k)}\right\}$ be a sequence generated as in line 5 of algorithm 14. Then we

Algorithm 14: Smooth Lasso (SL)

Input: $\mathbf{y} \in \mathbb{R}^n, \mathbf{X} \in \mathbb{R}^{n\times p}$, $\mu = [\sigma^2_{\max}(\mathbf{X}/\sqrt{n}) + \lambda\alpha/2]^{-1}$

Output: an estimator of $\boldsymbol{\beta}$, noted as $\boldsymbol{\beta}^{(k)}$, which satisfies the ϵ-precision.

initialization;

$\boldsymbol{\beta}^{(0)}$ **,** $k = 0$

while $F(\boldsymbol{\beta}^{(k)}) - F(\widehat{\boldsymbol{\beta}}) > \epsilon$ **do**

  $\mathbf{w}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \frac{k-2}{k+1}(\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}^{(k-1)})$

  $\boldsymbol{\beta}^{(k+1)} = \mathbf{w}^{(k+1)} - \mu\nabla F_\alpha(\mathbf{w}^{(k)})$

  $k = k + 1$

have

$$F(\boldsymbol{\beta}^{(k)}) - F(\widehat{\boldsymbol{\beta}}) \leq \frac{4\|\boldsymbol{\beta}^{(0)} - \widehat{\boldsymbol{\beta}}\|_2^2 \sigma^2_{\max}(\frac{\mathbf{X}}{\sqrt{n}})}{k^2} + \frac{4\sqrt{2\lambda n \log 2}\|\boldsymbol{\beta}^{(0)} - \widehat{\boldsymbol{\beta}}\|_2}{k}. \tag{4.20}$$

So to achieve the ϵ-precision, SL needs $O(1/\epsilon)$, which results in the order of complexity $O(p^2/\epsilon)$.

4.6.2 Path Following Lasso-Algorithm

As mentioned in section 4.1, the path following Lasso-algorithm has two drawbacks. First, it is not guaranteed to work in general cases. Second, there is no theoretical guarantee that the order of complexity of a path following Lasso-algorithm is low, considering that the maximum number of iterations can be as large as 2^p, where p is the number of predictors. In this section, we provide mathematical details to support the above two drawbacks. The structure of this section is described as follows. In the "Details to Support the First Drawback of Path Following Lasso Algorithm" section, we provide a counter example that the path following Lasso-algorithm is not workable, which represents a general category of design matrix $\mathbf{X}$ and coefficient $\boldsymbol{\beta}$. In the "Details to Support the Second Drawback of Path Following Lasso Algorithm" section, we provide mathematical details to support the second drawback of the path following Lasso-algorithm.

Details to Support the First Drawback of Path Following Lasso Algorithm

In this section, we provide a counter example that the path following Lasso-algorithm is not workable. This counter example represents a general category of design matrix $\mathbf{X}$ and coefficient $\boldsymbol{\beta}$. We use the following counter example to argue that a path following approach does not work in the most general setting.

Before representing the concrete counter example, let us discuss the key step in designing a path following Lasso-algorithm. For a general solution derived by path following Lasso-algorithm, i.e., $\widehat{\boldsymbol{\beta}}(\lambda)$, it is the minimizer of Equation 4.1, so it must satisfy the first order condition of Equation 4.1:

$$\mathbf{q} - \lambda \text{sign}(\widehat{\boldsymbol{\beta}}(\lambda)) = \mathbf{X}^\top \mathbf{X} \widehat{\boldsymbol{\beta}}(\lambda), \tag{4.21}$$

where $\mathbf{q} = \mathbf{X}^\top \mathbf{y}$ and is $\text{sign}(\widehat{\boldsymbol{\beta}}(\lambda))$ a vector, whose ith component is the sign function of $\widehat{\boldsymbol{\beta}}(\lambda)$:

$$\text{sign}(\beta_i(\lambda)) = \begin{cases} 1 & \text{if } \beta_i(\lambda) > 0 \\ -1 & \text{if } \beta_i(\lambda) < 0 \\ [-1, 1] & \text{if } \beta_i(\lambda) = 0 \end{cases} .$$

If we divide the indices of $\mathbf{q}, \boldsymbol{\beta}, \mathbf{X}$ into $\mathcal{S} = \{i : \widehat{\beta}_i(\lambda) \neq 0, \ \forall \ i = 1, \ldots, p\}$ and its complements $\mathcal{S}^c$, then we can rewrite Equation 4.21 as

$$\begin{pmatrix} \mathbf{q}_{\mathcal{S}} \\ \mathbf{q}_{\mathcal{S}^c} \end{pmatrix} - \begin{pmatrix} \lambda \text{sign}(\widehat{\boldsymbol{\beta}}_{\mathcal{S}}(\lambda)) \\ \lambda \text{sign}(\widehat{\boldsymbol{\beta}}_{\mathcal{S}^c}(\lambda)) \end{pmatrix} = \begin{pmatrix} \mathbf{X}_{\mathcal{S}}^\top \mathbf{X}_{\mathcal{S}} & \mathbf{X}_{\mathcal{S}}^\top \mathbf{X}_{\mathcal{S}^c} \\ \mathbf{X}_{\mathcal{S}^c}^\top \mathbf{X}_{\mathcal{S}} & \mathbf{X}_{\mathcal{S}^c}^\top \mathbf{X}_{\mathcal{S}^c} \end{pmatrix} \begin{pmatrix} \widehat{\boldsymbol{\beta}}_{\mathcal{S}}(\lambda) \\ \mathbf{0} \end{pmatrix},$$

where $\widehat{\boldsymbol{\beta}}_{\mathcal{S}}(\lambda)$ is the subvector of $\boldsymbol{\beta}$ only contains elements whose indices are in $\mathcal{S}$ and $\widehat{\boldsymbol{\beta}}_{\mathcal{S}^c}(\lambda)$ is the complement of $\boldsymbol{\beta}_{\mathcal{S}}$. Besides, $\text{sign}(\widehat{\boldsymbol{\beta}}_{\mathcal{S}}(\lambda))$ is the subset of $\text{sign}(\widehat{\boldsymbol{\beta}}(\lambda))$, only contains the elements whose indices are in $\mathcal{S}$, and $\text{sign}(\widehat{\boldsymbol{\beta}}_{\mathcal{S}^c}(\lambda))$ is the complement to $\text{sign}(\widehat{\boldsymbol{\beta}}_{\mathcal{S}}(\lambda))$. Matrix $\mathbf{X}_{\mathcal{S}}$ is the columns of $\mathbf{X}$ whose indices are in $\mathcal{S}$, and $\mathbf{X}_{\mathcal{S}^c}$ is the

complement of $\mathbf{X}_{\mathcal{S}}$.

Suppose we are interested in parameter estimated under λ and $\lambda - \Delta(\Delta \in (0, \lambda))$, i.e., $\widehat{\boldsymbol{\beta}}(\lambda), \widehat{\boldsymbol{\beta}}(\lambda - \Delta)$. Then $\widehat{\boldsymbol{\beta}}(\lambda), \widehat{\boldsymbol{\beta}}(\lambda - \Delta)$ must satisfy the following two system of equations:

$$\begin{cases} \mathbf{q}_{\mathcal{S}} - \lambda \text{sign}(\widehat{\boldsymbol{\beta}}_{\mathcal{S}}(\lambda)) &= \mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}}\widehat{\boldsymbol{\beta}}_{\mathcal{S}}(\lambda) \\ \mathbf{q}_{\mathcal{S}^c} - \lambda \text{sign}(\widehat{\boldsymbol{\beta}}_{\mathcal{S}^c}(\lambda)) &= \mathbf{X}_{\mathcal{S}^c}^{\top}\mathbf{X}_{\mathcal{S}}\widehat{\boldsymbol{\beta}}_{\mathcal{S}}(\lambda) \end{cases}, \tag{4.22}$$

$$\begin{cases} \mathbf{q}_{\mathcal{S}} - (\lambda - \Delta) \text{sign}(\widehat{\boldsymbol{\beta}}_{\mathcal{S}}(\lambda - \Delta)) &= \mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}}\widehat{\boldsymbol{\beta}}_{\mathcal{S}}(\lambda - \Delta) \\ \mathbf{q}_{\mathcal{S}^c} - (\lambda - \Delta) \text{sign}(\widehat{\boldsymbol{\beta}}_{\mathcal{S}^c}(\lambda - \Delta)) &= \mathbf{X}_{\mathcal{S}^c}^{\top}\mathbf{X}_{\mathcal{S}}\widehat{\boldsymbol{\beta}}_{\mathcal{S}}(\lambda - \Delta) \end{cases}. \tag{4.23}$$

From the above two system of equations, we have the following:

$$-(\lambda - \Delta)\text{sign}(\widehat{\boldsymbol{\beta}}_{\mathcal{S}^c}(\lambda - \Delta)) = -\lambda \text{sign}(\widehat{\boldsymbol{\beta}}_{\mathcal{S}^c}(\lambda)) + \Delta \mathbf{X}_{\mathcal{S}^c}^{\top}\mathbf{X}_{\mathcal{S}}(\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}})^{-1}\text{sign}(\widehat{\boldsymbol{\beta}}_{\mathcal{S}}(\lambda)). \tag{4.24}$$

That is, if one decrease λ to $\lambda - \Delta$, one must strictly follow Equation 4.24.

Following the above key step in the path following Lasso-algorithm, we represent a counter example as follows. Suppose $\beta_1 > \beta_2 > \beta_3 > \beta_4 = \beta_5 = \ldots = \beta_p = 0$. The model matrix $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_p)$, where $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$ is the first two columns from a orthogonal matrix $(\mathbf{x}_1, \mathbf{x}_2, \widetilde{\mathbf{x}}_3, \ldots, \widetilde{\mathbf{x}}_p)$, and for $j \geq 3$, we have $\mathbf{x}_j = \alpha_j \mathbf{x}_1 + (1 - \alpha_j)\mathbf{x}_2 + \sqrt{1 - \alpha_j^2 - (1 - \alpha_j)^2}\widetilde{\mathbf{x}}_j$ with $\alpha_j \in (0, 1)$. The response vector $\mathbf{y}$ is generated by

$$\mathbf{y} = \sum_{j=1}^{p} \beta_j \mathbf{x}_j.$$

If β_1, β_2 are very large number, say, 200, 100, and β_3 is not that large, say, 1. Then the following algorithm works as follows:

- Iteration 0: We start with $\lambda_0 = +\infty$, then we know that $\widehat{\boldsymbol{\beta}}(\lambda_0) = 0$ and $\mathcal{S}_0 = \varnothing$.
- Iteration 1: When λ changes from $\lambda_0 = +\infty$ to $\lambda_1 = \|\mathbf{q}\|_{\infty}$, from Equation 4.21, we know that $\mathcal{S}_1 = \{1\}$.

- Iteration 2: Similar to the first iteration, when λ decrease to λ_2, we have $\mathcal{S}_2 = \{1, 2\}$.

- Iteration 3: This is where problem happens. From Equation 4.24, we know that $\forall \lambda_2 - \Delta \in (\lambda_3, \lambda_2]$, we have

$$\text{sign}(\widehat{\boldsymbol{\beta}}_{\mathcal{S}_2^c}(\lambda - \Delta)) = \mathbf{X}_{\mathcal{S}_2^c}^\top \mathbf{X}_{\mathcal{S}_2} (\mathbf{X}_{\mathcal{S}_2}^\top \mathbf{X}_{\mathcal{S}_2})^{-1} \text{sign}(\widehat{\boldsymbol{\beta}}_{\mathcal{S}_2}(\lambda)).$$

  Since $\text{sign}(\widehat{\boldsymbol{\beta}}_{\mathcal{S}_2}(\lambda)) = (1, 1)^\top$ and $\mathbf{X}_{\mathcal{S}_2} = (\mathbf{x}_1, \mathbf{x}_2), \mathbf{X}_{\mathcal{S}_2^c} = (\mathbf{x}_3, \ldots, \mathbf{x}_p)$, we have the right hand side of the above equation as a all-one vector, i.e, $(1, 1, \ldots, 1)^\top$. To make the left hand side $\text{sign}(\widehat{\boldsymbol{\beta}}_{\mathcal{S}_2^c}(\lambda_2 - \Delta))$ equals to $(1, 1, \ldots, 1)^\top$, we can only take $\Delta = \lambda_2$, which gives us $\mathcal{S}_3 = \{1, 2, 3, \ldots, p\}$.

However, from the data generalization, we know that the true support set is $\{1, 2, 3\}$. Therefore, one will not be able to develop a path following algorithm to realize correct support set recovery. At least not in the sense of inserting one at a time to the support set. In the above example, since a path following approach can only visit three possible subsets, it won't solve the Lasso problem in general.

Details to Support the Second Drawback of Path Following Lasso-Algorithm

In this section, we provide more technitical details to support the second drawback of path following Lasso-Algorithm. Recall the main idea of path following Lasso-Algorithm is that, it begins with a large λ_0, which makes the estimated $\widehat{\boldsymbol{\beta}}(\lambda_0) = 0$, and accordingly its support set $\mathcal{S}_0 = \varnothing$ (empty set). Then it tries to identify a sequence of the penalty parameter λ as follows:

$$\lambda_0 > \lambda_1 > \lambda_2 > \ldots > \lambda_{T-1} > \lambda_T = 0,$$

such that for any $k \geq 1$, when we have $\lambda \in [\lambda_k, \lambda_{k-1}]$, the support of $\widehat{\boldsymbol{\beta}}(\lambda)$ (which is a function of λ) i.e., $\mathcal{S}_k$, remains unchanged. Moreover, within the interval $[\lambda_k, \lambda_{k-1}]$, vector

$\widehat{\boldsymbol{\beta}}(\lambda)$ elementwisely is a linear function of λ. However, when one is over the kink point, the support is changed/enlarged, i.e., we have $\mathcal{S}_k \neq \mathcal{S}_{k-1}$ or even $\mathcal{S}_k \subseteq \mathcal{S}_{k-1}$.

A point deserves attention is that, if T, the total number of kink points is small, then the path following algorithm is efficient, i.e., it only requires $O(nTp^2)$ numerical operations. In particular, if the size of supports are strictly increasing, i.e., we have

$$|\mathcal{S}_{k-1}| < |\mathcal{S}_k| \ \ \forall k \geq 1,$$

then we have $T \leq p$, and accordingly the computational complexity can be bounded by $O(np^3)$. However, it turns out bounding the value of T is an open question. In recent papers such as [122, 123], we can see that bounding T is an open problem.

4.6.3 An Important Theorem

Our proof will rely on a result on the number of steps in achieving certain accuracy in using the accelerate gradient descent (AGD) when the objective function is strongly convex. The result is the Theorem 3.7 in [135]. We represent the theorem here for readers' convenience. We introduce some notations first. Suppose ones wants to minimize a convex function $f : \mathbf{X} \to \mathbb{R}$ in a feasible closed convex set $\mathbf{X} \in \mathbb{R}^p$. We further assume that f is a differentiable convex function with Lipschitz continuous gradients L, i.e., $\forall \mathbf{x}, \mathbf{y} \in \mathbf{X}$, we have

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_2 \leq L \|\mathbf{x} - \mathbf{y}\|_2 ,$$

where $\nabla f(\mathbf{x})$ represents the gradient of function $f(\mathbf{x})$. Furthermore, we assume that f is a strongly convex function, i.e., $\forall \mathbf{x}, \mathbf{y} \in \mathbf{X}$, there exist $\mu > 0$, such that we have

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})(\mathbf{y} - \mathbf{x}) + \frac{\mu}{2} \|\mathbf{y} - \mathbf{x}\|_2^2 .$$

This type of function f is called the L-smooth and μ-strongly convex function. Recall that our objective is to solve the following problem:

$$\min_{\mathbf{x}\in\mathbf{X}} f(\mathbf{x}).$$

In the following, we present one version of the accelerated gradient descent (AGD) algorithm. Given $(\mathbf{x}^{(t-1)}, \bar{\mathbf{x}}^{(t-1)}) \in \mathbf{X} \times \mathbf{X}$ for $t = 1, 2, \ldots$, we set

$$\underline{\mathbf{x}}^{(t)} = (1 - q_t)\bar{\mathbf{x}}^{(t-1)} + q_t\mathbf{x}^{(t-1)} \tag{4.25}$$

$$\mathbf{x}^{(t)} = \arg\min_{\mathbf{x}\in\mathbf{X}} \left\{\gamma_t \left[\mathbf{x}^\top \nabla f\left(\underline{\mathbf{x}}^{(t)}\right) + \mu V\left(\underline{\mathbf{x}}^{(t)}, \mathbf{x}\right)\right] + V\left(\mathbf{x}^{(t-1)}, \mathbf{x}\right)\right\} \tag{4.26}$$

$$\bar{\mathbf{x}}^{(t)} = (1 - \alpha_t)\,\bar{\mathbf{x}}^{(t-1)} + \alpha_t\mathbf{x}^{(t)}, \tag{4.27}$$

for some $q_t \in [0, 1], \gamma_t \geq 0$, and $\alpha_t \in [0, 1]$. And here $V(\mathbf{x}, \mathbf{z})$ is the prox-function (or Bregman's distance), i.e.,

$$V(\mathbf{x}, \mathbf{z}) = v(\mathbf{z}) - \left[v(\mathbf{x}) + (\mathbf{z} - \mathbf{x})^\top \nabla v(\mathbf{x})\right],$$

with $v(\mathbf{x}) = \|\mathbf{x}\|_2^2/2$. By applying AGD as shown in Equation 4.25 - Equation 4.27, the following theorem presents an inequality that can be utilized to determine the number of iterations when certain precision of a solution is given.

Theorem 4.6.5. Let $\left(\underline{\mathbf{x}}^{(t)}, \mathbf{x}^{(t)}, \bar{\mathbf{x}}^{(t)}\right) \in \mathbf{X} \times \mathbf{X} \times \mathbf{X}$ be generated by accelerated gradient descent method in Equation 4.25-Equation 4.27. If $\alpha_t = \alpha, \gamma_t = \gamma$ and $q_t = q$, for $t = 1, \ldots, k$, satisfy $\alpha \geq q$, $\frac{L(\alpha-q)}{1-q} \leq \mu$, $\frac{Lq(1-\alpha)}{1-q} \leq \frac{1}{\gamma}$, and $\frac{1}{\gamma(1-\alpha)} \leq \mu + \frac{1}{\gamma}$, then for any $\mathbf{x} \in \mathbf{X}$, we have

$$\begin{aligned} & f\left(\bar{\mathbf{x}}^{(k)}\right) - f(\mathbf{x}) + \alpha\left(\mu + \frac{1}{\gamma}\right) V\left(\mathbf{x}^{(k-1)}, \mathbf{x}\right) \\ \leq\ & (1 - \alpha)^k \left[f\left(\bar{\mathbf{x}}^{(0)}\right) - f(\mathbf{x}) + \alpha\left(\mu + \frac{1}{\gamma}\right) V\left(\mathbf{x}^{(1)}, \mathbf{x}\right), \right]. \end{aligned}$$

In particular, if

$$\alpha = \sqrt{\frac{\mu}{L}}, q = \frac{\alpha - \mu/L}{1 - \mu/L}, \gamma = \frac{\alpha}{\mu(1-\alpha)},$$

then for any $\mathbf{x} \in \mathbf{X}$, we have

$$\begin{aligned} & f\left(\bar{\mathbf{x}}^{(k)}\right) - f(\mathbf{x}) + \alpha\left(\mu + \frac{1}{\gamma}\right) V\left(\mathbf{x}^{(k-1)}, \mathbf{x}\right). \\ \leq & \left(1 - \sqrt{\frac{\mu}{L}}\right)^k \left[f\left(\bar{\mathbf{x}}^{(0)}\right) - f(\mathbf{x}) + \alpha\left(\mu + \frac{1}{\gamma}\right) V\left(\mathbf{x}^{(1)}, \mathbf{x}\right)\right]. \end{aligned} \tag{4.28}$$

The above theorem gives a convergence rate of AGD under the scenario when the objective function is strongly convex. This result will be utilized in the proof of Theorem 4.3.1, which can be found in subsubsection 4.6.4.

4.6.4 Proofs

Proof of Lemma 4.2.1

Proof. In this proof, we will do two parts.

First, we will prove the existence of the initial point t_0 stated in Equation 4.4. We know that

$$\begin{aligned} & \lim_{t\to+\infty} \frac{\sum_{j=1}^{p} \mathbf{M}(t)_{ij} (\mathbf{X}^\top \mathbf{y}/n)_j}{t} \\ = & \lim_{t\to+\infty} \sum_{j=1}^{p} \left(\left[\frac{\mathbf{X}^\top \mathbf{X}}{n} + \frac{\lambda \left[\log(1+t)\right]^2}{3t^3} \mathbf{I} \right]^{-1} \right)_{ij} \left(\frac{\mathbf{X}^\top \mathbf{y}}{n} \right)_j \frac{1}{t} \\ = & \lim_{t\to+\infty} \sum_{j=1}^{p} \left(\left[\frac{\mathbf{X}^\top \mathbf{X} t}{n} + \frac{\lambda \left[\log(1+t)\right]^2}{3t^2} \mathbf{I} \right]^{-1} \right)_{ij} \left(\frac{\mathbf{X}^\top \mathbf{y}}{n} \right)_j. \\ = & 0. \end{aligned}$$

The above indicates that when t is very large, the t_0 defined in Equation 4.4 will exist.

Next, we will verify that, if t_0 is chosen as shown in Equation 4.4, i.e,

$$t_0 \in \left\{ t : \left| \sum_{j=1}^{p} \mathbf{M}(t)_{ij} (\mathbf{X}^\top \mathbf{y}/n)_j \right| \leq t, \forall i = 1, \ldots, p \right\},$$

we have $\left|\beta_i^{(0)}\right| < t_0$. It can be verified that,

$$\mathbf{M}(t)\frac{\mathbf{X}^\top \mathbf{y}}{n} = \arg\min_{\boldsymbol{\beta}} \left\{ \underbrace{\frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \frac{1}{3t^3}\left[\log(1+t)\right]^2 \boldsymbol{\beta}^\top \boldsymbol{\beta}}_{\mathcal{G}(\boldsymbol{\beta})} \right\}, \tag{4.29}$$

where $\mathcal{G}(\boldsymbol{\beta})$ is a special case of $F_t(\boldsymbol{\beta})$ when t is large enough to include all the coefficient β_i into the interval $[-t, t]$. Utilizing the above fact that the minimizer in Equation 4.29 when $t = t_0$ satisfies the condition that its coordinates are within $[-t_0, t_0]$, we have

$$|\beta_i(t_0)| = \left| \left(\mathbf{M}(t_0)\frac{\mathbf{X}^\top \mathbf{y}}{n} \right)_i \right| = \left| \sum_{j=1}^{p} \mathbf{M}(t_0)_{ij} (\mathbf{X}^\top \mathbf{y}/n)_j \right| \leq t_0.$$

Thus, if we choose t_0 as shown in Equation 4.4, i.e.,

$$t_0 \in \left\{ t : \left| \sum_{j=1}^{p} \mathbf{M}(t)_{ij} (\mathbf{X}^\top \mathbf{y}/n)_j \right| \leq t, \forall i = 1, \ldots, p \right\},$$

we can verify that $\forall i = 1, 2, \ldots p$, $|\beta_i(t_0)| \leq t_0$, i.e., $\left|\beta_i^{(0)}\right| \leq t_0$. □

Proof of Lemma 4.2.2

Proof. Because $f_t(x)$ is a even function, we only consider the positive x in the remaining of the proof.

First, when $0 \leq x \leq t$, one has

$$f_t(x) - x = \frac{1}{3t^3}\left[\log(1+t)\right]^2 x^2 - x,$$

which is a quadratic function with the axis of symmetry, $\frac{3t^3}{2[\log(1+t)]^2}$ being larger than t. Therefore, one has

$$\frac{1}{3t}\left[\log(1+t)\right]^2 - t \quad \leq f_t(x) - x \leq \quad 0.$$

Then we discuss the scenario when $x > t$, where

$$f_t(x) - x = \left[\left[\frac{\log(1+t)}{t}\right]^2 - 1\right] x + \frac{1}{3x}\left[\log(1+t)\right]^2 - \frac{1}{t}\left[\log(1+t)\right]^2,$$

which is a decreasing function of variable x. Therefore,

$$f_t(B) - B = \left[f_t(x) - x\right]|_{x=B} \leq f_t(x) - x \leq \left[f_t(x) - x\right]|_{x=t} = f_t(t) - t,$$

where we further have $f_t(t) - t = \frac{1}{3t}\left[\log(1+t)\right]^2 - t \leq 0$.

By the combination of two scenario ($x \leq t$ and $x > t$), we prove the statement in Equation 4.8. □

Proof of Theorem 4.3.1

Proof. To begin with, we revisit some notations in linear algebra. For matrix $\mathbf{A}$, we use A_{ij} to indicate the (i,j)th entry in matrix $\mathbf{A}$. Besides, its maximal/minimal eigenvalue is $\lambda_{\max}(\mathbf{A})/\lambda_{\min}(\mathbf{A})$, respectively.

It is known that, the condition number of function $F_{t_k}(\boldsymbol{\beta}) = \frac{1}{2n}\left\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right\|_2^2 + \lambda f_{t_k}(\boldsymbol{\beta})$ is defined by the ratio between the maximal and minimal eigenvalue of its Hessian. Recall that, the (i,j)th entry of the Hessian matrix of the surrogate function $f_{t_k}(\boldsymbol{\beta})$, noted as $H_{t_k,i,j}$, is

$$H_{t_k,i,j} \quad = \quad \begin{cases} \frac{2}{3}\left[\log(1+t_k)\right]^2 \max\{|\beta_i|, t_k\}^{-3}, & \text{if } i = j, \\ 0, & \text{otherwise.} \end{cases}$$

Note that the Hessian matrix $\mathbf{H}_{t_k}$ is diagonal and positive definite; therefore one can easily find its minimum and maximum eigenvalues. So the condition number of function $F_{t_k}(\boldsymbol{\beta}) = \frac{1}{2n} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda f_{t_k}(\boldsymbol{\beta})$, noted as κ_k, is

$$\begin{aligned}
\kappa_k &= \frac{\lambda_{\max}\left(\frac{\mathbf{X}^\top\mathbf{X}}{n} + \lambda\mathbf{H}_{t_k}\right)}{\lambda_{\min}\left(\frac{\mathbf{X}^\top\mathbf{X}}{n} + \lambda\mathbf{H}_{t_k}\right)} && (4.30)\\
&\leq \frac{\lambda_{\max}\left(\frac{\mathbf{X}^\top\mathbf{X}}{n}\right) + \lambda\lambda_{\max}\left(\mathbf{H}_{t_k}\right)}{\lambda_{\min}\left(\frac{\mathbf{X}^\top\mathbf{X}}{n}\right) + \lambda\lambda_{\min}\left(\mathbf{H}_{t_k}\right)} && (4.31)\\
&\leq \frac{\lambda_{\max}\left(\frac{\mathbf{X}^\top\mathbf{X}}{n}\right) + \lambda\lambda_{\max}\left(\mathbf{H}_{t_k}\right)}{\lambda\lambda_{\min}\left(\mathbf{H}_{t_k}\right)}\\
&= \frac{\lambda_{\max}\left(\frac{\mathbf{X}^\top\mathbf{X}}{n}\right) + \frac{2\lambda}{3t_k^3}\left[\log(1+t_k)\right]^2}{\frac{2\lambda}{3x^3}\left[\log(1+t_k)\right]^2} && (4.32)\\
&= \frac{3x^3\lambda_{\max}\left(\frac{\mathbf{X}^\top\mathbf{X}}{n}\right)}{2\lambda\left[\log(1+t_k)\right]^2} + \frac{x^3}{t_k^3}\\
&\leq \frac{3B^3\lambda_{\max}\left(\frac{\mathbf{X}^\top\mathbf{X}}{n}\right)}{2\lambda\left[\log(1+\tau)\right]^2} + \left(\frac{B}{\tau}\right)^3. && (4.33)
\end{aligned}$$

Equation 4.30 is due to the definition of the condition number. Equation 4.31 is because of the two fact. First, for the maximal eigenvalue of summation of two matrix $\mathbf{A} + \mathbf{B}$, i.e., $\lambda_{\max}(\mathbf{A} + \mathbf{B})$, is no more than summation of maximal eigenvalue separately, $\lambda_{\max}(\mathbf{A}) + \lambda_{\max}(\mathbf{B})$. Second, similar to the maximal eigenvalue, the minimal eigenvalue follows the similar rule that $\lambda_{\min}(\mathbf{A} + \mathbf{B}) \geq \lambda_{\min}(\mathbf{A}) + \lambda_{\min}(\mathbf{B})$. The equality in Equation 4.32 is due to the fact that matrix $\mathbf{H}_{t_k}$ is diagonal with positive diagonal entries. The x in Equation 4.32 refers to

$$x = \max\left\{|\beta_i| : \beta_i \text{ is the } i\text{th entry in } \boldsymbol{\beta}\right\}.$$

Equation 4.33 is because that $t_k \geq \tau$ and we assume that throughout the algorithm, all elements in $\boldsymbol{\beta}^{(k)}(k = 1, 2\ldots)$ is bounded by B.

By calling Theorem 3.7 in [135] (i.e., the theorem in subsection 4.6.3 in this paper), we

can prove the statement in Theorem 4.3.1. The details of the proof are listed as follows. Recall that we want to minimize $F_{t_k}(\boldsymbol{\beta})$ for a fixed k. From the previous analysis, we can find that $F_{t_k}(\boldsymbol{\beta})$ is L_k-smooth and μ_k-strongly convex, where $L_k = \lambda_{\max}\left(\frac{\mathbf{X}^\top\mathbf{X}}{n} + \lambda\mathbf{H}_{t_k}\right)$ and $\mu_k = \lambda_{\min}\left(\frac{\mathbf{X}^\top\mathbf{X}}{n} + \lambda\mathbf{H}_{t_k}\right)$. Consequently, the condition number in the kth outer-iteration $\kappa_k = \frac{L_k}{\mu_k}$ can be upper bounded by $\frac{3B^3\lambda_{\max}\left(\frac{\mathbf{X}^\top\mathbf{X}}{n}\right)}{2\lambda[\log(1+\tau)]^2} + \left(\frac{B}{\tau}\right)^3$ for any $\{k = 0, 1, 2, \ldots : t_k \geq \tau\}$.

For a fixed k, when applying AGD to minimize $F_{t_k}(\boldsymbol{\beta})$, our steps, which are line 8 - line 10 in algorithm 10, follows the AGD steps that are presented in Equation 4.25-Equation 4.27, by setting $\alpha_k = \sqrt{\frac{\mu_k}{L_k}}, q_k = \frac{\alpha_k - \mu_k/L_k}{1-\mu_k/L_k}, \gamma_k = \frac{\alpha_k}{\mu_k(1-\alpha_k)}$. According to Equation 4.28 in Theorem 4.6.5, if

$$(1-\alpha_k)^s \underbrace{\left[F_{t_k}(\bar{\boldsymbol{\beta}}^{(k)[0]}) - F_{k,\min} + \alpha_k\left(\mu_k + \frac{1}{\gamma_k}\right)V(\boldsymbol{\beta}^{(k-1)[1]}, \widehat{\boldsymbol{\beta}}_k)\right]}_{\mathcal{C}_k} - \underbrace{\alpha_k\left(\mu_k + \frac{1}{\gamma_k}\right)V(\boldsymbol{\beta}^{(k-1)[s-1]}, \widehat{\boldsymbol{\beta}}_k)}_{\mathcal{D}_k} \leq \widetilde{\epsilon}_k, \quad (4.34)$$

then $F_{t_k}(\boldsymbol{\beta}^{(k)[s]}) - F_{k,\min} \leq \widetilde{\epsilon}_k$ with a given $\widetilde{\epsilon}_k$. Here in Equation 4.34, we have $\widehat{\boldsymbol{\beta}}_k = \arg\min_{\boldsymbol{\beta}} F_{t_k}(\boldsymbol{\beta})$ and function $V(\cdot,\cdot)$ has been defined in subsection 4.6.3.

We then solve the inequality in Equation 4.34 to get an explicit formula for the quantity s. To achieve this goal, we simplify Equation 4.34 first. Note that quantities $\mathcal{C}_k$ and $\mathcal{D}_k$ are defined via underlining in Equation 4.34. It can be verified that $\mathcal{D}_k \geq 0$. This is because $v(\mathbf{x}) = \|\mathbf{x}\|_2^2/2$ (recall the definition of $v(\mathbf{x})$ in subsection 4.6.3) is a convex function, i.e., we have

$$V(\boldsymbol{\beta}^{(k-1)[s-1]}, \widehat{\boldsymbol{\beta}}_k) = v(\widehat{\boldsymbol{\beta}}_k) - \left[v(\boldsymbol{\beta}^{(k-1)[s-1]}) + (\widehat{\boldsymbol{\beta}}_k - \boldsymbol{\beta}^{(k-1)[s-1]})^\top \nabla v(\boldsymbol{\beta}^{(k-1)[s-1]})\right] \geq 0.$$

Since $\mathcal{D}_k > 0$, if we have

$$(1-\alpha_k)^s \mathcal{C}_k \leq \widetilde{\epsilon}_k,$$

then the inequality in Equation 4.34 will be satisfied. By introducing simple linear algebra, the above inequality can be rewritten as

$$(1-\alpha_k)^s \leq \frac{\widetilde{\epsilon}_k}{\mathcal{C}_k}.$$

By taking logarithm of both sides, we have

$$s\log(1-\alpha_k) \leq \log\left(\frac{\widetilde{\epsilon}_k}{\mathcal{C}_k}\right),$$

which gives

$$s \geq \frac{-\log\left(\frac{\widetilde{\epsilon}_k}{\mathcal{C}_k}\right)}{-\log(1-\alpha_k)} = \frac{\log\left(\frac{\mathcal{C}_k}{\widetilde{\epsilon}_k}\right)}{-\log(1-\alpha_k)}. \tag{4.35}$$

Furthermore, we know $\log\left(\frac{1}{1-x}\right) \geq x$ for $0 < x < 1$, so if

$$s \geq \frac{\log\left(\frac{\mathcal{C}_k}{\widetilde{\epsilon}_k}\right)}{\alpha_k}, \tag{4.36}$$

then the inequality in Equation 4.35 holds. In summary, if we have Equation 4.36, then we have $F_{t_k}(\boldsymbol{\beta}^{(k)[s]}) - F_{t_k}(\widehat{\boldsymbol{\beta}}_k) < \widetilde{\epsilon}_k$.

Now we will show that, both $\frac{1}{\alpha_k} = \sqrt{\frac{L_k}{\mu_k}}$ and $\log(\mathcal{C}_k)$ in Equation 4.36 can be bounded by a constant that does not depend on k (or equivalently, t_k). First, we prove that $\frac{1}{\alpha_k} = \sqrt{\frac{L_k}{\mu_k}}$ can be bound. This is essentially the argument that have been used in the step Equation 4.33. Second, we prove that $\mathcal{C}_k$ is also bounded. Because we have

$$\mathcal{C}_k = \underbrace{F_{t_k}(\boldsymbol{\beta}^{(k)[0]}) - F_{k,\min}}_{\mathcal{C}_{k,1}} + \underbrace{\alpha_k\left(\mu_k + \frac{1}{\gamma_k}\right)}_{\mathcal{C}_{k,2}} \underbrace{V(\boldsymbol{\beta}^{(k-1)[1]}, \widehat{\boldsymbol{\beta}}_k)}_{\mathcal{C}_{k,3}}.$$

Note that quantities $\mathcal{C}_{k,1}$, $\mathcal{C}_{k,2}$, and $\mathcal{C}_{k,3}$ are defined via underlining in the above equation. It

is evident that $\mathcal{C}_{k,1}$ and $\mathcal{C}_{k,3}$ are bounded. For $\mathcal{C}_{k,2}$, we have

$$C_{k,2} = \mu_k,$$

because we set $\gamma_k = \frac{\alpha_k}{\mu_k(1-\alpha_k)}$. Since μ_k is bounded above by a constant, quantity $\mathcal{C}_{k,2}$ is bounded as well. By combining the above several block, we know $\log(\mathcal{C}_k)$ is bounded.

In conclusion, after $C_1 \log(1/\widetilde{\epsilon}_k)$ inner-iterations, one is guaranteed to achieve the following precision

$$F_{t_k}(\boldsymbol{\beta}^{(k)}) - F_{\min,k} \leq \widetilde{\epsilon}_k,$$

where $\widetilde{\epsilon}_k = \frac{\lambda p}{3B} \left[\log(1+t_k)\right]^2$ and C_1 is a constant that does not depend on the value of t_k (or k).

□

Proof of Theorem 4.3.2

Proof. We start by showing that, for any $t \geq 0$, one has

$$F(\boldsymbol{\beta}^{(k)}) - F(\widehat{\boldsymbol{\beta}}) \leq \lambda p(2B+1)t_k.$$

This is because of the following sequence of inequalities for any $\boldsymbol{\beta} \in \mathbb{R}^p$:

$$
\begin{aligned}
F(\boldsymbol{\beta}^{(k)}) &= \frac{1}{2n}\left\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}^{(k)}\right\|_2^2+\lambda\left\|\boldsymbol{\beta}^{(k)}\right\|_1 \\
&= \frac{1}{2n}\left\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}^{(k)}\right\|_2^2+\lambda\sum_{i=1}^{p}\left|\beta_i^{(k)}\right| \\
&\leq \frac{1}{2n}\left\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}^{(k)}\right\|_2^2+\lambda\sum_{i=1}^{p} f_{t_k}(\beta_i^{(k)})-\lambda p\left[f_{t_k}(x)-x\right]|_{x=B} \qquad (4.37) \\
&= \frac{1}{2n}\left\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}^{(k)}\right\|_2^2+\lambda\sum_{i=1}^{p} f_{t_k}(\beta_i^{(k)})+ \\
&\quad \lambda pB\left[1-\left[\frac{\log(1+t_k)}{t_k}\right]^2\right]-\frac{\lambda p}{3B}\left[\log(1+t_k)\right]^2+\frac{\lambda p}{t_k}\left[\log(1+t_k)\right]^2 \qquad (4.38) \\
&\leq \frac{1}{2n}\left\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}^{(k)}\right\|_2^2+\lambda\sum_{i=1}^{p} f_{t_k}(\beta_i^{(k)})+ \\
&\quad \lambda pB\left[1-\left(\frac{1}{1+t_k}\right)^2\right]-\frac{\lambda p}{3B}\left[\log(1+t_k)\right]^2+\lambda p t_k \qquad (4.39) \\
&\leq \frac{1}{2n}\left\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}^{(k)}\right\|_2^2+\lambda\sum_{i=1}^{p} f_{t_k}(\beta_i^{(k)})+2\lambda pBt_k-\frac{\lambda p}{3B}\left[\log(1+t_k)\right]^2+ \\
&\quad \lambda p t_k \qquad (4.40) \\
&= \frac{1}{2n}\left\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}^{(k)}\right\|_2^2+\lambda\sum_{i=1}^{p} f_{t_k}(\beta_i^{(k)})+\lambda p(2B+1)t_k-\frac{\lambda p}{3B}\left[\log(1+t_k)\right]^2 \\
&\leq \frac{1}{2n}\left\|\mathbf{y}-\mathbf{X}\widehat{\boldsymbol{\beta}}^{(k)}\right\|_2^2+\lambda\sum_{i=1}^{p} f_{t_k}(\widehat{\beta}_i^{(k)})+\lambda p(2B+1)t_k- \\
&\quad \frac{\lambda p}{3B}\left[\log(1+t_k)\right]^2+\widetilde{\epsilon}_k \qquad (4.41) \\
&= \frac{1}{2n}\left\|\mathbf{y}-\mathbf{X}\widehat{\boldsymbol{\beta}}^{(k)}\right\|_2^2+\lambda\sum_{i=1}^{p} f_{t_k}(\widehat{\beta}_i^{(k)})+\lambda p(2B+1)t_k \qquad (4.42) \\
&\leq \frac{1}{2n}\left\|\mathbf{y}-\mathbf{X}\widehat{\boldsymbol{\beta}}\right\|_2^2+\lambda\sum_{i=1}^{p} f_{t_k}(\widehat{\beta}_i)+\lambda p(2B+1)t_k \qquad (4.43) \\
&\leq \frac{1}{2n}\left\|\mathbf{y}-\mathbf{X}\widehat{\boldsymbol{\beta}}\right\|_2^2+\lambda\left\|\widehat{\boldsymbol{\beta}}\right\|_1+\lambda p(2B+1)t_k \qquad (4.44) \\
&= F(\widehat{\boldsymbol{\beta}})+\lambda p(2B+1)t_k
\end{aligned}
$$

where Equation 4.37 is due to the left side hand of Equation 4.8, i.e., $[f_{t_k}(x) - |x|]\,|_{x=B} \leq f_t(x) - |x|$. And Equation 4.37 is by plugging in the value of $[f_{t_k}(x) - x]\,|_{x=t_0}$. Equation 4.39 utilizes the inequality that $\frac{t_k}{1+t_k} \leq \log(1+t_k)$ and inequality $\log(1+t_k) \leq t_k$. Equation 4.40 uses inequality $1 - \frac{1}{(1+t_k)^2} \leq 2t_k$. Equation 4.41 is because that we assume the precision in kth inner-iteration is $F_{t_k}(\boldsymbol{\beta}^{(k)}) - F_{t_k}(\widehat{\boldsymbol{\beta}}^{(k)}) \leq \widetilde{\epsilon}_k$. Equation Equation 4.42 is owing to the fact that we set $\widetilde{\epsilon}_k = \frac{\lambda p}{3B}\left[\log(1+t_k)\right]^2$. Equation 4.44 is due to the right hand side of Equation 4.8, i.e., $f_t(x) - |x| \leq 0$. Equation 4.43 is because $\widehat{\boldsymbol{\beta}}^{(k)}$ is the minimizer of $F_{t_k}(\boldsymbol{\beta})$, so $F_{t_k}(\widehat{\boldsymbol{\beta}}^{(k)}) < F_{t_k}(\widehat{\boldsymbol{\beta}})$. Equation 4.44 is because $f_{t_k}(x) - |x| \leq 0$ in Lemma 4.2.2.

Through the above series of equalities and inequalities, we know that

$$F(\boldsymbol{\beta}^{(k)}) - F(\widehat{\boldsymbol{\beta}}) \leq \lambda p(2B+1)t_k. \tag{4.45}$$

Besides, in the statement of the theorem, we have

$$k \geq \frac{-1}{\log(1-h)} \log\left(\frac{\lambda p(2B+1)t_0}{\epsilon}\right),$$

which is equivalent to

$$\lambda p(2B+1)t_k \leq \epsilon.$$

So the right side of Equation 4.45 isn't larger than ϵ. Thus, we prove that, when $k \geq \frac{-1}{\log(1-h)} \log\left(\frac{\lambda p(2B+1)t_0}{\epsilon}\right)$, we have $F(\boldsymbol{\beta}^{(k)}) - F(\widehat{\boldsymbol{\beta}}) \leq \epsilon$.

□

Proof of Theorem 4.3.3

Proof. The total number of numeric operations is determined by three factors, namely (1) the number of out-iterations, (2) the number of inner-iterations, and (3) the number of numeric operations in each inner-iterations. We adopt the assumption that different

basic operations can be treated equally. We have discussed (1) and (2) in subsection 4.3.1 and subsection 4.3.2, and we discuss (3) briefly here. The main computational cost of an inner-iteration in our proposed algorithm lies in line 9 of algorithm 10, which is the matrix multiplication in $\frac{\partial}{\partial \boldsymbol{\beta}^{(k)[s]}} F_{t_k}(\boldsymbol{\beta}^{(k)[s]}) = \frac{\mathbf{X}^\top \mathbf{X}}{n} \boldsymbol{\beta}^{(k)[s]} - \frac{\mathbf{X}^\top \mathbf{y}}{n} + \frac{\partial}{\partial \boldsymbol{\beta}^{(k)[s]}} f_{t_k}(\boldsymbol{\beta}^{(k)[s]})$. With matrix $\frac{\mathbf{X}^\top \mathbf{X}}{n}, \frac{\mathbf{X}^\top \mathbf{y}}{n}$ being pre-calculated and stored at the beginning of the execution, the calculation of $\frac{\partial}{\partial \boldsymbol{\beta}^{(k)[i]}} F_{t_k}(\boldsymbol{\beta}^{(k)[s]})$ requires $O(p^2)$ operations.

Now we count the total number of numerical operations that are need in our proposed method to achieve the ϵ precision. We know that to achieve $F(\boldsymbol{\beta}^{(k)}) - F_{\min} < \epsilon$, we need at least (Theorem 4.3.2)

$$N \triangleq \frac{-1}{\log(1-h)} \log\left(\frac{\lambda p(2B+1)t_0}{\epsilon}\right)$$

outer-iterations. Furthermore, we know that the number inner-iteration in an inner-loop k is $O(\log(\frac{1}{\widetilde{\epsilon}_k}))$ with a hidden constant which can be universally bounded, and the number of operations in each inner-iteration is p^2. Therefore, the total number of numerical operations to get the estimator $\boldsymbol{\beta}^{(k)}$ with precision $F(\boldsymbol{\beta}^{(k)}) - F(\widehat{\boldsymbol{\beta}}) \leq \epsilon$ can be upper bounded by the

following quantity:

$$
\begin{aligned}
& p^2 \sum_{k=1}^{N} \log\left(\frac{1}{\widetilde{\epsilon}_k}\right) && (4.46)\\
= \;& p^2 \sum_{k=1}^{N} \log\left(\frac{3B}{\lambda p}\left[\left[\log(1+t_k)\right]^2\right]^{-1}\right) && (4.47)\\
= \;& p^2 \sum_{k=1}^{N} \log\left(\frac{3B}{\lambda p}\right) - p^2 \sum_{k=1}^{N} \log\left(\left[\log(1+t_k)\right]^2\right) \\
= \;& p^2 N \log\left(\frac{3B}{\lambda p}\right) - 2p^2 \sum_{k=1}^{N} \log\left(\log(1+t_k)\right) \\
\leq \;& p^2 N \log\left(\frac{3B}{\lambda p}\right) - 2p^2 \sum_{k=1}^{N} \log\left(\frac{t_k}{1+t_k}\right) && (4.48)\\
= \;& p^2 N \log\left(\frac{3B}{\lambda p}\right) - 2p^2 \sum_{k=1}^{N} \log\left(t_k\right) + 2p^2 \sum_{k=1}^{N} \log\left(1+t_k\right) \\
= \;& p^2 N \log\left(\frac{3B}{\lambda p}\right) - 2p^2 \sum_{k=1}^{N} \log\left(t_0(1-h)^k\right) + 2p^2 \sum_{k=1}^{N} \log\left(1+t_k\right) \\
\leq \;& p^2 N \log\left(\frac{3B}{\lambda p}\right) - 2p^2 \sum_{k=1}^{N} \log\left(t_0(1-h)^k\right) + 2p^2 \sum_{k=1}^{N} t_k && (4.49)\\
= \;& p^2 N \log\left(\frac{3B}{\lambda p}\right) - 2p^2 \sum_{k=1}^{N} \left[\log\left(t_0\right) + k\log\left(1-h\right)\right] + 2p^2 \sum_{k=1}^{N} t_0(1-h)^k \\
= \;& p^2 N \log\left(\frac{3B}{\lambda p}\right) - 2p^2 N \log\left(t_0\right) - 2p^2 \log\left(1-h\right) \sum_{k=1}^{N} k + 2p^2 \sum_{k=1}^{N} t_0(1-h)^k \\
= \;& p^2 N \log\left(\frac{3B}{\lambda p}\right) - 2p^2 N \log\left(t_0\right) - 2p^2 \log\left(1-h\right) \frac{(N+1)N}{2} \\
& + 2p^2 \frac{t_0\left[1-(1-h)^N\right]}{h} \\
= \;& O(N^2)
\end{aligned}
$$

where Equation 4.47 is derived by plugging in that $\widetilde{\epsilon}_k = \frac{\lambda p}{3B}\left[\log(1+t_k)\right]^2$. To be more exactly, there is a hidden constant related to the big O notation in $O\left(\log\left(\frac{1}{\widetilde{\epsilon}_k}\right)\right)$ in Equation 4.47, however, as mention in the proof of Theorem 4.3.1, this hidden constant can be bounded universally. So in Equation 4.47, we omit this hidden constant. Equation 4.48 is

derived due to the inequality that $\log(1+x) \geq \frac{x}{1+x}$ for $x \geq 0$. Equation 4.49 is derived due to the inequality that $\log(1+x) \leq x$ for $x \geq 0$. □

Proof of Proposition 4.5.1

Proof. Because $\widehat{\boldsymbol{\beta}}$ is the minimizer of $\frac{1}{2}\left\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right\|_2^2 + \lambda\left\|\boldsymbol{\beta}\right\|_1$, we can get its first-order condition as:

$$\frac{1}{n}\left(\mathbf{X}^\top\mathbf{X}\widehat{\boldsymbol{\beta}} + \mathbf{X}^\top\mathbf{y}\right) + \lambda \text{sign}\left(\widehat{\boldsymbol{\beta}}\right) = 0 \tag{4.50}$$

And because $\widetilde{\boldsymbol{\beta}}$ is the minimizer of $\frac{1}{2}\left\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right\|_2^2 + \lambda f_t\left(\boldsymbol{\beta}\right)$, we can get its first-order condition as:

$$\frac{1}{n}\left(\mathbf{X}^\top\mathbf{X}\widetilde{\boldsymbol{\beta}} + \mathbf{X}^\top\mathbf{y}\right) + \lambda\nabla f_t\left(\widetilde{\boldsymbol{\beta}}\right) = 0, \tag{4.51}$$

where $\nabla f_t\left(\widetilde{\boldsymbol{\beta}}\right)$ is the gradient of $f_t\left(\widetilde{\boldsymbol{\beta}}\right)$. By subtracting Equation 4.50 from Equation 4.51, we have

$$\frac{1}{n}\mathbf{X}^\top\mathbf{X}\left(\widetilde{\boldsymbol{\beta}}-\widehat{\boldsymbol{\beta}}\right) + \lambda\left[\nabla f_t\left(\boldsymbol{\beta}\right) - \text{sign}\left(\widehat{\boldsymbol{\beta}}\right)\right] = 0.$$

By left multiplying $\left(\widetilde{\boldsymbol{\beta}}-\widehat{\boldsymbol{\beta}}\right)^\top$ on both sides of the above equation, we have

$$\frac{1}{n}\left(\widetilde{\boldsymbol{\beta}}-\widehat{\boldsymbol{\beta}}\right)^\top\mathbf{X}^\top\mathbf{X}\left(\widetilde{\boldsymbol{\beta}}-\widehat{\boldsymbol{\beta}}\right) + \lambda\left(\widetilde{\boldsymbol{\beta}}-\widehat{\boldsymbol{\beta}}\right)^\top\left[\nabla f_t\left(\widetilde{\boldsymbol{\beta}}\right) - \text{sign}\left(\widehat{\boldsymbol{\beta}}\right)\right] = 0.$$

The above is equivalent to

$$\begin{aligned}
&\frac{1}{n}\left(\widetilde{\boldsymbol{\beta}}-\widehat{\boldsymbol{\beta}}\right)^\top\mathbf{X}^\top\mathbf{X}\left(\widetilde{\boldsymbol{\beta}}-\widehat{\boldsymbol{\beta}}\right)\\
=\;& -\lambda\left(\widetilde{\boldsymbol{\beta}}-\widehat{\boldsymbol{\beta}}\right)^\top\left[\nabla f_t\left(\widetilde{\boldsymbol{\beta}}\right) - \text{sign}\left(\widehat{\boldsymbol{\beta}}\right)\right]\\
=\;& -\lambda\left(\widetilde{\boldsymbol{\beta}}-\widehat{\boldsymbol{\beta}}\right)^\top\nabla f_t\left(\widetilde{\boldsymbol{\beta}}\right) + \lambda\left(\widetilde{\boldsymbol{\beta}}-\widehat{\boldsymbol{\beta}}\right)^\top\text{sign}\left(\widehat{\boldsymbol{\beta}}\right)\\
=\;& -\lambda\left(\widetilde{\boldsymbol{\beta}}-\widehat{\boldsymbol{\beta}}\right)^\top\nabla f_t\left(\widetilde{\boldsymbol{\beta}}\right) + \lambda\widetilde{\boldsymbol{\beta}}^\top\text{sign}\left(\widehat{\boldsymbol{\beta}}\right) - \lambda\widehat{\boldsymbol{\beta}}^\top\text{sign}\left(\widehat{\boldsymbol{\beta}}\right)\\
=\;& -\lambda\left(\widetilde{\boldsymbol{\beta}}-\widehat{\boldsymbol{\beta}}\right)^\top\nabla f_t\left(\widetilde{\boldsymbol{\beta}}\right) + \lambda\widetilde{\boldsymbol{\beta}}^\top\text{sign}\left(\widehat{\boldsymbol{\beta}}\right) - \lambda\left\|\widehat{\boldsymbol{\beta}}\right\|_1.
\end{aligned}$$

Because $f_t(\boldsymbol{\beta})$ is a convex function, we have

$$\frac{1}{n}\left(\widetilde{\boldsymbol{\beta}}-\widehat{\boldsymbol{\beta}}\right)^{\top}\mathbf{X}^{\top}\mathbf{X}\left(\widetilde{\boldsymbol{\beta}}-\widehat{\boldsymbol{\beta}}\right)\leq-\lambda\left[f_t\left(\widehat{\boldsymbol{\beta}}\right)-f_t\left(\widetilde{\boldsymbol{\beta}}\right)\right]+\lambda\widetilde{\boldsymbol{\beta}}^{\top}\text{sign}\left(\widehat{\boldsymbol{\beta}}\right)-\lambda\left\|\widehat{\boldsymbol{\beta}}\right\|_1.$$

So we have

$$\frac{1}{n}\left\|\mathbf{X}\left(\widetilde{\boldsymbol{\beta}}-\widehat{\boldsymbol{\beta}}\right)\right\|_2^2\leq-\lambda\left[f_t\left(\widehat{\boldsymbol{\beta}}\right)-f_t\left(\widetilde{\boldsymbol{\beta}}\right)\right]+\lambda\left\|\widetilde{\boldsymbol{\beta}}\right\|_1-\lambda\left\|\widehat{\boldsymbol{\beta}}\right\|_1.$$

When $t\to 0$, we have $f_t(\boldsymbol{\beta})$ very close to $\|\boldsymbol{\beta}\|_1$, so we have $\frac{1}{n}\left\|X\left(\widetilde{\boldsymbol{\beta}}-\widehat{\boldsymbol{\beta}}\right)\right\|_2^2\to 0$. □

Proof of Proposition 4.5.2

Proof. From Proposition 4.5.1, we know that

$$\left\|\mathbf{X}\left(\widetilde{\boldsymbol{\beta}}-\widehat{\boldsymbol{\beta}}\right)\right\|_2^2\to 0$$

when $t\to 0$, where $\widehat{\boldsymbol{\beta}}=\arg\min_{\boldsymbol{\beta}}\frac{1}{2n}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2+\lambda\|\boldsymbol{\beta}\|_1$, and $\widetilde{\boldsymbol{\beta}}=\arg\min_{\boldsymbol{\beta}}\frac{1}{2n}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2+\lambda f_t(\boldsymbol{\beta})$. The above can be written as

$$\mathbf{X}_{\mathcal{S}}\left(\widetilde{\boldsymbol{\beta}}_{\mathcal{S}}-\widehat{\boldsymbol{\beta}}_{\mathcal{S}}\right)+\mathbf{X}_{\mathcal{S}^c}\widetilde{\boldsymbol{\beta}}_{\mathcal{S}^c}=\boldsymbol{\delta},\tag{4.52}$$

where $\mathcal{S}$ is the support set of $\widehat{\boldsymbol{\beta}}$ and $\|\boldsymbol{\delta}\|_2^2\approx 0$. By left multiplying $\left(\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}}\right)^{-1}\mathbf{X}_{\mathcal{S}}^{\top}$ on both sides of Equation 4.52, we have

$$\left(\widetilde{\boldsymbol{\beta}}_{\mathcal{S}}-\widehat{\boldsymbol{\beta}}_{\mathcal{S}}\right)+\left(\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}}\right)^{-1}\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}^c}\widetilde{\boldsymbol{\beta}}_{\mathcal{S}^c}=\left(\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}}\right)^{-1}\mathbf{X}_{\mathcal{S}}^{\top}\boldsymbol{\delta},\tag{4.53}$$

By left multiplying $\mathbf{X}_{\mathcal{S}^c}^{\dagger}$ on both sides of Equation 4.52, we have

$$\mathbf{X}_{\mathcal{S}^c}^{\dagger}\mathbf{X}_{\mathcal{S}}\left(\widetilde{\boldsymbol{\beta}}_{\mathcal{S}}-\widehat{\boldsymbol{\beta}}_{\mathcal{S}}\right)+\mathbf{X}_{\mathcal{S}^c}^{\dagger}\mathbf{X}_{\mathcal{S}^c}\widetilde{\boldsymbol{\beta}}_{\mathcal{S}^c}=\mathbf{X}_{\mathcal{S}^c}^{\dagger}\boldsymbol{\delta},\tag{4.54}$$

where $\mathbf{X}_{\mathcal{S}^c}^{\dagger}$ is the *pseudo-inverse* of matrix $\mathbf{X}_{\mathcal{S}^c}$. The mathematical meaning of pseudo-inverse is that, suppose $\mathbf{X}_{\mathcal{S}^c} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}$, which is the singular value decomposition (SVD) of $\mathbf{X}_{\mathcal{S}^c}$. Then $\mathbf{X}_{\mathcal{S}^c}^{\dagger} = \mathbf{V}^{\top}\mathbf{\Sigma}^{\dagger}\mathbf{U}^{\top}$. For the rectangular diagonal matrix $\mathbf{\Sigma}$, we get $\mathbf{\Sigma}^{\dagger}$ by taking the reciprocal of each non-zero elements on the diagonal, leaving the zeros in place, and then transposing the matrix.

By reorganizing Equation 4.53 and Equation 4.54 into block matrix, we have

$$\underbrace{\begin{pmatrix} \mathbf{I} & \left(\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}}\right)^{-1}\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}^c} \\ \mathbf{X}_{\mathcal{S}^c}^{\dagger}\mathbf{X}_{\mathcal{S}} & \mathbf{X}_{\mathcal{S}^c}^{\dagger}\mathbf{X}_{\mathcal{S}^c} \end{pmatrix}}_{M} \begin{pmatrix} \widetilde{\boldsymbol{\beta}}_{\mathcal{S}} - \widehat{\boldsymbol{\beta}}_{\mathcal{S}} \\ \widetilde{\boldsymbol{\beta}}_{\mathcal{S}^c} \end{pmatrix} = \begin{pmatrix} \left(\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}}\right)^{-1}\mathbf{X}_{\mathcal{S}}^{\top}\boldsymbol{\delta} \\ \mathbf{X}_{\mathcal{S}^c}^{\dagger}\boldsymbol{\delta} \end{pmatrix}.$$

Through this system of equations, we can solve $\left\| \begin{pmatrix} \widetilde{\boldsymbol{\beta}}_{\mathcal{S}} - \widehat{\boldsymbol{\beta}}_{\mathcal{S}} \\ \widetilde{\boldsymbol{\beta}}_{\mathcal{S}^c} \end{pmatrix} \right\|_2^2$ as

$$\left\| \begin{pmatrix} \widetilde{\boldsymbol{\beta}}_{\mathcal{S}} - \widehat{\boldsymbol{\beta}}_{\mathcal{S}} \\ \widetilde{\boldsymbol{\beta}}_{\mathcal{S}^c} \end{pmatrix} \right\|_2^2 = \left\| \widetilde{\boldsymbol{\beta}}_{\mathcal{S}} - \widehat{\boldsymbol{\beta}}_{\mathcal{S}} \right\|_2^2 + \left\| \widetilde{\boldsymbol{\beta}}_{\mathcal{S}^c} \right\|_2^2 = \left\| \mathbf{M}^{-1} \begin{pmatrix} \left(\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}}\right)^{-1}\mathbf{X}_{\mathcal{S}}^{\top}\boldsymbol{\delta} \\ \mathbf{X}_{\mathcal{S}^c}^{\dagger}\boldsymbol{\delta} \end{pmatrix} \right\|_2^2.$$

Because for a matrix $\mathbf{A}$ and vector $\mathbf{x}$, we have $\|\mathbf{A}\mathbf{x}\|_2^2 \leq \|\mathbf{A}\|_F^2 \|\mathbf{x}\|_2^2$, we can bound $\left\| \widetilde{\boldsymbol{\beta}}_{\mathcal{S}} - \widehat{\boldsymbol{\beta}}_{\mathcal{S}} \right\|_2^2 + \left\| \widetilde{\boldsymbol{\beta}}_{\mathcal{S}^c} \right\|_2^2$ as

$$\begin{aligned} \left\| \widetilde{\boldsymbol{\beta}}_{\mathcal{S}} - \widehat{\boldsymbol{\beta}}_{\mathcal{S}} \right\|_2^2 + \left\| \widetilde{\boldsymbol{\beta}}_{\mathcal{S}^c} \right\|_2^2 &\leq \left\| \mathbf{M}^{-1} \right\|_F^2 \left\| \begin{pmatrix} \left(\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}}\right)^{-1}\mathbf{X}_{\mathcal{S}}^{\top}\boldsymbol{\delta} \\ \mathbf{X}_{\mathcal{S}^c}^{\dagger}\boldsymbol{\delta} \end{pmatrix} \right\|_2^2 \\ &= \left\| \mathbf{M}^{-1} \right\|_F^2 \left(\left\| \left(\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}}\right)^{-1}\mathbf{X}_{\mathcal{S}}^{\top}\boldsymbol{\delta} \right\|_2^2 + \left\| \mathbf{X}_{\mathcal{S}^c}^{\dagger}\boldsymbol{\delta} \right\|_2^2 \right). \end{aligned}$$

Because $\left\| \mathbf{M}^{-1} \right\|_F \leq \sqrt{\text{rank}(\mathbf{M}^{-1})} \left\| \mathbf{M}^{-1} \right\|_2$, we can further bound $\left\| \widetilde{\boldsymbol{\beta}}_{\mathcal{S}} - \widehat{\boldsymbol{\beta}}_{\mathcal{S}} \right\|_2^2 + \left\| \widetilde{\boldsymbol{\beta}}_{\mathcal{S}^c} \right\|_2^2$

as

$$\begin{aligned}
&\left\|\widetilde{\boldsymbol{\beta}}_{\mathcal{S}}-\widehat{\boldsymbol{\beta}}_{\mathcal{S}}\right\|_2^2+\left\|\widetilde{\boldsymbol{\beta}}_{\mathcal{S}^c}\right\|_2^2 \\
\leq\ & \operatorname{rank}(\mathbf{M}^{-1})\left\|\mathbf{M}^{-1}\right\|_2^2\left(\left\|\left(\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}}\right)^{-1}\mathbf{X}_{\mathcal{S}}^{\top}\boldsymbol{\delta}\right\|_2^2+\left\|\mathbf{X}_{\mathcal{S}^c}^{\dagger}\boldsymbol{\delta}\right\|_2^2\right) \\
=\ & \operatorname{rank}(\mathbf{M}^{-1})\left[\frac{1}{\lambda_{\min}(\mathbf{M})}\right]^2\left(\left\|\left(\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}}\right)^{-1}\mathbf{X}_{\mathcal{S}}^{\top}\boldsymbol{\delta}\right\|_2^2+\left\|\mathbf{X}_{\mathcal{S}^c}^{\dagger}\boldsymbol{\delta}\right\|_2^2\right). \qquad (4.55)
\end{aligned}$$

For $\left\|\left(\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}}\right)^{-1}\mathbf{X}_{\mathcal{S}}^{\top}\boldsymbol{\delta}\right\|_2^2$ in Equation 4.55, we have

$$\begin{aligned}
\left\|\underbrace{\left(\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}}\right)^{-1}\mathbf{X}_{\mathcal{S}}^{\top}}_{\mathbf{Q}}\boldsymbol{\delta}\right\|_2^2 &= \|\mathbf{Q}\boldsymbol{\delta}\|_2^2 \\
&= \sum_{i=1}^{|\mathcal{S}|}(\mathbf{q}_i^{\top}\boldsymbol{\delta})^2 \\
&\leq \sum_{i=1}^{|\mathcal{S}|}\|\mathbf{q}_i\|_2^2\,\|\boldsymbol{\delta}\|_2^2 \\
&= \|\mathbf{Q}\|_F^2\,\|\boldsymbol{\delta}\|_2^2,
\end{aligned}$$

where $\mathbf{q}_i^{\top}$ denotes the ith row in matrix $\mathbf{Q}$, and $\mathbf{Q}$ denotes $\left(\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}}\right)^{-1}\mathbf{X}_{\mathcal{S}}^{\top}$. Because $\|\mathbf{Q}\|_F^2$ is bounded and $\|\boldsymbol{\delta}\|_2^2\to 0$, we have $\left\|\left(\mathbf{X}_{\mathcal{S}}^{\top}\mathbf{X}_{\mathcal{S}}\right)^{-1}\mathbf{X}_{\mathcal{S}}^{\top}\boldsymbol{\delta}\right\|_2^2\to 0$.

For $\left\|\mathbf{X}_{\mathcal{S}^c}^{\dagger}\boldsymbol{\delta}\right\|_2^2$ in Equation 4.55, following the similar logic, we have

$$\left\|\mathbf{X}_{\mathcal{S}^c}^{\dagger}\boldsymbol{\delta}\right\|_2^2 \leq \left\|\mathbf{X}_{\mathcal{S}^c}^{\dagger}\right\|_F^2\|\boldsymbol{\delta}\|_2^2,$$

Because $\left\|\mathbf{X}_{\mathcal{S}^c}^{\dagger}\right\|_F^2$ is bounded and $\|\boldsymbol{\delta}\|_2^2\to 0$, we have $\left\|\mathbf{X}_{\mathcal{S}^c}^{\dagger}\boldsymbol{\delta}\right\|_2^2$.

For $\lambda_{\min}(\mathbf{M})$ in Equation 4.55, let us start with a general eigenvalue of matrix $\mathbf{M}$, and we denote the eigenvalue of $\mathbf{M}$ as $\lambda(\mathbf{M})$. If we prove that all the eigenvalue of matrix $\mathbf{M}$ is strictly larger than 0, than $\frac{1}{\lambda_{\min}}(\mathbf{M})$ can be bounded. This is equivalent to prove that $\mathbf{M}-\lambda(\mathbf{M})\mathbf{I}$ is positive semidefinite for any eigenvalue $\lambda(\mathbf{M})$.

If we denote $\mathbf{M}^* = \frac{\mathbf{M}+\mathbf{M}^\top}{2}$, then we notice that $\lambda(\mathbf{M}) = \lambda(\mathbf{M}^*)$. We will verify that $\mathbf{M}^* - \lambda(\mathbf{M})\mathbf{I}$ is positive semidefinite under the conditions of Proposition 4.5.2. To verify it, we know that for any $\boldsymbol{\alpha}, \boldsymbol{\beta}$, we have

$$
\begin{aligned}
& \begin{pmatrix} \boldsymbol{\alpha}^\top & \boldsymbol{\beta}^\top \end{pmatrix} \mathbf{M}^* \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} \\
= & \begin{pmatrix} \boldsymbol{\alpha}^\top & \boldsymbol{\beta}^\top \end{pmatrix} \begin{pmatrix} (1-\lambda)\mathbf{I} & \frac{\mathbf{A}+\mathbf{B}^\top}{2} \\ \frac{\mathbf{A}^\top+\mathbf{B}}{2} & \frac{1}{2}\mathbf{X}_{\mathcal{S}^c}^\dagger \mathbf{X}_{\mathcal{S}^c} + \frac{1}{2}\left(\mathbf{X}_{\mathcal{S}^c}^\dagger \mathbf{X}_{\mathcal{S}^c}\right)^\top - \lambda\mathbf{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} \\
= & (1-\lambda)\left\|\boldsymbol{\alpha}\right\|_2^2 + \boldsymbol{\beta}^\top \left[\frac{1}{2}\mathbf{X}_{\mathcal{S}^c}^\dagger \mathbf{X}_{\mathcal{S}^c} + \frac{1}{2}\left(\mathbf{X}_{\mathcal{S}^c}^\dagger \mathbf{X}_{\mathcal{S}^c}\right)^\top - \lambda\mathbf{I}\right]\boldsymbol{\beta} + \\
& \boldsymbol{\alpha}^\top(\mathbf{A}+\mathbf{B}^\top)\boldsymbol{\beta}, \qquad (4.56)
\end{aligned}
$$

where $\mathbf{A} = \left(\mathbf{X}_{\mathcal{S}}^\top \mathbf{X}_{\mathcal{S}}\right)^{-1} \mathbf{X}_{\mathcal{S}}^\top \mathbf{X}_{\mathcal{S}^c}$, $\mathbf{B} = \mathbf{X}_{\mathcal{S}^c}^\dagger \mathbf{X}_{\mathcal{S}}$. For the last term in Equation 4.56, we can apply SVD to $\mathbf{A}+\mathbf{B}^\top$, i.e., $\mathbf{A}+\mathbf{B}^\top = \mathbf{U}_1\boldsymbol{\Sigma}_1\mathbf{V}_1$, then we have

$$
\begin{aligned}
|\boldsymbol{\alpha}^\top(\mathbf{A}+\mathbf{B}^\top)\boldsymbol{\beta}| &= \boldsymbol{\alpha}^\top \mathbf{U}_1 \boldsymbol{\Sigma}_1 V_1 \boldsymbol{\beta} \\
&\leq \sigma_{\max}(\boldsymbol{\Sigma}_1)\langle \boldsymbol{\alpha}^\top \mathbf{U}_1, \mathbf{V}_1\boldsymbol{\beta}\rangle \\
&\leq \sigma_{\max}(\boldsymbol{\Sigma}_1)\left\|\boldsymbol{\alpha}^\top \mathbf{U}_1\right\|_2 \left\|\mathbf{V}_1\boldsymbol{\beta}\right\|_2 \\
&\leq \sigma_{\max}(\boldsymbol{\Sigma}_1)\left\|\boldsymbol{\alpha}^\top\right\|_2 \left\|\boldsymbol{\beta}\right\|_2 \\
&\leq \frac{1}{2}\sigma_{\max}(\boldsymbol{\Sigma}_1)\left(\left\|\boldsymbol{\alpha}^\top\right\|_2^2 + \left\|\boldsymbol{\beta}\right\|_2^2\right),
\end{aligned}
$$

where $\sigma_{\max}(\boldsymbol{\Sigma}_1)$ is the maximal absolute value in the diagonal entry of $\boldsymbol{\Sigma}_1$.

By plugging the above result into Equation 4.56, we have

$$
\begin{aligned}
& \begin{pmatrix} \boldsymbol{\alpha}^\top & \boldsymbol{\beta}^\top \end{pmatrix} \mathbf{M}^* \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} \\
\geq & \ (1-\lambda)\left\|\boldsymbol{\alpha}\right\|_2^2 + \boldsymbol{\beta}^\top \left[\frac{1}{2}\mathbf{X}_{\mathcal{S}^c}^\dagger \mathbf{X}_{\mathcal{S}^c} + \frac{1}{2}\left(\mathbf{X}_{\mathcal{S}^c}^\dagger \mathbf{X}_{\mathcal{S}^c}\right)^\top - \lambda \mathbf{I}\right] \boldsymbol{\beta} \\
& -|\boldsymbol{\alpha}^\top (\mathbf{A}+\mathbf{B}^\top)\boldsymbol{\beta}| \\
\geq & \ (1-\lambda)\left\|\boldsymbol{\alpha}\right\|_2^2 + \\
& \boldsymbol{\beta}^\top \left[\frac{1}{2}\mathbf{X}_{\mathcal{S}^c}^\dagger \mathbf{X}_{\mathcal{S}^c} + \frac{1}{2}\left(\mathbf{X}_{\mathcal{S}^c}^\dagger \mathbf{X}_{\mathcal{S}^c}\right)^\top - \lambda \mathbf{I}\right] \boldsymbol{\beta} \\
& -\frac{1}{2}\sigma_{\max}(\boldsymbol{\Sigma}_1)\left(\left\|\boldsymbol{\alpha}^\top\right\|_2^2 + \left\|\boldsymbol{\beta}\right\|_2^2\right) \\
= & \left(1-\lambda-\frac{1}{2}\sigma_{\max}(\boldsymbol{\Sigma}_1)\right)\left\|\boldsymbol{\alpha}\right\|_2^2 + \\
& \boldsymbol{\beta}^\top \left[\frac{1}{2}\mathbf{X}_{\mathcal{S}^c}^\dagger \mathbf{X}_{\mathcal{S}^c} + \frac{1}{2}\left(\mathbf{X}_{\mathcal{S}^c}^\dagger \mathbf{X}_{\mathcal{S}^c}\right)^\top - \left(\lambda + \frac{1}{2}\sigma_{\max}(\boldsymbol{\Sigma}_1)\right)\mathbf{I}\right] \boldsymbol{\beta},
\end{aligned}
\tag{4.57}
$$

where $\sigma_{\max}\left(\boldsymbol{\Sigma}_1\right)$ is the maximal absolute diagonal value of matrix Σ_1. Because we have $\sigma\left(\boldsymbol{\Sigma}_1\right) < 2$, so the first term in Equation 4.57 is greater than 0. Besides, because the minimal singular value of $\frac{1}{2}\mathbf{X}_{\mathcal{S}^c}^\dagger \mathbf{X}_{\mathcal{S}^c} + \frac{1}{2}\left(\mathbf{X}_{\mathcal{S}^c}^\dagger \mathbf{X}_{\mathcal{S}^c}\right)^\top$ is larger than $\frac{1}{2}\sigma_{\max}(\boldsymbol{\Sigma}_1)$, i.e., $\frac{1}{2}\mathbf{X}_{\mathcal{S}^c}^\dagger \mathbf{X}_{\mathcal{S}^c} + \frac{1}{2}\left(\mathbf{X}_{\mathcal{S}^c}^\dagger \mathbf{X}_{\mathcal{S}^c}\right)^\top = \mathbf{U}_2\boldsymbol{\Sigma}_2\mathbf{V}_2$ and $2\sigma_{\min}\left(\boldsymbol{\Sigma}_2\right) > \sigma_{\max}\left(\boldsymbol{\Sigma}_1\right)$, the second term in Equation 4.57 is also greater than 0. Thus, we prove that $\mathbf{M}^*$ is a positive semidefinite matrix, whose eigenvalue would be strictly larger than 0. According, $\mathbf{M}$, which shares the same eigenvalue with $\mathbf{M}^*$ also has eigenvalues strictly larger than 0. So we have $\frac{1}{\lambda_{\min}(\mathbf{M})}$ bounded.

In conclusion, because $\lambda_{\min}\left(\mathbf{M}\right)$ is bounded, $\left\|\left(\mathbf{X}_{\mathcal{S}}^\top \mathbf{X}_{\mathcal{S}}\right)^{-1}\mathbf{X}_{\mathcal{S}}^\top \boldsymbol{\delta}\right\|_2^2 \to 0$, and $\left\|\mathbf{X}_{\mathcal{S}^c}^\dagger \boldsymbol{\delta}\right\|_2^2 \to 0$, we have

$$
\left\|\widetilde{\boldsymbol{\beta}} - \widehat{\boldsymbol{\beta}}\right\|_2^2 = \left\|\widetilde{\boldsymbol{\beta}}_{\mathcal{S}} - \widehat{\boldsymbol{\beta}}_{\mathcal{S}}\right\|_2^2 + \left\|\widetilde{\boldsymbol{\beta}}_{\mathcal{S}^c}\right\|_2^2 \to 0.
$$

□

CHAPTER 5
CONCLUSION AND FUTURE RESEARCH

In this chapter, we summarize the contribution of the previous four chapters and discuss the future works.

5.1 Summary of Our Contributions

This thesis have four main works, and each work is presented in one chapter. Chapter 1 discusses the hot-spots detection in spatio-temporal data. Chapter 2 improves the methodology in Chapter 1. Chapter 3 proposes the SAPDEMI method to identify the underlying PDE models from noisy data. Chapter 4 develops an efficient algorithm to solve the Lasso-type problem. The contributions of these four chapters are summarized as follows.

- **Chapter 1**:

  In this chapter, we have three contributions. First, we focus on the hot-spots detection in the non-stationary multivariate spatio-temporal data, while most of the existing work focuses on either univariate spatio-temporal data, or the spatio-temporal data with i.i.d. background or following simple probability distribution. Second, our research work helps with sparse hot-spots detection in multivariate spatio-temporal data, while the traditional SPC mainly focuses on detecting global or system-wise hot-spots. Third, we propose to use tensor decomposition method to model multivariate spatio-temporal data, which can be easily extended to more complicated spatio-temporal data.

- **Chapter 2**:

  In this chapter, we improve the methodology in Chapter 1 both statistically and computationally. The statistical improvement is the capability to detect hot-spots with

temporal circularity, instead of temporal consistency as in Chapter 1. The computational improvement is the development of a more efficient algorithm with lower computational complexity.

- **Chapter 3**:

  The statistical contribution in this chapter is that we realize the identification of underlying dynamic models, which can also be called "model selection" in the statistics terminology, while lots of existing literature focus on "parameter estimation". Meanwhile, we establish the statistical properties of our method in the context of model selection, which has not been reported in the literature. The computational contribution of this chapter is that our proposed method is computationally efficient in the functional estimation stage, since it only requires the computational complexity of the linear polynomial of sample size, which achieves the theoretical minimal lower bound.

- **Chapter 4**:

  We propose an efficient algorithm to solve the Lasso problem, which is faster than the state-of-the-art algorithms for Lasso. Namely, the state-of-the-art algorithms can only guarantee $O(1/\epsilon)$ or $O(1/\sqrt{\epsilon})$ convergence rates, while we can prove an $O([\log(1/\epsilon)]^2)$ for the newly proposed algorithm.

5.2 Future Research

This thesis contributes to three areas: (1) hot-spots detection in spatial-temporal data, (2) PDE-based model identification, and (3) optimization in the Lasso-type problem. For the above three areas, there are some promising future research ideas, whose detailed description is listed as follows.

- **hot-spots detection in spatial-temporal data:**

  In both Chapter 1 and Chapter 2, we assume that after applying some transforma-

tions to the raw data, the transformed raw data would follow the normal distribution. Specifically, in Chapter 1, we assume that the logarithm of the number of crime events follows the normal distribution, and in Chapter 2, we assume that the logarithm of the number of gonorrhea events follows the normal distribution. It can be seen that, the datasets we use in Chapter 1 and Chapter 2 – the number of crime/gonorrhea events – are both count data. In the future, we would like to research on the Poisson distributions and take the effect of popularization size into considerations.

- **PDE-based model identification:**

  In this area, there are three promising future research directions. First, in Chapter 3, we take the spatial variable $x \in \mathbb{R}$ as an illustration example, and it would be interesting to investigate the case when the spatial variable $\mathbf{x} \in \mathbb{R}^d$ ($d \geq 2$) due to its wide existence in practice. Second, future research could consider the interaction between the spatial variable and the temporal variable. For instance, we can explore the time-varying coefficient $\boldsymbol{\beta}(t) = (\beta_1(t), \ldots, \beta_K(t))^\top$ in Equation 3.3. Besides, we can also consider the case when the ϵ_i^n in Equation 3.2 has spatial-temporal patterns. Third, in our paper, we utilize the cubic spline in the functional estimation stage, and it would be interesting to investigate other fitting methods, such as B-splines, reproducing kernel Hilbert space and so on.

- **optimization in the Lasso-type problem:**

  In Chapter 4, we take the Lasso problem as an illustration example, whose lose function is the summation of squared residual, i.e., $\|y - X\beta\|_2^2$. Future research could consider generalized loss functions, for instance, the Huber loss, the 0-1 loss function and so on. Moreover, future research can investigate the necessary and sufficient conditions of the surrogate function paths to realize the homotopic method.

REFERENCES

[1] J. I. Naus, *Clustering of random points in line and plane*. Harvard University Press, 1963.

[2] M. Kulldorff, "A spatial scan statistic," *Communications in Statistics-Theory and methods*, vol. 26, no. 6, pp. 1481–1496, 1997.

[3] T. Tango, K. Takahashi, and K. Kohriyama, "A space–time scan statistic for detecting emerging outbreaks," *Biometrics*, vol. 67, no. 1, pp. 106–115, 2011.

[4] D. B. Neill, A. W. Moore, M. Sabhnani, and K. Daniel, "Detection of emerging space-time clusters," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 2005, pp. 218–227.

[5] M. Kulldorff, "Prospective time periodic geographical disease surveillance using a scan statistic," *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, vol. 164, no. 1, pp. 61–72, 2001.

[6] D. B. Neill, A. W. Moore, and G. F. Cooper, "A bayesian spatial scan statistic," in *Advances in neural information processing systems*, 2006, pp. 1003–1010.

[7] R. Tibshirani, "Regression shrinkage and selection via the Lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[8] C. Zou and P. Qiu, "Multivariate statistical process control using Lasso," *Journal of the American Statistical Association*, vol. 104, no. 488, pp. 1586–1596, 2009.

[9] C. Zou, F. Tsung, and Z. Wang, "Monitoring profiles based on nonparametric regression methods," *Technometrics*, vol. 50, no. 4, pp. 512–526, 2008.

[10] C. Zou, X. Ning, and F. Tsung, "Lasso-based multivariate linear profile monitoring," *Annals of Operations Research*, vol. 192, no. 1, pp. 3–19, 2012.

[11] J. Šaltytė Benth and L. Šaltytė, "Spatial–temporal model for wind speed in lithuania," *Journal of Applied Statistics*, vol. 38, no. 6, pp. 1151–1168, 2011.

[12] H. Yan, K. Paynabar, and J. Shi, "Real-time monitoring of high-dimensional functional data streams via spatio-temporal smooth sparse decomposition," *Technometrics*, vol. 60, no. 2, pp. 181–197, 2018.

[13] H. Yan, M. Grasso, K. Paynabar, and B. M. Colosimo, "Real-time detection of clustered events in video-imaging data with applications to additive manufacturing," *IISE Transactions*, pp. 1–22, 2021.

[14] R. Y. Liu, "Control charts for multivariate processes," *Journal of the American Statistical Association*, vol. 90, no. 432, pp. 1380–1387, 1995.

[15] K. Paynabar, C. Zou, and P. Qiu, "A change-point approach for phase-i analysis in multivariate profile monitoring and diagnosis," *Technometrics*, vol. 58, no. 2, pp. 191–204, 2016.

[16] K. Paynabar, J. Jin, and M. Pacella, "Monitoring and diagnosis of multichannel nonlinear profile variations using uncorrelated multilinear principal component analysis," *Iie transactions*, vol. 45, no. 11, pp. 1235–1247, 2013.

[17] B. De Ketelaere, M. Hubert, and E. Schmitt, "Overview of PCA-based statistical process-monitoring methods for time-dependent, high-dimensional data," *Journal of Quality Technology*, vol. 47, no. 4, pp. 318–335, 2015.

[18] D. Louwerse and A. Smilde, "Multivariate statistical process control of batch processes based on three-way models," *Chemical Engineering Science*, vol. 55, no. 7, pp. 1225–1235, 2000.

[19] K. Hu and J. Yuan, "Batch process monitoring with tensor factorization," *Journal of Process Control*, vol. 19, no. 2, pp. 288–296, 2009.

[20] B. R. Bakshi, "Multiscale PCA with application to multivariate statistical process monitoring," *AIChE journal*, vol. 44, no. 7, pp. 1596–1610, 1998.

[21] H. Yan, K. Paynabar, and J. Shi, "Image-based process monitoring using low-rank tensor decomposition," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, pp. 216–227, 2014.

[22] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight, "Sparsity and smoothness via the fused Lasso," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 1, pp. 91–108, 2005.

[23] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[24] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[25] R. J. Tibshirani, J. Taylor, *et al.*, "The solution path of the generalized Lasso," *The Annals of Statistics*, vol. 39, no. 3, pp. 1335–1371, 2011.

[26] B. Wahlberg, S. Boyd, M. Annergren, and Y. Wang, "An ADMM algorithm for a class of total variation regularized estimation problems," *IFAC Proceedings Volumes*, vol. 45, no. 16, pp. 83–88, 2012.

[27] B. He and X. Yuan, "On non-ergodic convergence rate of douglas–rachford alternating direction method of multipliers," *Numerische Mathematik*, vol. 130, no. 3, pp. 567–577, 2015.

[28] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 57, no. 11, pp. 1413–1457, 2004.

[29] J. Liu, L. Yuan, and J. Ye, "An efficient algorithm for a class of fused Lasso problems," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2010, pp. 323–332.

[30] Y. Zhu, "An augmented ADMM algorithm with application to the generalized Lasso problem," *Journal of Computational and Graphical Statistics*, vol. 26, no. 1, pp. 195–204, 2017.

[31] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. Springer US, 1986.

[32] D. M. Hawkins, "Regression adjustment for variables in multivariate quality control," *Journal of Quality Technology*, vol. 25, no. 3, pp. 170–182, 1993.

[33] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.

[34] Lorden, "Procedures for reacting to a change in distribution," *The Annals of Mathematical Statistics*, vol. 42, no. 6, pp. 1897–1908, 1971.

[35] X. Fang, K. Paynabar, and N. Gebraeel, "Image-based prognostics using penalized tensor regression," *Technometrics*, vol. 61, no. 3, pp. 369–384, 2019.

[36] P. Qiu, *Introduction to Statistical Process Control*. Chapman and Hall/CRC, Oct. 2013.

[37] Y.-X. Wang, J. Sharpnack, A. J. Smola, and R. J. Tibshirani, "Trend filtering on graphs," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 3651–3691, 2016.

[38] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[39] C. Zou and P. Qiu, "Multivariate statistical process control using Lasso," *Journal of the American Statistical Association*, vol. 104, no. 488, pp. 1586–1596, 2009.

[40] L. Tran, C. Navasca, and J. Luo, "Video detection anomaly via low-rank and sparse decompositions," in *2012 Western New York Image Processing Workshop*, IEEE, 2012, pp. 17–20.

[41] H. Yan, K. Paynabar, and J. Shi, "Anomaly detection in images with smooth background via smooth-sparse decomposition," *Technometrics*, vol. 59, no. 1, pp. 102–114, 2017.

[42] E. J. Hannan and B. G. Quinn, "The determination of the order of an autoregression," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 190–195, 1979.

[43] J. D. Hamilton, *Time series analysis*. Princeton university press Princeton, NJ, 1994, vol. 2.

[44] K. Reynolds and L. Madden, "Analysis of epidemics using spatio-temporal autocorrelation," *Phytopathology*, vol. 78, no. 2, pp. 240–246, 1988.

[45] J. W. Lichstein, T. R. Simons, S. A. Shriner, and K. E. Franzreb, "Spatial autocorrelation and autoregressive models in ecology," *Ecological monographs*, vol. 72, no. 3, pp. 445–463, 2002.

[46] H. Lan, C. Zhou, L. Wang, H. Zhang, and R. Li, "Landslide hazard spatial analysis and prediction using gis in the xiaojiang watershed, yunnan, china," *Engineering geology*, vol. 76, no. 1-2, pp. 109–128, 2004.

[47] J. P. Elhorst, "Spatial panel data models," in *Spatial econometrics*, Springer, 2014, pp. 37–93.

[48] M. A. Call and P. R. Voss, "Spatio-temporal dimensions of child poverty in america, 1990–2010," *Environment and Planning A*, vol. 48, no. 1, pp. 172–191, 2016.

[49] J. Zhu, H.-C. Huang, and J. Wu, "Modeling spatial-temporal binary data using markov random fields," *Journal of Agricultural, Biological, and Environmental Statistics*, vol. 10, no. 2, p. 212, 2005.

[50] T. L. Lai and J. Lim, "Asymptotically efficient parameter estimation in hidden markov spatio-temporal random fields," *Statistica Sinica*, pp. 403–421, 2015.

[51] P. J. Diggle, *Statistical analysis of spatial and spatio-temporal point patterns*. CRC Press, 2013.

[52] G. C. Reinsel, *Elements of multivariate time series analysis*. Springer Science & Business Media, 2003.

[53] R. S. Tsay, *Multivariate time series analysis: with R and financial applications*. John Wiley & Sons, 2013.

[54] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[55] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 57, no. 11, pp. 1413–1457, 2004.

[56] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *Journal of statistical software*, vol. 33, no. 1, p. 1, 2010.

[57] P. J. Olver, *Introduction to partial differential equations*. Springer, 2014.

[58] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical learning with sparsity: the Lasso and generalizations*. CRC press, 2015.

[59] H. Liang and H. Wu, "Parameter estimation for differential equation models using a framework of measurement error in regression models," *Journal of the American Statistical Association*, vol. 103, no. 484, pp. 1570–1583, 2008.

[60] H. Wu, H. Xue, and A. Kumar, "Numerical discretization-based estimation methods for ordinary differential equation models via penalized spline smoothing with applications in biomedical research," *Biometrics*, vol. 68, no. 2, pp. 344–352, 2012.

[61] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the national academy of sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[62] G. Tran and R. Ward, "Exact recovery of chaotic systems from highly corrupted data," *Multiscale Modeling & Simulation*, vol. 15, no. 3, pp. 1108–1129, 2017.

[63] C. W. Ueberhuber, *Numerical computation 1: methods, software, and analysis*. Springer Science & Business Media, 2012.

[64] R. Butt, *Introduction to numerical analysis using MATLAB*. Laxmi Publications, Ltd., 2008.

[65] B. Fornberg, "Numerical differentiation of analytic functions," *ACM Transactions on Mathematical Software (TOMS)*, vol. 7, no. 4, pp. 512–526, 1981.

[66] W. Squire and G. Trapp, "Using complex variables to estimate derivatives of real functions," *SIAM review*, vol. 40, no. 1, pp. 110–112, 1998.

[67] M. Bär, R. Hegger, and H. Kantz, "Fitting partial differential equations to space-time dynamics," *Physical Review E*, vol. 59, no. 1, p. 337, 1999.

[68] H. Schaeffer, "Learning partial differential equations via data discovery and sparse optimization," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 473, no. 2197, p. 20 160 446, 2017.

[69] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Data-driven discovery of partial differential equations," *Science Advances*, vol. 3, no. 4, e1602614, 2017.

[70] U. Parlitz and C. Merkwirth, "Prediction of spatiotemporal time series based on reconstructed local states," *Physical review letters*, vol. 84, no. 9, p. 1890, 2000.

[71] H. U. Voss, P. Kolodner, M. Abel, and J. Kurths, "Amplitude equations from spatiotemporal binary-fluid convection data," *Physical review letters*, vol. 83, no. 17, p. 3422, 1999.

[72] J. O. Ramsay, G. Hooker, D. Campbell, and J. Cao, "Parameter estimation for differential equations: A generalized smoothing approach," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 69, no. 5, pp. 741–796, 2007.

[73] J. O. Ramsay, "Principal differential analysis: Data reduction by differential operators," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 3, pp. 495–508, 1996.

[74] X. Xun, J. Cao, B. Mallick, A. Maity, and R. J. Carroll, "Parameter estimation of partial differential equation models," *Journal of the American Statistical Association*, vol. 108, no. 503, pp. 1009–1020, 2013.

[75] D. Wang, K. Liu, and X. Zhang, "Spatiotemporal thermal field modeling using partial differential equations with time-varying parameters," *IEEE Transactions on Automation Science and Engineering*, 2019.

[76] H. Miao, C. Dykes, L. M. Demeter, and H. Wu, "Differential equation modeling of HIV viral fitness experiments: Model identification, model selection, and multimodel inference," *Biometrics*, vol. 65, no. 1, pp. 292–300, 2009.

[77] L. Azzimonti, L. M. Sangalli, P. Secchi, M. Domanin, and F. Nobile, "Blood flow velocity field estimation via spatial regression with PDE penalization," *Journal of the American Statistical Association*, vol. 110, no. 511, pp. 1057–1071, 2015.

[78] S. H. Kang, W. Liao, and Y. Liu, "Ident: Identifying differential equations with numerical time evolution," *arXiv preprint arXiv:1904.03538*, 2019.

[79] N. M. Mangan, J. N. Kutz, S. L. Brunton, and J. L. Proctor, "Model selection for dynamical systems via sparse regression and information criteria," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 473, no. 2204, p. 20 170 009, 2017.

[80] N. M. Mangan, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Inferring biological networks by sparse identification of nonlinear dynamics," *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, vol. 2, no. 1, pp. 52–63, 2016.

[81] H. Schaeffer, R. Caflisch, C. D. Hauck, and S. Osher, "Sparse dynamics for partial differential equations," *Proceedings of the National Academy of Sciences*, vol. 110, no. 17, pp. 6634–6639, 2013.

[82] S. McKinley and M. Levine, "Cubic spline interpolation," *College of the Redwoods*, vol. 45, no. 1, pp. 1049–1060, 1998.

[83] J. Ahlberg, J. Walsh, R. Bellman, and E. N. Nilson, *The theory of splines and their applications*. Academic press, 1967.

[84] S. G. Rubin and R. A. Graves Jr, "A cubic spline approximation for problems in fluid mechanics," *NASA STI/Recon Technical Report N*, vol. 75, p. 33 345, 1975.

[85] J. Rashidinia and R. Mohammadi, "Non-polynomial cubic spline methods for the solution of parabolic equations," *International Journal of Computer Mathematics*, vol. 85, no. 5, pp. 843–850, 2008.

[86] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *Journal of statistical software*, vol. 33, no. 1, p. 1, 2010.

[87] A. Beck and L. Tetruashvili, "On the convergence of block coordinate descent type methods," *SIAM journal on Optimization*, vol. 23, no. 4, pp. 2037–2060, 2013.

[88] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization," *Journal of optimization theory and applications*, vol. 109, no. 3, pp. 475–494, 2001.

[89] B. W. Silverman, "Spline smoothing: The equivalent variable kernel method," *The Annals of Statistics*, pp. 898–916, 1984.

[90] R. Haberman, *Elementary applied partial differential equations*. Prentice Hall Englewood Cliffs, NJ, 1983, vol. 987.

[91] J. Fan, T. Gasser, I. Gijbels, M. Brockmann, and J. Engel, "Local polynomial regression: Optimal kernels and asymptotic minimax efficiency," *Annals of the Institute of Statistical Mathematics*, vol. 49, no. 1, pp. 79–99, 1997.

[92] K. Messer *et al.*, "A comparison of a spline estimate to its equivalent kernel estimate," *The Annals of Statistics*, vol. 19, no. 2, pp. 817–829, 1991.

[93] P. Craven and G. Wahba, "Smoothing noisy data with spline functions," *Numerische mathematik*, vol. 31, no. 4, pp. 377–403, 1978.

[94] J. Rice and M. Rosenblatt, "Smoothing splines: Regression, derivatives and deconvolution," *The annals of Statistics*, pp. 141–156, 1983.

[95] Y.-p. Mack and B. W. Silverman, "Weak and strong uniform consistency of kernel regression estimates," *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, vol. 61, no. 3, pp. 405–415, 1982.

[96] G. Tusnády, "A remark on the approximation of the sample df in the multidimensional case," *Periodica Mathematica Hungarica*, vol. 8, no. 1, pp. 53–55, 1977.

[97] M. Rosenblatt, "Remarks on a multivariate transformation," *The annals of mathematical statistics*, vol. 23, no. 3, pp. 470–472, 1952.

[98] B. W. Silverman, "Weak and strong uniform consistency of the kernel estimate of a density and its derivatives," *The Annals of Statistics*, pp. 177–184, 1978.

[99] A. Winkelbauer, "Moments and absolute moments of the normal distribution," *arXiv preprint arXiv:1209.4340*, 2012.

[100] L. M. Brègman, "Relaxation method for finding a common point of convex sets and its application to optimization problems," in *Doklady Akademii Nauk*, Russian Academy of Sciences, vol. 171, 1966, pp. 1019–1022.

[101] M. R. Hestenes, "Multiplier and gradient methods," *Journal of optimization theory and applications*, vol. 4, no. 5, pp. 303–320, 1969.

[102] R. T. Rockafellar, "Monotone operators and the proximal point algorithm," *SIAM journal on control and optimization*, vol. 14, no. 5, pp. 877–898, 1976.

[103] N. G. Polson, J. G. Scott, B. T. Willard, *et al.*, "Proximal algorithms in statistics and machine learning," *Statistical Science*, vol. 30, no. 4, pp. 559–581, 2015.

[104] K. Lange, D. R. Hunter, and I. Yang, "Optimization transfer using surrogate objective functions," *Journal of computational and graphical statistics*, vol. 9, no. 1, pp. 1–20, 2000.

[105] D. R. Hunter and R. Li, "Variable selection using mm algorithms," *Annals of statistics*, vol. 33, no. 4, p. 1617, 2005.

[106] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[107] Y. E. Nesterov, "A method for solving the convex programming problem with convergence rate o (1/k$\hat{2}$)," in *Dokl. akad. nauk Sssr*, vol. 269, 1983, pp. 543–547.

[108] Y. Nesterov, "Smooth minimization of non-smooth functions," *Mathematical programming*, vol. 103, no. 1, pp. 127–152, 2005.

[109] ——, "Gradient methods for minimizing composite functions," *Mathematical Programming*, vol. 140, no. 1, pp. 125–161, 2013.

[110] H. Li and Z. Lin, "Accelerated proximal gradient methods for nonconvex programming," in *Advances in neural information processing systems*, 2015, pp. 379–387.

[111] S. Ghadimi and G. Lan, "Accelerated gradient methods for nonconvex nonlinear and stochastic programming," *Mathematical Programming*, vol. 156, no. 1-2, pp. 59–99, 2016.

[112] C. Hildreth, "A quadratic programming procedure," *Naval research logistics quarterly*, vol. 4, no. 1, pp. 79–85, 1957.

[113] J. Warga, "Minimizing certain convex functions," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 3, pp. 588–593, 1963.

[114] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization," *Journal of optimization theory and applications*, vol. 109, no. 3, pp. 475–494, 2001.

[115] W. J. Fu, "Penalized regressions: The bridge versus the Lasso," *Journal of computational and graphical statistics*, vol. 7, no. 3, pp. 397–416, 1998.

[116] S. K. Shevade and S. S. Keerthi, "A simple and efficient algorithm for gene selection using sparse logistic regression," *Bioinformatics*, vol. 19, no. 17, pp. 2246–2253, 2003.

[117] J. Friedman, T. Hastie, H. Höfling, R. Tibshirani, *et al.*, "Pathwise coordinate optimization," *The annals of applied statistics*, vol. 1, no. 2, pp. 302–332, 2007.

[118] T. T. Wu, K. Lange, *et al.*, "Coordinate descent algorithms for Lasso penalized regression," *The Annals of Applied Statistics*, vol. 2, no. 1, pp. 224–244, 2008.

[119] M. Schmidt, G. Fung, and R. Rosales, "Fast optimization methods for l1 regularization: A comparative study and two new approaches," in *European Conference on Machine Learning*, Springer, 2007, pp. 286–297.

[120] M. A. Figueiredo, "Adaptive sparseness for supervised learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 25, no. 9, pp. 1150–1159, 2003.

[121] S. Mukherjee and C. S. Seelamantula, "Convergence rate analysis of smoothed Lasso," in *Communication (NCC), 2016 Twenty Second National Conference on*, IEEE, 2016, pp. 1–6.

[122] R. J. Tibshirani, J. Taylor, *et al.*, "The solution path of the generalized Lasso," *The Annals of Statistics*, vol. 39, no. 3, pp. 1335–1371, 2011.

[123] S. Rosset and J. Zhu, "Piecewise linear regularized solution paths," *The Annals of Statistics*, pp. 1012–1030, 2007.

[124] M. Y. Park and T. Hastie, "L1-regularization path algorithm for generalized linear models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 69, no. 4, pp. 659–677, 2007.

[125] S. Bubeck, M. B. Cohen, Y. T. Lee, and Y. Li, "An homotopy method for ℓ_p regression provably beyond self-concordance and in input-sparsity time," in *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, ACM, 2018, pp. 1130–1137.

[126] L. Xiao and T. Zhang, "A proximal-gradient homotopy method for the sparse least-squares problem," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 1062–1091, 2013.

[127] Q. Lin and L. Xiao, "An adaptive accelerated proximal gradient method and its homotopy continuation for sparse optimization," in *International Conference on Machine Learning*, 2014, pp. 73–81.

[128] Z. Wang, H. Liu, and T. Zhang, "Optimal computational and statistical rates of convergence for sparse nonconvex learning problems," *Annals of statistics*, vol. 42, no. 6, p. 2164, 2014.

[129] T. Zhao, H. Liu, T. Zhang, *et al.*, "Pathwise coordinate optimization for sparse learning: Algorithm and theory," *The Annals of Statistics*, vol. 46, no. 1, pp. 180–218, 2018.

[130] H. Pang, H. Liu, R. J. Vanderbei, and T. Zhao, "Parametric simplex method for sparse learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 188–197.

[131] G. I. Allen, "Automatic feature selection via weighted kernels and regularization," *Journal of Computational and Graphical Statistics*, vol. 22, no. 2, pp. 284–299, 2013.

[132] G. I. Allen, C. Peterson, M. Vannucci, and M. Maletić-Savatić, "Regularized partial least squares with an application to nmr spectroscopy," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 6, no. 4, pp. 302–314, 2013.

[133] S. Boyd and L. Vandenberghe, *Numerical linear algebra background*, 2016.

[134] A. Beck and L. Tetruashvili, "On the convergence of block coordinate descent type methods," *SIAM journal on Optimization*, vol. 23, no. 4, pp. 2037–2060, 2013.

[135] G. Lan, "Lectures on optimization. methods for machine learning," *H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA*, 2019.