Supervised Functional PCA with Covariate Dependent Mean and Covariance Structure

Fei Ding, Shiyuan He, David Jones, Jianhua Huang

Abstract: Few functional data analysis methods incorporate covariate information, often referred to as supervision information. Most supervised functional principal components analysis (FPCA) methods that do incorporate covariate information assume that only the scores are related to the covariates, which is often violated in practice. In this article, we propose a framework to incorporate covariate information related to both the mean and covariance structure. To ensure the covariance matrix is positive definite, we design a map from Euclidean space to the symmetric positive-definite matrix manifold. We also develop an efficient estimation algorithm and propose a smoothness penalty. Cross-validation is used to choose the tuning parameters. We demonstrate the advantages of our approach through a simulation study and an astronomical data analysis.

Keywords and phrases: Supervised functional principal component analysis, Penalized likelihood, Covariate information.

1. Introduction

Functional data is becoming increasingly important in many scientific fields, including astronomy, biology, and neuroscience. Due to the prevalence of different types of functional data and associative computational challenges, many functional data analysis (FDA) methods have been developed, see Ramsay and Silverman (2005) for an overview.

In most of these FDA methods, principal component analysis (PCA) plays a pivotal role because it is usually necessary to reduce the dimensionality of the data. Functional PCA (FPCA) has been an important tool for many years and there are well-established modeling frameworks and fitting algorithms for its different variants. Most FPCA models are based on the work of James, Hastie and Sugar (2000), which established a mixed effect framework for irregular and sparse functional data. Fitting FPCA models is often performed using maximum likelihood or restricted maximum likelihood (REML) methods. The latter approach was proposed Peng and Paul (2009) and can estimate the eigenvalues and eigenfunctions of the covariance matrix in the case of irregular measurements. Paul et al. (2009) establish the consistency and rate of convergence for the REML estimator. Suarez et al. (2017) proposed an alternative PCA based method that approximates the covariance matrix eigenfunctions using spectral decomposition.

Some studies have characterized the relationship between multiple or paired longitudinal curves. For example, Zhou, Huang and Carroll (2008) provide a framework to connect two paired longitudinally observed variables via scores of PCA. Similarly, multidimensional functional data has also attracted interest. Kayano and Konishi (2009) explore multidimensional functional data estimation strategies based on Gaussian basis functions. Functional data methods have also been developed in both the nonparametric and Bayesian framework, because the nonparametric methods have less stringent assumptions than their parametric counterparts and the Bayesian framework can incorporate prior information. Cai and Yuan (2010) give a method which considers nonparametric covariance function estimation in the reproducing kernel Hilbert space. Some scholars try to treat the functional data method from Bayesian perspective, but there are few FPCA methods that are proposed in Bayesian framework because of difficulties such as it's hard to choose the functional space as sample space or as parameter space in FDA and it is also cost more computational cost compared to the frequentist framework. Van Der Linde (2008) suggests a Bayesian approach to analyze the functional data by variational inference.

Smoothness penalties are one of the fundamental components of FDA because they allow continuous control over smoothness. In contrast, controlling smoothness by limiting the number of basis functions is discontinuous in nature. It is therefore natural that our method also emphasizes the role of smoothness penalties. Pezzulli (1993) and Silverman et al. (1996) developed some of the first methods and theoretical insights related to smoothness penalties. Cai et al. (2011) established a minimax bounds on the convergence rate for the estimation of mean functions and gave situations under which smoothness is necessary.

Recently, supervised functional methods that incorporate covariates have gained popularity because covariates are often measured in addition to functional data and can help make predictions more accurate. The majority of recent supervised functional methods are based on Yao, Müller and Wang (2005), but all rely on local smoothing which leads to high computational cost. For example, Jiang et al. (2010) proposed an approach to accommodate covariate information using a local linear smoother. Other examples include Jiang et al. (2011), Zhang et al. (2016), and Zhang, Park and Wang (2013). Jiang et al. (2011) established a single index model for longitudinal data. Zhang et al. (2016) considered an approach that allows a general weighing scheme of a local smoother for different kinds of functional data. Zhang, Park and Wang (2013) extend the additive model for longitudinal data.

In this paper we develop a supervised FPCA framework to incorporate covariate information in a more computationally efficient manner. We name it *supervised functional PCA with positive definite manifold structure*, or SFPDM. Our method is an extension of the supervised sparse and functional principal component (SupSFPC) method proposed by Li, Shen and Huang (2016). SupSFPC incorporates supervision information by assuming that the scores vary linearly with covariates. However, the linear restriction and only allowing dependence on the covariates through the scores creates limitations and both assumptions are often violated in practice. Our method does not have these limitations and has performed better in all our numerical studies.

In classical FPCA, we denote by $x_n(t)$ the value of an underlying latent function at time t, for $n = 1, \dots, N$. More generally, we could consider functions of another variable, such as location, but here restrict our attention to functions of time. For each function $x_n(t)$, its covariance function $\text{cov}(x_n(t), x_n(t')) = G(t, t')$ can be decomposed as

$$G(t, t') = \sum_{k=1}^{\infty} d_k f_k(t) f_k(t'), \tag{1}$$

where f_k is the corresponding orthogonal unit-norm k-th eigenfunction and d_k is the k-th eigenvalue. By the Karhunen–Loeve theorem, the function $x_n(t)$ can be expressed by a linear combination of a mean function and eigenfunctions, i.e. $x_n(t) = \mu(t) + \sum_{j=1}^{\infty} \alpha_j^{(n)} f_j(t)$, where $\alpha_j^{(n)} = \int x_n(t) f_j(t) dt$ is random variable with mean 0 and variance d_j. Following James, Hastie and Sugar (2000) consider a mixed-effects model for functional data, in which the latent curve are approximated by a mean and a small number r of principal components, i.e.

$$y_n(t) = x_n(t) + \epsilon(t) \approx \mu(t) + \sum_{j=1}^{r} \alpha_j^{(n)} f_j(t) + \epsilon(t) = \mu(t) + \boldsymbol{f}^T(t)\boldsymbol{\alpha}^{(n)} + \epsilon(t), \tag{2}$$

where $\mu(t)$ is the mean function, $f_j(t)$ is the j-th eigenfunction of covariance with eigenvalue d_j, $d_1 \geq d_2 \geq d_3 \geq \cdots$, respctively, $j = 1, \dots, r$. $\alpha_j^{(n)}$ is the j-th score with $\alpha_j \sim \mathbb{N}(0, d_j)$. $\epsilon(t)$ denotes white noise with mean 0 and variance σ_e^2. On the right side of (2), we have used the vector notation $\boldsymbol{f}(t) = (f_1(t), \dots, f_r(t))^T$ and $\boldsymbol{\alpha}^{(n)} = (\alpha_1, \dots, \alpha_r)^T$. As data collection explodes i many areas, more and more functional data are accompanied by covariates $\boldsymbol{z}$. However, the mean function and eigenfunctions of classical FPCA do not depend on the covariates $\boldsymbol{z}$. In our approach both the mean function and the covariance structure depend on the covariates. Since the covariance structure is on a symmetric positive-definite matrix manifold, we construct a map from Euclidean space to symmetric positive-definite matrix manifold. Some recent papers explore methods to model manifold-valued data, e.g., Lin et al. (2017).

This article is organized in the following order. In Section 2 we first introduce our procedure in two aspects. One is modeling the mean functional and another is the covariance functions. After that, we focus on the smoothness penalty which is inspired by Wood (2006). Section 3 details our algorithm which uses some approximations to reduce computational cost. In Section 4 and 5, we compare our method with SupSFPC through simulation studies and an astronomical real data analysis.

2. Model

Since we assume that both mean and covariance structure all depend on covariate $\boldsymbol{z}_n$, we need to construct the maps between mean and covariance structure, respectively. For mean function, we model it by directly applying the tensor product spline basis. For convariance function, since the covariance structure is on a symmetric positive-definite matrix manifold, we model the covariance function by constructing a map from Euclidean space to a symmetric positive-definite matrix manifold.

For the n-th sample function x_n with $n = 1, \cdots, N$, we collected noisy observations at time points $t_1^{(n)}, \dots, t_{m_n}^{(n)} \in \mathcal{T}$, denoted as $\boldsymbol{y}_n = (y_n(t_1^{(n)}), \dots, y_n(t_{m_n}^{(n)}))^T$. In addition, the n-th observation is coupled with a covariate $\boldsymbol{z}_n$. We assume the covariate resides in a compact domain $\mathcal{Z}$.

2.1. Incorporating Covariate Information to Model the Mean function

The mean function $\mu(t)$ in the classical FPCA model (2) is allowed to vary smoothly with the covariate $\boldsymbol{z}$. The model is parameterized by B-spline basis. In particular, cubic spline is constructed inside the domain $\mathcal{T}$ with l' equally spaced knots. We get a basis vector $\boldsymbol{a}(t) \in \mathbb{R}^l$ with degree of freedom $l = l' + 4$. Similarly, we also construct basis vector $\boldsymbol{c}(\boldsymbol{z}) \in \mathbb{R}^p$ with degree of freedom p. Via the tensor product of these basis functions, the mean function can be expressed as

$$\mu(t, \boldsymbol{z}) = \boldsymbol{a}^T(t)\boldsymbol{\Theta}_\mu \boldsymbol{c}(\boldsymbol{z}) = \sum_{i=1}^{l}\sum_{j=1}^{p} a_i(t)c_j(\boldsymbol{z})\theta_{ij} = \mathbf{H}(t, \boldsymbol{z})\boldsymbol{\theta}_\mu. \tag{3}$$

In the above, $\boldsymbol{\Theta}_\mu = \big(\theta_{ij}\big) \in \mathbb{R}^{l\times p}$ is a matrix of basis coefficients. In addition, $\mathbf{H}(t, \boldsymbol{z}) = \boldsymbol{b}(t) \otimes \boldsymbol{c}(\boldsymbol{z})$ is the tensor product spline basis constructed from $\boldsymbol{a}(t)$and $\boldsymbol{c}(\boldsymbol{z})$, and $\boldsymbol{\theta}_\mu = \mathrm{vec}(\boldsymbol{\Theta}_\mu)$.

2.2. Incorporating Covariate Information to Model the Covariance Function

In classical FPCA, the covariance function $\mathrm{cov}(x_n(t), x_n(t')) = G(t, t')$ is approximated by

$$G(t, t') \approx \sum_{k=1}^{r} d_k f_k(t) f_k(t'), \tag{4}$$

where f_k is the k-th eigenfunction, d_k is the k-th eigenvalue. Here we first expand the k-th eigenfunction $f_k(t)$ by spline basis $\boldsymbol{b}(t) \in \mathbb{R}^m$, i.e., $f_k(t) = \boldsymbol{b}(t)^T\boldsymbol{\theta}_k$, for $k = 1\cdots r$, where $\boldsymbol{\theta}_k$ is the vector of coefficients for the k-th eigenfunction. Following that, the covariance function approximation above (4) can be expressed as

$$G(t, t^{'}) \approx \boldsymbol{b}(t)^T \boldsymbol{\Theta}\mathbf{D}\boldsymbol{\Theta}^T \boldsymbol{b}(t) = \boldsymbol{b}(t)^T \boldsymbol{\Sigma}\boldsymbol{b}(t), \tag{5}$$

where $\mathbf{D} = \mathrm{diag}(d_1, \cdots, d_r)$ is a diagonal matrix containing the eigenvalues and $\boldsymbol{\Theta}$ is the coefficient matrix whose k-th column is $\boldsymbol{\theta}_k$.

Since we want the covariance function G to depend on the covariates $\boldsymbol{z}$, inspired by the structure of covariance function approximation in equation 5 and we now capture that dependence by modeling $\boldsymbol{\Sigma}$ there. Notice that the covariance function is symmetric positive-definite (SPD), and this lead to the SPD matrix requirement of $\boldsymbol{\Sigma}$ when capturing dependence on $\boldsymbol{z}$. To ensure that $\boldsymbol{\Sigma}$ is a SPD matrix, we construct a map $\boldsymbol{\Sigma}(\cdot, \cdot)$ from Euclidean space to a SPD matrix manifold $\mathrm{Sym}^+(m)$. Inspired by the work of Zhu et al. (2009), we simultaneously ensure that $\boldsymbol{\Sigma} \in \mathrm{Sym}^+(m)$ and specify $\boldsymbol{\Sigma}$ by restricting it to be of the form,

$$\boldsymbol{\Sigma}(\boldsymbol{u}, \boldsymbol{\beta}) = \mathbf{C}(\boldsymbol{z}, \boldsymbol{\beta})\mathbf{C}(\boldsymbol{z}, \boldsymbol{\beta})^T, \tag{6}$$

where $\mathbf{C}(\boldsymbol{z}, \boldsymbol{\beta}) = (c_{ij}(\boldsymbol{z}, \boldsymbol{\beta})) \in \mathbb{R}^{m\times r}$ and $r \leq m$ impose a low rank structure on $\mathbf{C}$. Notice that each entry c_{ij} of $\mathbf{C}$ is a function of $\boldsymbol{z}$ and $\boldsymbol{\beta}$ represents the vector of all the parameters in this map. We model c_{ij} by a spline basis $\boldsymbol{d}(\boldsymbol{z}) \in \mathbb{R}^k$, i.e. $c_{ij} = \boldsymbol{d}(\boldsymbol{z})^T\boldsymbol{\beta}_{ij}$, where $\boldsymbol{\beta}_{ij}$ is a k dimensional vector of the coefficients. In summary, for $m \times r$ matrix $\mathbf{C}$, there are $k \times m \times r$ parameters to be estimated. Then we stack all the coefficients $\boldsymbol{\beta}_{ij}$ into a large matrix $\boldsymbol{\Gamma} \in \mathbb{R}^{(mp)\times r}$

$$\boldsymbol{\Gamma} = \begin{pmatrix} \boldsymbol{\beta}_{11} & \boldsymbol{\beta}_{12} & \cdots & \boldsymbol{\beta}_{1r} \\ \boldsymbol{\beta}_{21} & \boldsymbol{\beta}_{22} & \cdots & \boldsymbol{\beta}_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\beta}_{m1} & \boldsymbol{\beta}_{m2} & \cdots & \boldsymbol{\beta}_{mr} \end{pmatrix} \tag{7}$$

where $\boldsymbol{\beta}_{ij}s$ are vectors of coefficients used to model $\boldsymbol{\Sigma}$, see equation(6). Following that, we write our model as follows

$$\begin{gathered} y_n(t, \boldsymbol{z}) = \mathbf{H}(t, \boldsymbol{z})\boldsymbol{\theta}_\mu + \boldsymbol{b}(t)^T\mathbf{I}_m \otimes \boldsymbol{d}(\boldsymbol{z})^T\boldsymbol{\Gamma}\mathbf{D}_{\boldsymbol{z}}^{-1/2}\boldsymbol{\alpha} + \epsilon(t) \\ \epsilon(t) \sim \mathcal{N}(0, \sigma_e),\ \boldsymbol{\alpha} \sim \mathcal{N}(\mathbf{0}, \mathbf{D}_{\boldsymbol{z}}). \end{gathered} \tag{8}$$

where $y_n(t, \boldsymbol{z})$ is the n-th observation at time t coupled with covariates $\boldsymbol{z}$ and $\mathbf{D}_{\boldsymbol{z}} = \mathrm{diag}(d_1, \cdots, d_r)$ is a is a diagonal matrix containing all eigenvalues. In other words, our observation $\boldsymbol{y}_n$ at time t and covariate $\boldsymbol{z}$ follows a Gaussion distribution with mean $\mathbf{H}(t, \boldsymbol{z})\boldsymbol{\theta}_\mu$ and variance $\boldsymbol{b}(t)^T\mathbf{C}(\boldsymbol{z}, \boldsymbol{\beta})\mathbf{C}(\boldsymbol{z}, \boldsymbol{\beta})^T\boldsymbol{b}(t) + \sigma_e^2$.

For $i = 1, \cdots, m_n$, if we stack all basis $\boldsymbol{b}(t_i^{(n)})$ of eigenfunctions expression into a large matrix $\mathbf{B}_n$, i.e.,

$$\mathbf{B}_n = (\boldsymbol{b}(t_1^{(n)})^T, \boldsymbol{b}(t_2^{(n)})^T, \ldots, \boldsymbol{b}(t_{m_n}^{(n)})^T)^T, \tag{9}$$

stack all basis $\mathbf{H}(t_i^{(n)}, \boldsymbol{z}_n)$ of mean function expression into a large matrix $\mathbf{H}_n$, i.e.,

$$\mathbf{H}_n = (\mathbf{H}(t_1^{(n)}, \boldsymbol{z}_n)^T, \mathbf{H}(t_2^{(n)}, \boldsymbol{z}_n)^T, \dots, \mathbf{H}(t_{m_n}^{(n)}, \boldsymbol{z}_n)^T)^T \tag{10}$$

and denote $\mathbf{C}_n = \mathbf{C}(\boldsymbol{z}_n, \boldsymbol{\beta})$, then we can write the log likelihood as

$$\mathcal{L} = \sum_{n=1}^{N} \log L(\boldsymbol{y}_n, \boldsymbol{z}_n) = \sum_{n=1}^{N} \{\log \det \boldsymbol{\Sigma}_n + \operatorname{tr}(\mathbf{S}_n \boldsymbol{\Sigma}_n^{-1})\}, \tag{11}$$

where $\boldsymbol{\Sigma}_n = \mathbf{B}_n \mathbf{C}_n \mathbf{C}_n^T \mathbf{B}_n^T + \sigma_e^2 \mathbf{I}$ with $\mathbf{C}_n \in \mathbb{R}^{K \times r}$ and $\mathbf{S}_n = (\boldsymbol{y}_n - \mathbf{H}_n \boldsymbol{\theta}_\mu)(\boldsymbol{y}_n - \mathbf{H}_n \boldsymbol{\theta}_\mu)^T$.

2.3. Roughness Penalty

In order to ensure the smoothness of the mean and eigenfunctions, we use a penalized maximum log likelihood approach to estimate the model parameters. We follow the method proposed by Wood (2006) for constructing a roughness penalty for tensor product basis. The classical roughness penalty for arbitrary function f is

$$J_t(f_t) = \int_a^b f^{''}(t)^2 dt. \tag{12}$$

If $f(t)$ has basis functions $\{\phi_k, k = 1, \cdots, K\}$ expansion, i.e.,

$$f(t) = \sum_{k=1}^{K} \widetilde{\alpha}_k \phi_k(t) = \widetilde{\boldsymbol{\alpha}}^T \boldsymbol{\phi}(t), \tag{13}$$

then classical penalty $J_t(f_t)$ can be expressed as,

$$\begin{aligned} J_t(f_t) &= \int_a^b [\frac{\partial^2}{\partial t^2} \widetilde{\boldsymbol{\alpha}}^T \boldsymbol{\phi}(t)]^2 dt \\ &= \widetilde{\boldsymbol{\alpha}}^T \int_a^b [\frac{\partial^2}{\partial t^2} \boldsymbol{\phi}(t)][\frac{\partial^2}{\partial t^2} \boldsymbol{\phi}^T(t)] dt \widetilde{\boldsymbol{\alpha}} \\ &= \widetilde{\boldsymbol{\alpha}}^T \mathbf{S}_t \widetilde{\boldsymbol{\alpha}}. \end{aligned} \tag{14}$$

The matrix $\mathbf{S}_t$ contains known coefficients and $\widetilde{\boldsymbol{\alpha}}$ is the vector of smooth coefficients. Following Wood (2006), we can write the roughness penalty for eigenfunctions of our model as,

$$J(f) = \lambda_t \sum_{j=1}^{r} \int_{\boldsymbol{z}} J_t(f_{t|\boldsymbol{z}}^{(j)}) d\boldsymbol{z} + \lambda_{\boldsymbol{z}} \sum_{j=1}^{r} \int_t J_{\boldsymbol{z}}(f_{\boldsymbol{z}|t}^{(j)}) dt, \tag{15}$$

where λ_t and $\lambda_{\boldsymbol{z}}$ are tuning parameters that need to be determined by cross-validation, $f_{t|\boldsymbol{z}}^{(j)}(t) = f_j(t, \boldsymbol{z})$ is the j-th eigenfunction of t when $\boldsymbol{z}$ is fixed and $f_{\boldsymbol{z}|t}^{(j)}(\boldsymbol{z}) = f_j(t, \boldsymbol{z})$ is the j-th eigenfunction of $\boldsymbol{z}$ when t is fixed. For function $f_{t|z}^{(j)}(t), k = 1, \cdots, r$, it can be expend by spline basis $\boldsymbol{b}(t) \in \mathbb{R}^m$ as

$$f_{t|\boldsymbol{z}}^{(j)}(t) = \sum_{i=1}^{m} \widetilde{\alpha}_i^{(j)}(\boldsymbol{z}) b_i(t) = \widetilde{\boldsymbol{\alpha}}^{(j)}(\boldsymbol{z})^T \boldsymbol{b}(t) \tag{16}$$

and for any function $\widetilde{\alpha}_i^{(j)}(\boldsymbol{z}), i = 1, \dots, m$, we can always expend it by spline basis $\boldsymbol{d}(\boldsymbol{z}) \in \mathbb{R}^k$,i.e.,

$$\widetilde{\alpha}_i^{(j)}(\boldsymbol{z}) = \sum_{l=1}^{k} \beta_{il}^{(j)} d_l(\boldsymbol{z}) = \boldsymbol{\beta}^{(j)^T} \boldsymbol{d}(\boldsymbol{z}), \tag{17}$$

Hence, we can we can always find a matrix $\mathbf{M}_{\boldsymbol{z}} = \mathbf{I}_m \otimes \boldsymbol{d}^T(\boldsymbol{z})$ such that $\widetilde{\boldsymbol{\alpha}}^{(j)}(\boldsymbol{z}) = \mathbf{M}_{\boldsymbol{z}} \boldsymbol{\beta}^{(j)}$ where $\boldsymbol{\beta}^{(j)}$ is the vector of coefficient $\beta_{il}^{(j)}$ arranged in appropriate order, i.e., $\boldsymbol{\beta}^{(j)} = (\beta_{11}^{(j)}, \beta_{12}^{(j)}, \cdots, \beta_{1k}^{(j)}, \beta_{21}^{(j)}, \cdots, \beta_{mk}^{(j)})^T$, that is the j-th colum of coefficients matrix $\boldsymbol{\Gamma}$ in (7). Hence, the penalty $J_t(f_{t|\boldsymbol{z}}^{(j)})$ can be expressed in the following form,

$$J_t(f_{t|\boldsymbol{z}}^{(j)}) = \widetilde{\boldsymbol{\alpha}}^{(j)}(\boldsymbol{z})^T \mathbf{S}_t \widetilde{\boldsymbol{\alpha}}^{(j)}(\boldsymbol{z}) = \boldsymbol{\beta}^{(j)T} \mathbf{M}_{\boldsymbol{z}}^T \mathbf{S}_t \mathbf{M}_{\boldsymbol{z}} \boldsymbol{\beta}^{(j)}. \tag{18}$$

Wood (2006) approximates $\int_{\boldsymbol{z}} J_t(f_{t|\boldsymbol{z}}^{(j)})d\boldsymbol{z}$ by,

$$\int_{\boldsymbol{z}} J_t(f_{t|\boldsymbol{z}}^{(j)})d\boldsymbol{z} = \boldsymbol{\beta}^{(j)T}\widetilde{\mathbf{S}}_t\boldsymbol{\beta}^{(j)} \tag{19}$$

where $\widetilde{\mathbf{S}}_t = \mathbf{S}_t^{'} \otimes \mathbf{I}_k$, $\mathbf{S}_t^{'} = \mathbf{B}^{-T}\mathbf{S}_t\mathbf{B}^{-1}$, $\mathbf{B} = (b_i(t_j^*)) \in \mathbb{R}^{m\times m}$ and $\{t_i^* : i = 1, \dots, m\}$ is a set of values of t spread evenly over the range of the observed t values. Similarly, $\int_t J_{\boldsymbol{z}}(f_{\boldsymbol{z}|t}^{(j)})dt$ is approximated by,

$$\int_t J_{\boldsymbol{z}}(f_{z|t}^{(j)})dt = \boldsymbol{\beta}^{(j)T}\widetilde{\mathbf{S}}_{\boldsymbol{z}}\boldsymbol{\beta}^{(j)}, \tag{20}$$

where $\widetilde{\mathbf{S}}_{\boldsymbol{z}} = \mathbf{I}_m \otimes \mathbf{S}_{\boldsymbol{z}}^{'}$, $\mathbf{S}_{\boldsymbol{z}}^{'} = \mathbf{A}^{-T}\mathbf{S}_{\boldsymbol{z}}\mathbf{A}^{-1}$, $\mathbf{A} = (d_i(\boldsymbol{z}_j^*)) \in \mathbb{R}^{k\times k}$ and $\{\boldsymbol{z}_i^* : i = 1, \dots, k\}$ is a set of values of $\boldsymbol{z}$ spread evenly over the range of the observed $\boldsymbol{z}$ values.

Therefore, the penalty for the eigenfunctions of our model can be expressed as,

$$J(f) = \sum_{j=1}^{r} \lambda_t\boldsymbol{\beta}^{(j)T}\widetilde{\mathbf{S}}_t\boldsymbol{\beta}^{(j)} + \sum_{j=1}^{r} \lambda_{\boldsymbol{z}}\boldsymbol{\beta}^{(j)T}\widetilde{\mathbf{S}}_{\boldsymbol{z}}\boldsymbol{\beta}^{(j)} = \boldsymbol{\beta}^T(\lambda_t\mathbf{I}_r \otimes \widetilde{\mathbf{S}}_t + \lambda_{\boldsymbol{z}}\mathbf{I}_r \otimes \widetilde{\mathbf{S}}_{\boldsymbol{z}})\boldsymbol{\beta}. \tag{21}$$

where $\boldsymbol{\beta} = \mathrm{vec}(\boldsymbol{\Gamma})$. Since the spline basis for the mean function is the tensor spline, we can obtain the penalty of mean function by directly applying this method, i.e.,

$$\boldsymbol{\theta}_\mu^T(\lambda_t^{(\mu)}\widetilde{\mathbf{S}}_t^{(\mu)} + \lambda_{\boldsymbol{z}}^{(\mu)}\widetilde{\mathbf{S}}_{\boldsymbol{z}}^{(\mu)})\boldsymbol{\theta}_\mu \tag{22}$$

where $\widetilde{\mathbf{S}}_t^{(\mu)} = \mathbf{S}_t^{(\mu)'} \otimes \mathbf{I}_p$, $\mathbf{S}_t^{(\mu)'} = \mathbf{E}^{-T}\mathbf{S}_t^{(\mu)}\mathbf{E}^{-1}$, $\mathbf{E} = (a_i(t_j^*)) \in \mathbb{R}^{l\times l}$ and $\{t_i^* : i = 1, \dots, l\}$ is a set of values of t spread evenly over the range of the observed t values, and $\widetilde{\mathbf{S}}_{\boldsymbol{z}}^{(\mu)} = \mathbf{I}_l \otimes \mathbf{S}_{\boldsymbol{z}}^{(\mu)'}$, $\mathbf{S}_{\boldsymbol{z}}^{(\mu)'} = \mathbf{F}^{-T}\mathbf{S}_{\boldsymbol{z}}^{(\mu)}\mathbf{F}^{-1}$, $\mathbf{F} = (c_i(\boldsymbol{z}_j^*)) \in \mathbb{R}^{p\times p}$ and $\{\boldsymbol{z}_i^* : i = 1, \dots, p\}$ is a set of values of $\boldsymbol{z}$ spread evenly over the range of the observed $\boldsymbol{z}$ values.

For tuning parameters, we apply cross-validation to determine them. Since we have two tuning parameters for the mean function and two tuning parameters for the eigenfunctions, we need to search four-dimensional space. For each tuning parameters, we search three points which means we need to search 81 combinations of tuning parameters to get a good smoothness estimation.

Combining the negative log-likelihood (equation 11) with the roughness penalty (equation 21 and 22), we obtain the objective function for our supervised FPCA models, i.e.,

$$\sum_{n=1}^{N}\{\log\det\boldsymbol{\Sigma}_n + \mathrm{tr}(\mathbf{S}_n\boldsymbol{\Sigma}_n^{-1})\} + \boldsymbol{\theta}_\mu^T(\lambda_t^{(\mu)}\widetilde{\mathbf{S}}_t^{(\mu)} + \lambda_{\boldsymbol{z}}^{(\mu)}\widetilde{\mathbf{S}}_{\boldsymbol{z}}^{(\mu)})\boldsymbol{\theta}_\mu + \boldsymbol{\beta}^T(\lambda_t\mathbf{I}_r \otimes \widetilde{\mathbf{S}}_t + \lambda_{\boldsymbol{z}}\mathbf{I}_r \otimes \widetilde{\mathbf{S}}_{\boldsymbol{z}})\boldsymbol{\beta}. \tag{23}$$

The miminizer serves as an estimate for the parameter $\boldsymbol{\theta}_\mu$ for the mean function, the parameter for the covariance function $\boldsymbol{\Gamma}$, as well as the noise variance σ_e^2. The algorithm for solving this optimization problem will be developed in the next section.

3. Algorithm

In this section, we propose an algorithm for parameter estimation of the SFPDM model. We summarize our main algorithm in section 3.1 which also contains the method of getting good initial values of our algorithm. We derive the scores prediction in section 3.2. At the end of this section, we will introduce the method to reduce the computational cost, more details can be found in the Appendix.

3.1. Model Training

Fitting our model by optimizing the objective function (23) can not be derived directly because of the nonconvex property of the objective function. We solve this problem by first obtaining a good initial value of the estimators which will be introduced later in this section. Suppose that we have already got the well-initialized estimator of the coefficient matrix $\boldsymbol{\Gamma}$ or its vector form $\boldsymbol{\beta}$, we treat this initial estimator as the start point of our algorithm. Then we modified the gradient descent framework to get the final estimates of $\boldsymbol{\theta}_\mu$, $\boldsymbol{\beta}$, and σ_e^2, which is shown in Algorithm(1).

Algorithm 1 Modified Gradient Descent for fitting SFPDM

```
1:  Set start points $\boldsymbol{\beta}^{(0)}$ by well-initialized estimators given by Algorithm(2), set start points $\boldsymbol{\theta}_\mu^{(0)}$ as a vector of all 0
    and $\sigma_e^{2(0)}$ as an appropiate small positive value.
2:  **while** $\boldsymbol{\theta}_\mu$, $\boldsymbol{\beta}$, and $\sigma_e^2$ are not converged **do**
3:
4:      **while** $\boldsymbol{\theta}_\mu$ is not converged **do**
5:          using gradient 33, update $\boldsymbol{\theta}_\mu$ by gradient descent started with $\boldsymbol{\theta}_\mu^{(0)}$;
6:          Set $\boldsymbol{\theta}_\mu^{(0)} = \boldsymbol{\theta}_\mu$
7:      **end while**
8:
9:      **while** $\boldsymbol{\beta}$ is not converged **do**
10:         using gradient 35, update $\boldsymbol{\beta}$ by gradient descent started with $\boldsymbol{\beta}^{(0)}$;
11:         Set $\boldsymbol{\beta}^{(0)} = \boldsymbol{\beta}$:
12:     **end while**
13:
14:     **while** $\sigma_e^2$ is not converged **do**
15:         using gradient 34, update $\sigma_e^2$ by gradient descent started with $\sigma_e^{2(0)}$;
16:         Set $\sigma_e^{2(0)} = \sigma_e^2$:
17:     **end while**
18:
19: **end while**
20:
21: **while** $\boldsymbol{\theta}_\mu$ is not converged **do**
22:     update $\boldsymbol{\theta}_\mu$ by gradient descent started with $\boldsymbol{\theta}_\mu^{(0)}$;
23: **end while**
```

Now we focus on deriving the explicit forms of the gradients which is used in Algorithm(1). We first note that the objective function (23) can be decomposed into two terms, the loglikelihood term $\mathcal{L}$ and penalty term $\mathcal{P}$. Notice that deriving the gradient of penalty term with respect to $\boldsymbol{\theta}_\mu$, $\boldsymbol{\beta}$ is easier. They can be written as,

$$\frac{\partial \mathcal{P}}{\partial \boldsymbol{\theta}_\mu} = \lambda_t^{(\mu)}(\widetilde{\mathbf{S}}_t^{(\mu)} + \widetilde{\mathbf{S}}_t^{(\mu)T})\boldsymbol{\theta}_\mu + \lambda_{\boldsymbol{z}}^{(\mu)}(\widetilde{\mathbf{S}}_{\boldsymbol{z}}^{(\mu)} + \widetilde{\mathbf{S}}_{\boldsymbol{z}}^{(\mu)T})\boldsymbol{\theta}_\mu \tag{24}$$

$$\frac{\partial \mathcal{P}}{\partial \boldsymbol{\beta}} = \lambda_t[(\mathbf{I}_r \otimes \widetilde{\mathbf{S}}_t) + (\mathbf{I}_r \otimes \widetilde{\mathbf{S}}_t)^T]\boldsymbol{\beta} + \lambda_{\boldsymbol{z}}[(\mathbf{I}_r \otimes \widetilde{\mathbf{S}}_{\boldsymbol{z}}) + (\mathbf{I}_r \otimes \widetilde{\mathbf{S}}_{\boldsymbol{z}})^T]\boldsymbol{\beta}. \tag{25}$$

For loglikelihood term $\mathcal{L}$, the gradient of $\boldsymbol{\theta}_\mu$ and σ_e^2 are, respectively,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_\mu} = \sum_{n=1}^{N} 2\mathbf{H}_n^T \boldsymbol{\Sigma}_n^{-1}(\mathbf{H}_n\boldsymbol{\theta}_\mu - \boldsymbol{y}_n), \tag{26}$$

$$\frac{\partial \mathcal{L}}{\partial \sigma_e^2} = \sum_{n=1}^{N} \mathrm{tr}(\boldsymbol{\Sigma}_n^{-1}) - \sum_{n=1}^{N}(\boldsymbol{y}_n - \mathbf{H}_n\boldsymbol{\theta}_\mu)^T\boldsymbol{\Sigma}_n^{-1}\boldsymbol{\Sigma}_n^{-1}(\boldsymbol{y}_n - \mathbf{H}_n\boldsymbol{\theta}_\mu) \tag{27}$$

For the k-th element of the vector $\boldsymbol{\beta}_{ij}$, the gradient for β_{ijk} can expressed in the form of the inner product,

$$\sum_{n=1}^{N} \langle \frac{\partial \mathcal{L}}{\partial \mathbf{C}_n}, \frac{\partial \mathbf{C}_n}{\partial \boldsymbol{\beta}_{ijk}} \rangle, \tag{28}$$

where the matrix $\frac{\partial \mathcal{L}}{\partial \mathbf{C}_n}$ is,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{C}_n} = 2 \times \mathbf{B}_n^T\left[\boldsymbol{\Sigma}_n^{-1} - \boldsymbol{\Sigma}_n^{-1}\mathbf{S}_n\boldsymbol{\Sigma}_n^{-1}\right]\mathbf{B}_n\mathbf{C}_n, \tag{29}$$

and $\frac{\partial \mathbf{C}_n}{\partial \boldsymbol{\beta}_{ijk}} = d_j(\boldsymbol{z})$.

Notice that the dimension of $\mathbf{\Sigma}_n$ is $m_n \times m_n$, and this will lead to $\mathcal{O}(m_n^3)$ computational cost when evaluating $\mathbf{\Sigma}^{-1}$. To reduce the computational cost, we focus on simplified log-likelihood and gradient of each parameter by applying the determinant and Sherman-Morrison-Woodbury lemmas, i.e.,

$$\det(\mathbf{A} + \mathbf{U}\mathbf{V}^T) = \det(\mathbf{I} + \mathbf{V}^T\mathbf{A}^{-1}\mathbf{U})\det(\mathbf{A}) \tag{30}$$

and

$$(\mathbf{A} + \mathbf{U}\mathbf{V}^T)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{I} + \mathbf{V}^T\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}^T\mathbf{A}^{-1}, \tag{31}$$

respectively. The next four lemmas reduce computational cost from $\mathcal{O}(m_n^3)$ to $\mathcal{O}(r^3)$ when calculating the log-likelihood and the gradient of $\boldsymbol{\theta}_\mu$, σ_e^2 and $\boldsymbol{\beta}$, see detailed proof of them in Appendix.

Lemma 1. *The log-likelihood $\mathcal{L}$ given in (11) can be expressed as,*

$$\mathcal{L} \propto 2\sum_{n=1}^N \log\det(\mathbf{F}_n) - (\sigma_e^{-2})^2 \sum_{n=1}^N \|\boldsymbol{h}_n\|_2^2 + \sigma_e^{-2}\sum_{n=1}^N \|\boldsymbol{y}_n\|_2^2. \tag{32}$$

where $\mathbf{W}_n = \mathbf{C}_n^T\mathbf{B}_n^T\mathbf{B}_n\mathbf{C}_n$, $\boldsymbol{g}_n = \mathbf{C}_n^T\mathbf{B}_n^T\boldsymbol{y}_n$, $\boldsymbol{h}_n = \mathbf{F}_n^{-1}\boldsymbol{g}_n$ and $\mathbf{F}_n$ is the Cholesky decomposition of $\mathbf{I}_r + \sigma_e^{-2}\mathbf{W}_n$, i.e., $\mathbf{F}_n\mathbf{F}_n^T = \mathbf{I}_r + \sigma_e^{-2}\mathbf{W}_n$.

Lemma 2. *The gradient $\frac{\partial\mathcal{L}}{\partial\boldsymbol{\theta}_\mu}$ given in (26) can be expressed as,*

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{\theta}_{\boldsymbol{\mu}}} = \sum_{n=1}^N -2\sigma_e^{-2}(\mathbf{H}_n^T\boldsymbol{y}_n - \mathbf{H}_n^T\mathbf{H}_n\boldsymbol{\theta}_{\boldsymbol{\mu}}) + 2\sigma_e^{-2}\mathbf{H}_n^T\mathbf{B}_n\mathbf{E}_n^T\mathbf{E}_n(\mathbf{B}_n^T\boldsymbol{y}_n - \mathbf{B}_n^T\mathbf{H}_n\boldsymbol{\theta}_{\boldsymbol{\mu}}), \tag{33}$$

where $\mathbf{E}_n = \mathbf{L}_n^{-1}\mathbf{C}_n^T$ and $\mathbf{L}_n$ is the Cholesky decomposition of $\sigma_e^2\mathbf{I}_r + \mathbf{C}_n^T\mathbf{B}_n^T\mathbf{B}_n\mathbf{C}_n$, i.e. $\mathbf{L}_n\mathbf{L}_n^T = \sigma_e^2\mathbf{I}_r + \mathbf{C}_n^T\mathbf{B}_n^T\mathbf{B}_n\mathbf{C}_n$.

Lemma 3. *The gradient $\frac{\partial\mathcal{L}}{\partial\sigma_e^2}$ given in (27) can be expressed as,*

$$\frac{\partial\mathcal{L}}{\partial\sigma_e^2} = \sum_{n=1}^N \left[\sigma_e^{-2}m_n - \sigma_e^{-2}\mathrm{tr}(\mathbf{E}_n^T\mathbf{E}_n\mathbf{B}_n^T\mathbf{B}_n) - \sigma_e^{-2}\boldsymbol{y}_n^T\boldsymbol{y}_n + \sigma_e^{-2}\boldsymbol{y}_n^T\mathbf{B}_n\mathbf{E}_n^T\mathbf{E}_n\mathbf{B}_n^T\boldsymbol{y}_n\right], \tag{34}$$

where $\mathbf{E}_n = \mathbf{L}_n^{-1}\mathbf{C}_n^T$ and $\mathbf{L}_n$ is the Cholesky decomposition of $\sigma_e^2\mathbf{I}_r + \mathbf{C}_n^T\mathbf{B}_n^T\mathbf{B}_n\mathbf{C}_n$, i.e. $\mathbf{L}_n\mathbf{L}_n^T = \sigma_e^2\mathbf{I}_r + \mathbf{C}_n^T\mathbf{B}_n^T\mathbf{B}_n\mathbf{C}_n$.

Lemma 4. *For the k-th element of the vector $\boldsymbol{\beta}_{ij}$, the gradient for β_{ijk} can expressed in the form of the inner product,*

$$\sum_{n=1}^N \langle \frac{\partial L}{\partial\mathbf{C}_n}, \frac{\partial\mathbf{C}_n}{\partial\boldsymbol{\beta}_{ijk}} \rangle, \tag{35}$$

$\frac{\partial\mathbf{C}_n}{\partial\boldsymbol{\beta}_{ijk}} = d_j(\boldsymbol{z})$ and $\frac{\partial\mathcal{L}}{\partial\mathbf{C}_n}$ can be expressed as,

$$\frac{\partial\mathcal{L}}{\partial\mathbf{C}_n} = 2\sigma_e^{-2}(\mathbf{B}_n^T\mathbf{B}_n\mathbf{C}_n - \mathbf{B}_n^T\mathbf{K}_n\mathbf{W}_n) - 2(\sigma_e^2)^{-2}(\mathbf{B}_n^T - \mathbf{B}_n^T\mathbf{K}_n\mathbf{C}_n^T\mathbf{B}_n^T)\mathbf{S}_n(\mathbf{B}_n\mathbf{C}_n - \mathbf{K}_n\mathbf{W}_n), \tag{36}$$

where $\mathbf{W}_n = \mathbf{C}_n^T\mathbf{B}_n^T\mathbf{B}_n\mathbf{C}_n$ and $\mathbf{K}_n = \mathbf{B}_n\mathbf{C}_n\{\sigma_e^2\mathbf{I}_r + \mathbf{W}_n\}^{-1}$.

So far we have got the procedure of fitting our SFPDM model and derive the detailed simplified gradient of each parameter. However, recall that the objective function (23)is a non-convex problem, we cannot entirely rely on gradient descent, and instead, use a direct approach to find a good initialization. Now we introduce the procedure on how to get a good estimator of the coefficient matrix $\mathbf{\Gamma}$ or its vector form $\boldsymbol{\beta}$. Specifically, we first divide the covariate domain into small bins and apply classical FPCA in each bin to get a rough estimator of $\mathbf{I}_m \otimes \boldsymbol{d}(\boldsymbol{z})^T\mathbf{\Gamma}\mathbf{D}_{\boldsymbol{z}}^{-1/2}$ and $\mathbf{D}_{\boldsymbol{z}}^{-1/2}$ in equation (8). We then treat these estimates $\mathbf{I}_m \otimes \boldsymbol{d}(\boldsymbol{z})^T\mathbf{\Gamma}\mathbf{D}_{\boldsymbol{z}}^{-1/2}$ as the response in a linear regression with $\mathbf{I}_m \otimes \boldsymbol{d}(\boldsymbol{z})^T$ as predictors to get an initial estimation of the coefficient matrix $\mathbf{\Gamma}$. The algorithm is given in Algorithm 2.

Algorithm 2 Initial Estimate of coefficient matrix $\mathbf{\Gamma}$

```
1: Divide the covariates domain into $S$ bins.
2: Applying the classical FPCA to get an estimate $\mathbf{I}_m \otimes \boldsymbol{d}(\boldsymbol{z})^T \mathbf{\Gamma} \mathbf{D}_{\boldsymbol{z}}^{-1/2}$ and $\mathbf{D}_{\boldsymbol{z}}^{-1/2}$.
3: Treating the rough estimate $\mathbf{I}_m \otimes \boldsymbol{d}(\boldsymbol{z})^T \mathbf{\Gamma} \mathbf{D}_{\boldsymbol{z}}^{-1/2}$ as the response and $\mathbf{I}_m \otimes \boldsymbol{d}(\boldsymbol{z})^T$ as the predictor, use least squares
   to get matrix $\mathbf{\Gamma} \mathbf{D}_{\boldsymbol{z}}^{-1/2}$.
4: Scaled by $\mathbf{D}_{\boldsymbol{z}}^{1/2}$ to get an initial value of coefficient matrix $\mathbf{\Gamma}$.
```

3.2. Scores Prediction

The posterior distribution of $\boldsymbol{\alpha}^{(n)}$ is a normal distribution with mean $\mathbf{E}(\boldsymbol{\alpha}^{(n)}|\boldsymbol{y}_n)$ and $\mathbf{cov}(\boldsymbol{\alpha}^{(n)}|\boldsymbol{y}_n)$, where

$$\mathbf{E}(\boldsymbol{\alpha}^{(n)}|\boldsymbol{y}_n) = \mathbf{D}_{\boldsymbol{z}} \mathbf{\Theta}_{\boldsymbol{z}}^T \mathbf{B}_n^T (\mathbf{B}_n \mathbf{\Theta}_{\boldsymbol{z}} \mathbf{D}_{\boldsymbol{z}} \mathbf{\Theta}_{\boldsymbol{z}}^T \mathbf{B}_n^T + \sigma_e^2 \mathbf{I})^{-1} (\boldsymbol{y}_n - \mathbf{H}_n \boldsymbol{\theta}_\mu), \tag{37}$$

$$\mathbf{cov}(\boldsymbol{\alpha}^{(n)}|\boldsymbol{y}_n) = \mathbf{D}_{\boldsymbol{z}} - \mathbf{D}_{\boldsymbol{z}} \mathbf{\Theta}_{\boldsymbol{z}}^T \mathbf{B}_n^T (\mathbf{B}_n \mathbf{\Theta}_{\boldsymbol{z}} \mathbf{D}_{\boldsymbol{z}} \mathbf{\Theta}_{\boldsymbol{z}}^T \mathbf{B}_n^T + \sigma_e^2 \mathbf{I})^{-1} \mathbf{B}_n \mathbf{\Theta}_{\boldsymbol{z}} \mathbf{D}_{\boldsymbol{z}}, \tag{38}$$

and,

$$\mathbf{\Theta}_z = \mathbf{C}_{\boldsymbol{z}}\ \mathbf{D}_{\boldsymbol{z}}^{-1/2} = \mathbf{I}_m \otimes \boldsymbol{d}(\boldsymbol{z})^T \mathbf{\Gamma} \mathbf{D}_{\boldsymbol{z}}^{-1/2}. \tag{39}$$

Suppose we want to predict function value $y_n(t)$ at time t^*, we can make use of $\mathbf{E}(\boldsymbol{\alpha}^{(n)}|\boldsymbol{y}_n)$ and $\mathbf{cov}(\boldsymbol{\alpha}^{(n)}|\boldsymbol{y}_n)$ to get the estimation of prediction as well as its variance, that are,

$$\mathbf{H}(t^*, \boldsymbol{z}_n) \boldsymbol{\theta}_\mu + \boldsymbol{b}(t^*)^T \mathbf{\Theta}_{\boldsymbol{z}} \mathbf{E}(\boldsymbol{\alpha}^{(n)}|\boldsymbol{y}_n), \tag{40}$$

and

$$\boldsymbol{b}(t^*)^T \mathbf{\Theta}_{\boldsymbol{z}} \mathbf{cov}(\boldsymbol{\alpha}^{(n)}|\boldsymbol{y}_n) \mathbf{\Theta}_{\boldsymbol{z}}^T \boldsymbol{b}(t^*) + \sigma_e^2. \tag{41}$$

If we are interested in the underlying latent function value $x_n(t^*)$, then the prediction will be the same with $y_n(t^*)$ and the corresponding variance will be,

$$\boldsymbol{b}(t^*)^T \mathbf{\Theta}_{\boldsymbol{z}} \mathbf{cov}(\boldsymbol{\alpha}^{(n)}|\boldsymbol{y}_n) \mathbf{\Theta}_{\boldsymbol{z}}^T \boldsymbol{b}(t^*). \tag{42}$$

Sometimes, such as in astronomy, measurement errors are provided with each observation $y_n(t)$. In this case, we modify our model by replacing σ_e^2 above with $\sigma_j^{(n)}$, where $\sigma_j^{(n)}$ is the measurement error at time $t_j^{(n)}$, $j = 1, 2, \ldots, m_n$.

4. Simulation

We now compare our method to one of state-of-the-art supervised FPCA model called supervised sparse and functional principal component (SupSFPC) method proposed by Li, Shen and Huang (2016). SupSFPC is based on the assumption that the scores have a linear relationship with the covariates, which is often violated in practice. In addition, compared with our SFPDM model, SupSFPC does not handle irregularly spaced data and can not incorporate measurement error information.

4.1. Simulation design

We simulate a dataset of noisy realizations of $n = 7500$ latent functions. The n-th latent function $x_n(t, z)$ is a linear combination of a mean function $\mu(t, z)$ and $r = 3$ orthonormal eigenfunctions $f_j(t, z)$, $j = 1, \ldots, r$. Here we set covariate z be univariate. We set the mean function to be

$$30(t - z)^2 \tag{43}$$

and the three eigenfunctions to be

$$f_1(t, z) = \sqrt{2}\{\cos(\pi(t + z))\}, \tag{44}$$

$$f_2(t, z) = \sqrt{2}\{\sin(\pi(t + y))\}, \tag{45}$$

$$f_3(t, z) = \sqrt{2}\{\cos(3\pi(t - y))\}. \tag{46}$$

TABLE 1
Comparison of the FVE for the SupSFPC and SFPDM methods.

	SupSFPC	SFPDM
r	FVE ($\times 10^{-1}$)	FVE($\times 10^{-1}$)
r = 1	5.9987	7.8485
r = 2	9.7460	9.8380
r = 3	9.8792	9.9837

To further impose dependence of the covariance structure on the covariate z we set the eigenvalues to be

$$\big(2(z+20), z+10, z\big). \tag{47}$$

The scores $\boldsymbol{\alpha}^{(n)}$ are sampled from a Normal distribution with mean $\mathbf{0}$ and covariance matrix

$$\begin{pmatrix} 2(z+2) & 0 & 0 \\ 0 & z+10 & 0 \\ 0 & 0 & z \end{pmatrix}. \tag{48}$$

The number of the spline basis functions used to capture the dependence of the mean function on t and z are set to 10 and 5, respectively, i.e. $l = 10$ and $p = 5$ (see (3)). The number used to capture the dependence of the eigenfunctions on t and z are set to 10 and 7, respectively (see 8). All the spline bases are cubic splines.

The final dataset contains m noisy realizations of each function x_n, i.e.,

$$y_n(t_i, \boldsymbol{z}) = x_n(t_i, \boldsymbol{z}) + \epsilon(t_i) = \mu(t_i, \boldsymbol{z}) + \sum_{j=1}^{r} \alpha_j^{(n)} f_j(t_i, \boldsymbol{z}) + \epsilon(t_i) = \mu(t_i, \boldsymbol{z}) + \boldsymbol{f}^T(t_i, \boldsymbol{z})\boldsymbol{\alpha}^{(n)} + \epsilon(t_i), \tag{49}$$

for $i = 1, \dots, m$, where ϵ is an independent white noise process with variance $\sigma_e^2 = 0.1$. We repeat the simulation 500 times to get the standard errors associated with the estimated of the mean function and eigenfunctions. We also obtain the 95% confidence interval of each observation prediction by making use of the prediction variance given by (41).

4.2. Results

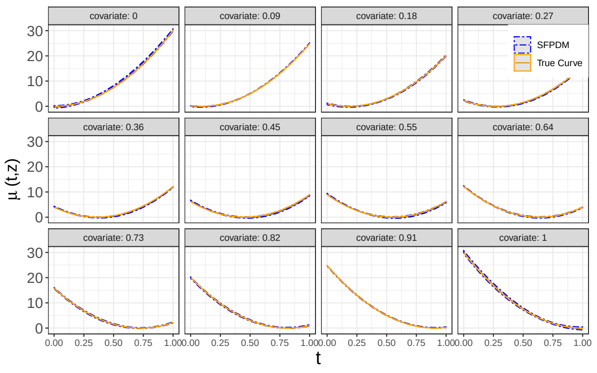
Because SupSFPC does not give the estimation of mean function, we only summarizes the mean function estimation performance of our method in Figure 1(a). Figure(1(b),1(c),1(d)) compares the estimation performance of our method and SupSFPC for eigenfunctions f_1, f_2, and f_3, respectively. For all three eigenfunctions, our method performs better in terms of estimation accuracy. Indeed, our method recovers the true mean function and three eigenfunctions almost exactly. Error bars of two times standard error are plotted for all both methods but cannot be seen because in all cases the standard error are very small.

Next, to assess the prediction accuracy of both methods. We generate $n = 7500$ observations of functions as our training set and another 7500 observation as our test set. Then, we fit the mean function and eigenfunctions using the training data. Next, for each test function, we estimate the scores by their posterior mean (see (37)) computed based on a random selection of 20% of the noisy realizations generated for that function. Finally, we make predictions for the remaining 80% of the observations by making use of (40). Some example predictions are shown in Figure 2. Compared with SupSFPC method, the coverage of our method is much closer to the nominal coverage and each time points are all fall within the 95% confidence interval of our prediction.

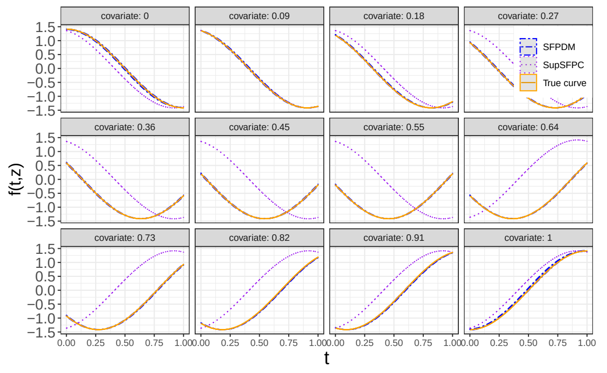
Table 1 gives fraction of variation explained (FVE) for the two models on the test data.

$$\frac{\sum_{n=1}^{N}\sum_{i=1}^{m_n}\{y_n(t_i) - \hat{\mu}^{(n)}(t_i) - \sum_{j=1}^{r}\hat{\alpha}_j^{(n)}\hat{f}_j^{(n)}(t_i)\}^2}{\sum_{n=1}^{N}\sum_{i=1}^{m_n}\{y_n(t_i) - \widetilde{\mu}(t_i)\}^2}, \tag{50}$$

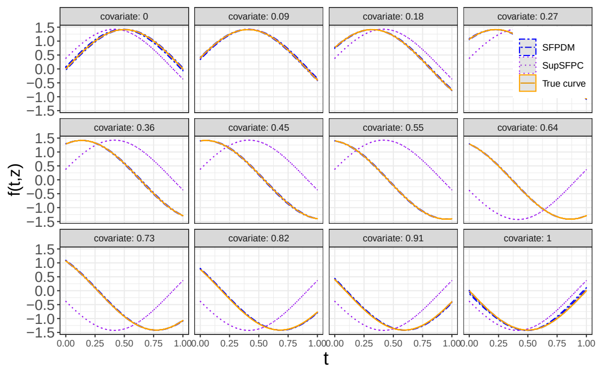
where $\hat{\alpha}$ is the estimator of the scores, and $\widetilde{\mu}(t_i) = \frac{1}{N}\sum_{n=1}^{N} y(t_i)$, for $i = 1, \dots, m$.. $\hat{\mu}$ and $\hat{f}$ are estimators of mean function and eigenfunctions. For our model, $\hat{\mu}^{(n)}(t_i^{(n)}) = \hat{\mu}(t^{(n)}, \boldsymbol{z}_n)$ and $\hat{f}^{(n)}(t_i^{(n)}) = \hat{f}(t_i^{(n)}, \boldsymbol{z}_n)$. For SupSFPC, $\hat{\mu}^{(n)}(t_i^{(n)}) = \hat{\mu}(t^{(n)})$ and $\hat{f}^{(n)}(t_i^{(n)}) = \hat{f}(t_i^{(n)})$. Our model has a higher FVE for all three values of r.



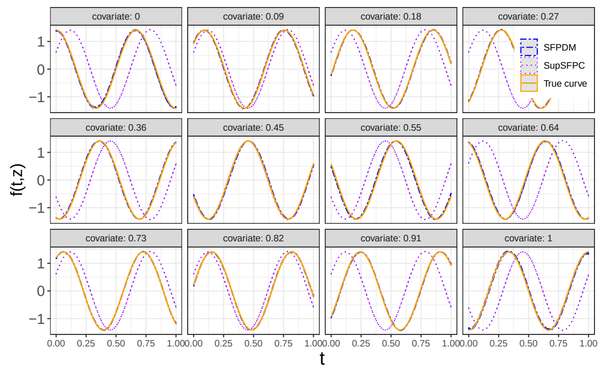
(a) mean function



(b) first eigenfunction

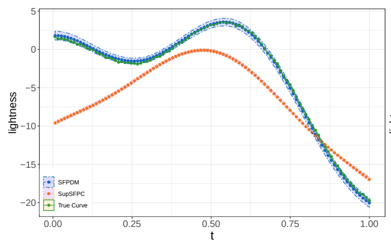


(c) second eigenfunction

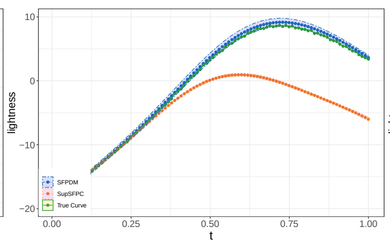


(d) third eigenfunction

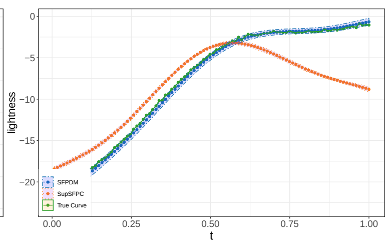
FIG 1. *Panel (a) compares the SFPDM (dashed blue lines) and SupSFPC (dotted purple lines) estimates of the mean function, for a range of covariate values. Panels (b), (c), and (d) compare the SFPDM and supSFPC estimates of eigenfunctions 1, 2, and 3, respectively, for a range of covariate values. The true functions are shown as orange solid lines.*



(a) covariate: 0.2



(b) covariate: 0.5



(c) covariate: 0.7

FIG 2. *Panels (a), (b) and (c) show example curve predictions and 95% prediction intervals for the SFPDM (blue dash-dot lines) and SupSFPC (dotted lines) methods. The solid green line shows the true curve. The covariates for the curves are given beneath the panels.*

5. Modeling Astronomical Lightcurves

In astronomy, a lightcurve is a time series of the observed brightness of a source, e.g., a star or galaxy. Some astronomical sources vary in brightness over time, and these variations can be used to classify the type of source or infer its properties, e.g., the period of star pulsations (from which additional physical insights can be gained). One type of variable source is an eclipsing binary system, which is a system of two stars orbiting each other. Many stars visible to the eye are in fact eclipsing binary systems. If the orbits of the two stars lie in the plane that also contains our line of sight then, viewed from Earth, the stars will alternately eclipse each other. The stars cannot usually be resolved, but the periodic eclipses block some of the light from reaching us and create periodic dips in the observed lightcurve. These characteristics can be used to distinguish eclipsing binary sources from other variable sources and help us to infer properties of the two stars, e.g., their relative masses.

Our data set consists of $n = 35615$ eclipsing binary lightcurves from the Catalina Real-Time Transient Survey (CRTS) (Drake et al., 2009) which were classified by the CRTS team in Drake et al. (2014). Each observed magnitude (brightness) measurement is accompanied by a known measurement error (i.e., standard deviation), that is determined by astronomers based on the properties of the telescope. The data are publicly available from `http://crts.caltech.edu/`. Figure 3(a) shows standardized versions of 10 eclipsing binary lightcurves. For visual purposes the measurement errors are not plotted. The mean value of it is near 0.15. The y-axis is standardized magnitude, where magnitude is an astronomical measure of the intensity of light from a star, with negative numbers indicating greater intensity. Standardizing so that all the measurements fall in $[0.5, -0.5]$ is necessary here because we are principally interested in modeling the shape of the lightcurves and the observed magnitude depends on the distance of the source from us. In Figure 3(a), we set the x-axis to be phase of oscillation, which is the conventional approach because eclipsing binary lightcurves are periodic. The periods of oscillations were found by Drake et al. (2014) and are treated as known for the purposes of our analysis. In practice, the periods would have to be estimated which is itself a challenging inference problem, but the improved modeling we present here will facilitate improved period estimation accuracy in future.

An important feature of Figure 3(a) is that for some lightcurves the depth of the two eclipses are similar, and for others they are very different. This distinction is due to there being different types of eclipsing binary system. Eclipsing binaries are often divided into two classes, contact binaries which are sufficiently close to exchange mass, and detached binaries which are more separated. Contact binaries typically consist of two sources with similar properties (e.g., size and brightness), meaning that the two eclipses are similar. In contrast, detached binaries may have eclipses of any relative size, because the two sources can have completely different properties.

The above considerations raise an important modeling challenge: eclipsing binary lightcurves can be modeled using somewhat similar functions, because they have similar shapes and covariance structure, but it does not make sense to treat them as coming from a completely homogeneous distribution, as is typically assumed in FPCA. The current solution is to divide eclipsing binaries into contact and detached binaries and treat these groups as homogeneous, but this is still unsatisfactory because the detached binaries group is heterogeneous. Treating eclipsing binaries as homogeneous, means that any models we use to fit them will either be inaccurate or unnecessarily complicated, which in turn will reduce our ability to classify them, estimate their periods, and learn their other properties. Instead, we use our SFPDM method to learn a mean function and a set of covariance matrix eignefunctions that smoothly vary with the relative depth of the two eclipses. This approach captures the fact that eclipsing binary lightcurves are similar, while also accounting for a physically interpretable difference.

To implement our approach we need a parameter or covariate related to the relative depth of the two eclipses of each lightcurve. In practice, such information may sometimes be available from a separate observation of the eclipsing binary system. However, in many cases a parameter capturing the relative depth would need to be inferred from the data. For the sake of simplicity, in this work we calculate an approximation of the relative depth of the eclipses of each lightcurve from the data and treat it as a covariate.

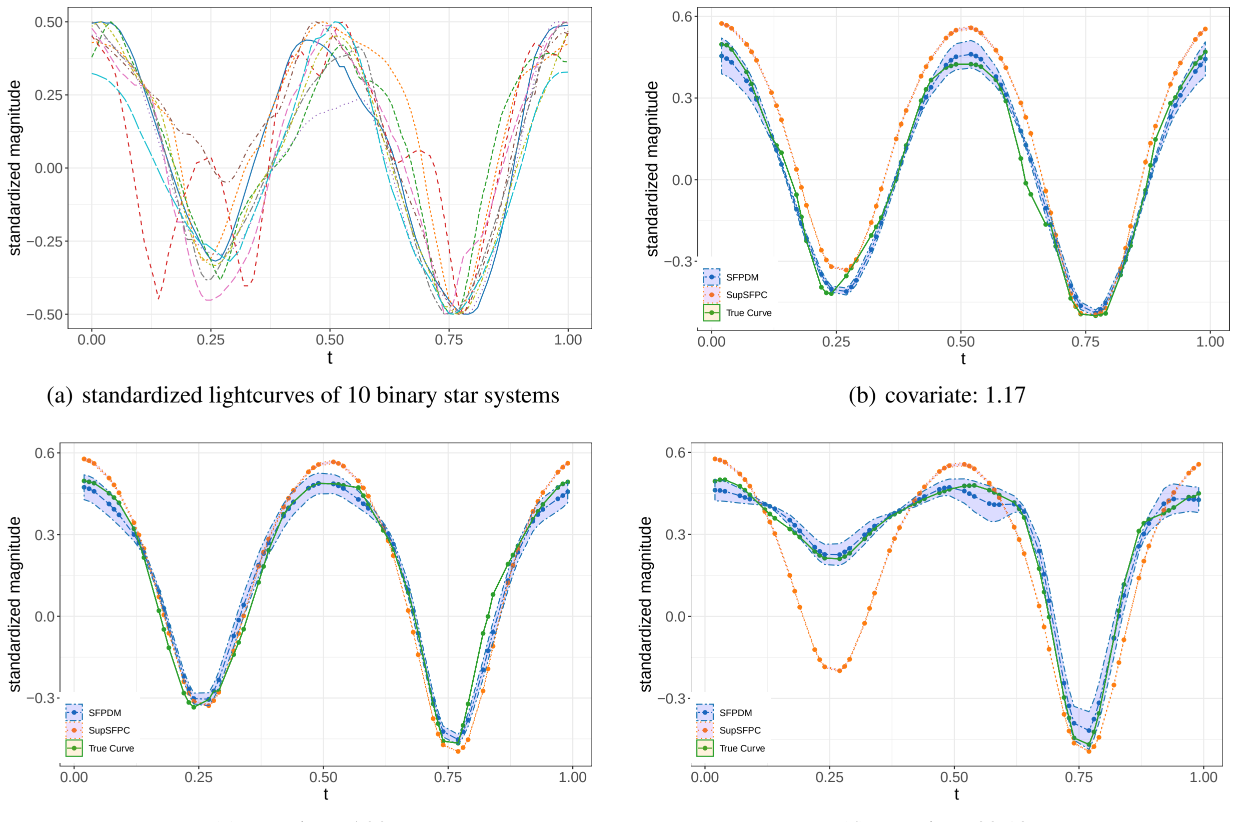
The SupSFPC method only handles regularly spaced data and does make use of measurement errors. Therefore, for the purpose of comparing our approach to the SupSFPC method, we first consider a processed version of the data where all the observations lie on a regular phase grid. In particular, we use a B-splines fit to each lightcurve to obtain 101 observations at regularly spaced phases, regardless of the number of observations in the original lightcurve. For this grid data, we do not have measurement errors. After comparing the methods on the grid data, we will also apply our model to the raw data (including the measurement errors) to further demonstrate its performance.

We randomly divided the grid dataset into a training set and a test set, composed of 70% and 30% of the total number of ligthcurves, respectively. Table 2 shows the training and test prediction mean square error (MSE) for both SupSFPC and our method. The predictions are computed in the same way as in Section 4.2, except a random 25% of the observations in each lightcurve are used to estimate the scores, and predictions are made for the other 75% (as opposed to 20% for estimation and 80% for prediction). The rows of Table 2 correspond to different values of r, the rank of the matrix $\boldsymbol{C}$ used in approximating the covariance matrix $\boldsymbol{\Sigma}$, see (6). Table 2 shows that the training set prediction MSE is similar for both methods when $r = 2$, which suggests that the degrees of freedom of the two models are similar in this case. However, the test set prediction MSE is much lower for our SFPDM method when $r = 2$, and in fact for all three choices of r.

Figure (3(b), 3(c),3(d)) shows 95% prediction intervals for SupSFPCA (orange) and our SFPDM method (blue), for three example lightcurves in the test set. The true lightcurve is also shown (green). Our method well captures the way the lightcurve shapes vary with the covariate, and also provides reasonable prediction intervals. In contrast, the

TABLE 2
Comparison of prediction MSE for the grid data.

	SupSFPC ($\times 10^{-2}$)		SFPDM($\times 10^{-2}$)	
r	Training Set	Test Set	Training Set	Test Set
r = 1	1.6337	3.2535	1.9334	2.0039
r = 2	1.5335	3.2388	1.5671	1.7304
r = 3	1.4948	3.1842	0.5185	0.8649



(a) standardized lightcurves of 10 binary star systems

(b) covariate: 1.17

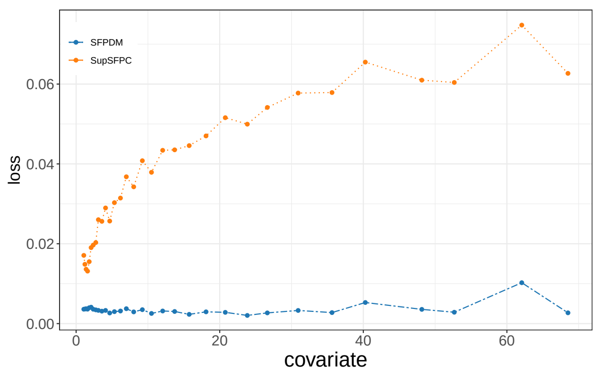
(c) covariate: 5.90

(d) covariate: 32.13

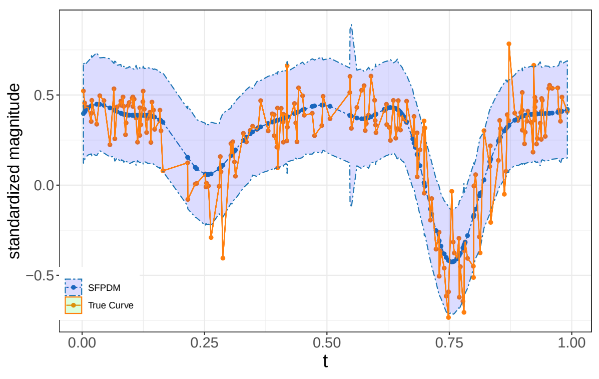
FIG 3. *Panel (a) shows standardized lightcurves of 10 eclipsing binary sources. Panels (b), (c), and (d) show example lightcurve predictions and 95% prediction intervals for the SFPDM (blue dash-dot lines) and SupSFPC (dotted lines) methods. The green solid line shows the true lightcurve. The covariates values of the three lightcurves are given beneath the panels.*

SupSFPC method does not capture the lightcurve shapes well, because its assumption that the scores vary linearly with the covariate is not valid. Furthermore, the 95% prediction intervals are clearly inadequate. To further demonstrate this, we define loss to be the summation of the prediction squared errors, and plot the loss for each method in Figure 4(a). SFPDM has similar loss for all values of the covariate, but SupSFPC has much higher loss for larger values of the covariate. In particular, since the SupSFPC method cannot properly capture the way the lightcurves change with the covariate, it focuses on fitting lightcurves with low covariate values, which constitute the majority of the data.

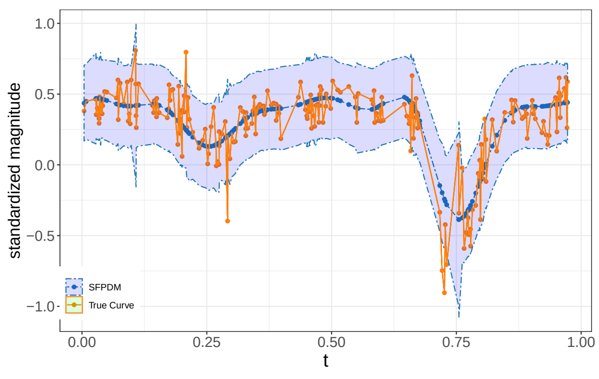
Next we apply SFPDM to the raw data, which cannot be analyzed by the SupSFP method. In this case the training and test set prediction MSE are almost identical (0.0087), indicating that the model well captures the data. Figure (4(b), 4(c), 4(d)) shows 95% prediction intervals for three example lightcurves in the test set. In this case 95% prediction interval is actually more consistent with the data, than in the grid data case. This is because, in the raw data, the individual observation measurement errors are available both for fitting the model and for making predictions.



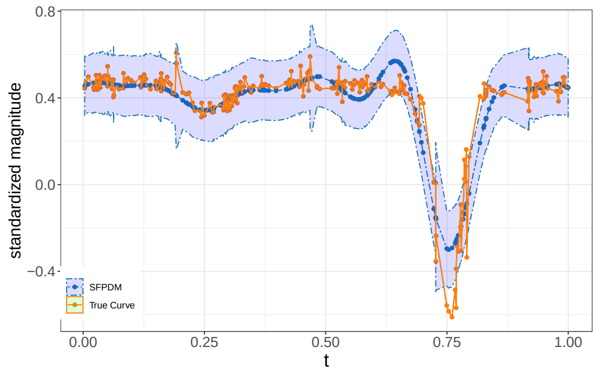
(a) loss



(b) covariate: 4.27



(c) covariate: 5.45



(d) covariate: 27.34

FIG 4. *Panel (a) is the comparison of loss (i.e., the prediction MSE) for the SFPDM (blue) and SupSFPC (orange) methods. Panels (b), (c) and (d) are SFPDM predictions and 95% prediction intervals (blue) for three example raw data lightcurves in the test set. The orange solid line shows the raw data. The covariate value for each lightcurve is given beneath the panels.*

6. Discussion

In both our simulation studies and real data analysis, our method performs better than SupSFPC in all aspects, e.g., mean and eigenfunction estimation, and prediction accuracy. Furthermore, our method can handle irregular observations times and incorporate measurement errors, whereas SupSFPC does not. By imposing a low rank structure on matrix $\mathbf{C}$, and applying the determinant and Sherman-Morrison-Woodbury lemmas we reduce the the computation cost of our approach from $\mathcal{O}(m_n^3)$ to $\mathcal{O}(r^3)$. Thus, our method has substantionally lower computational cost than local smoother based covariate adjustment models (Jiang et al. (2010), Jiang et al. (2011), Zhang et al. (2016), and Zhang, Park and Wang (2013)). In future work we will explore ways to construct a map from Euclidean space to the Stiefel manifold, because the coefficient matrix $\mathbf{\Theta}_z$ lies on the Stiefel manifold. The Stiefel manifold has more structure than the positive definite matrix manifold and optimization methods on the Stiefel manifold have been well developed.

7. Appendix

proof of lemma 1. Specifically, we set $\mathbf{A} = \sigma_e^2\mathbf{I}$, $\mathbf{U} = \mathbf{B}_n\mathbf{C}_n$ and $\mathbf{V} = \mathbf{B}_n\mathbf{C}_n$ so that the determinant and inverse of

$$\mathbf{\Sigma}_n = \mathbf{B}_n\mathbf{C}_n\mathbf{C}_n^T\mathbf{B}_n^T + \sigma_e^2\mathbf{I} \tag{51}$$

are given by

$$|\mathbf{\Sigma}_n| = \det(\mathbf{I}_r + \sigma_e^{-2}\mathbf{C}_n^T\mathbf{B}_n^T\mathbf{B}_n\mathbf{C}_n) + m_n \log \sigma_e^2, \tag{52}$$

and

$$\mathbf{\Sigma}_n^{-1} = \sigma_e^{-2}(\mathbf{I}_{m_n} - \mathbf{B}_n\mathbf{C}_n\{\sigma_e^2\mathbf{I}_r + \mathbf{W}_n\}^{-1}\mathbf{C}_n^T\mathbf{B}_n^T), \tag{53}$$

respectively. Similarly, using the same two lemmas, the log likelihood in (11) can be simplified to

$$\mathcal{L} \propto \sum_{n=1}^{N} \log\det(\mathbf{I}_r + \sigma_e^{-2}\mathbf{W}_n) + \operatorname{tr}(\sigma_e^{-2}\mathbf{S}_n[\mathbf{I}_{m_n} - \mathbf{B}_n\mathbf{C}_n\{\sigma_e^2\mathbf{I}_r + \mathbf{W}_n\}^{-1}\mathbf{C}_n^T\mathbf{B}_n^T]), \tag{54}$$

where $\mathbf{W}_n = \mathbf{C}_n^T\mathbf{B}_n^T\mathbf{B}_n\mathbf{C}_n$.

The simplified progress above is,

$$\begin{aligned}
\mathcal{L} &= \sum_{n=1}^{N} \log\det(\boldsymbol{l}_r + \sigma_e^{-2}\mathbf{C}_n^T\mathbf{B}_n^T\mathbf{B}_n\mathbf{C}_n) + m_n\log\sigma_e^2 + \operatorname{tr}(\sigma_e^{-2}\mathbf{S}_n[\mathbf{I}_{m_n} - \mathbf{B}_n\mathbf{C}_n\{\sigma_e^2\mathbf{I}_r + \mathbf{C}_n^T\mathbf{B}_n^T\mathbf{B}_n\mathbf{C}_n\}^{-1}\mathbf{C}_n^T\mathbf{B}_n^T]) \\
&\propto \sum_{n=1}^{N} \log\det(\mathbf{I}_r + \sigma_e^{-2}\mathbf{C}_n^T\mathbf{B}_n^T\mathbf{B}_n\mathbf{C}_n) + \operatorname{tr}(\sigma_e^{-2}\mathbf{S}_n[\mathbf{I}_{m_n} - \mathbf{B}_n\mathbf{C}_n\{\sigma_e^2\mathbf{I}_r + \mathbf{C}_n^T\mathbf{B}_n^T\mathbf{B}_n\mathbf{C}_n\}^{-1}\mathbf{C}_n^T\mathbf{B}_n^T]) \\
&\propto \sum_{n=1}^{N} \log\det(\mathbf{I}_r + \sigma_e^{-2}\mathbf{W}_n) + \operatorname{tr}(\sigma_e^{-2}\mathbf{S}_n[\mathbf{I}_{m_n} - \mathbf{B}_n\mathbf{C}_n\{\sigma_e^2\mathbf{I}_r + \mathbf{W}_n\}^{-1}\mathbf{C}_n^T\mathbf{B}_n^T])\,,
\end{aligned} \tag{55}$$

where $\mathbf{W}_n = \mathbf{C}_n^T\mathbf{B}_n^T\mathbf{B}_n\mathbf{C}_n$.

To further reduce computational cost, we use a Cholesky decomposition when evaluating the log likelihood. Firstly, note that

$$\mathcal{L} \propto \sum_{n=1}^{N} \log\det(\mathbf{I}_r + \sigma_e^{-2}\mathbf{W}_n) + \operatorname{tr}(\sigma_e^{-2}\mathbf{S}_n) - \operatorname{tr}(\sigma_e^{-2}\mathbf{S}_n\mathbf{B}_n\mathbf{C}_n\{\sigma_e^2\mathbf{I}_r + \mathbf{W}_n\}^{-1}\mathbf{C}_n^T\mathbf{B}_n^T]) \tag{56}$$

$$\propto \sum_{n=1}^{N} \left[\log\det(\mathbf{I}_r + \sigma_e^{-2}\mathbf{W}_n) - (\sigma_e^{-2})^2\operatorname{tr}(\boldsymbol{g}_n^T\{\mathbf{I}_r + \sigma_e^{-2}\mathbf{W}_n\}^{-1}\boldsymbol{g}_n])\right] + \sigma_e^{-2}\sum_{n=1}^{N}\|\boldsymbol{y}_n\|_2^2, \tag{57}$$

where $\boldsymbol{g}_n = \mathbf{C}_n^T\mathbf{B}_n^T\boldsymbol{y}_n$. Thus, if we compute the Cholesky decomposition of $\mathbf{I}_r + \sigma_e^{-2}\mathbf{W}_n$, that is $\mathbf{I}_r + \sigma_e^{-2}\mathbf{W}_n = \mathbf{F}_n\mathbf{F}_n^T$, and set $\boldsymbol{h}_n = \mathbf{F}_n^{-1}\boldsymbol{g}_n$, then the objective function can be expressed as,

$$\mathcal{L} \propto 2\sum_{n=1}^{N}\log\det(\mathbf{F}_n) - (\sigma_e^{-2})^2\sum_{n=1}^{N}\|\boldsymbol{h}_n\|_2^2 + \sigma_e^{-2}\sum_{n=1}^{N}\|\boldsymbol{y}_n\|_2^2. \tag{58}$$

□

proof of lemma 2. We use a similar approach with the process of simplification of log-likelihood to reduce the computational cost of evaluating the log likelihood gradients when we apply gradient decent algorithm. The gradient of the log likelihood with respect to $\boldsymbol{\theta}_\mu$ given in (26) can be simplified to

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{\theta}_{\boldsymbol{\mu}}} = \sum_{n=1}^{N} -2\sigma_e^{-2}(\mathbf{H}_n^T\boldsymbol{y}_n - \mathbf{H}_n^T\mathbf{H}_n\boldsymbol{\theta}_\mu) + 2\sigma_e^{-2}\mathbf{H}_n^T\mathbf{B}_n\mathbf{E}_n^T\mathbf{E}_n(\mathbf{B}_n^T\boldsymbol{y}_n - \mathbf{B}_n^T\mathbf{H}_n\boldsymbol{\theta}_\mu), \tag{59}$$

where $\mathbf{E}_n = \mathbf{L}_n^{-1}\mathbf{C}_n^T$ and $\mathbf{L}_n$ is the Cholesky decomposition of $\sigma_e^2\mathbf{I}_r + \mathbf{C}_n^T\mathbf{B}_n^T\mathbf{B}_n\mathbf{C}_n$, i.e. $\mathbf{L}_n\mathbf{L}_n^T = \sigma_e^2\mathbf{I}_r + \mathbf{C}_n^T\mathbf{B}_n^T\mathbf{B}_n\mathbf{C}_n$. □

proof of lemma 3. Similarly with the proof process of simplification of log-likelihood, the gradient of the log likelihood with respect to σ_e^2 given in (27) can be simplified as

$$\frac{\partial \mathcal{L}}{\partial \sigma_e^2} = \sum_{n=1}^{N} \left[\sigma_e^{-2} m_n - \sigma_e^{-2} \mathrm{tr}(\mathbf{E}_n^T \mathbf{E}_n \mathbf{B}_n^T \mathbf{B}_n) - (\Sigma_n^{-1} \boldsymbol{y}_n)^T (\boldsymbol{\Sigma}_n^{-1} \boldsymbol{y}_n) \right], \tag{60}$$

where $\boldsymbol{\Sigma}_n^{-2} y_n = \sigma_e^{-2} \boldsymbol{y}_n - \sigma_e^{-2} \mathbf{B}_n \mathbf{E}_n^T \mathbf{E}_n \mathbf{B}_n^T \boldsymbol{y}_n$. □

proof of lemma 4. Again using the determinant and Sherman-Morrison-Woodbury lemmas, we replace $\boldsymbol{\Sigma}_n^{-1}$ in (29) by

$$\sigma_e^{-2} (\mathbf{I}_{m_n} - \mathbf{B}_n \mathbf{C}_n \{\sigma_e^2 \mathbf{I}_r + \mathbf{W}_n\}^{-1} \mathbf{C}_n^T \mathbf{B}_n^T). \tag{61}$$

Then, the final form of $\frac{\partial \mathcal{L}}{\partial \mathbf{C}_n}$ is

$$\frac{\partial \mathcal{L}}{\partial \mathbf{C}_n} = 2\sigma_e^{-2} (\mathbf{B}_n^T \mathbf{B}_n \mathbf{C}_n - \mathbf{B}_n^T \mathbf{K}_n \mathbf{W}_n) - 2(\sigma_e^2)^{-2} (\mathbf{B}_n^T - \mathbf{B}_n^T \mathbf{K}_n \mathbf{C}_n^T \mathbf{B}_n^T) \mathbf{S}_n (\mathbf{B}_n \mathbf{C}_n - \mathbf{K}_n \mathbf{W}_n),$$

where $\mathbf{K}_n = \mathbf{B}_n \mathbf{C}_n \{\sigma_e^2 \mathbf{I}_r + \mathbf{W}_n\}^{-1}$. □

References

CAI, T. and YUAN, M. (2010). Nonparametric covariance function estimation for functional and longitudinal data. *University of Pennsylvania and Georgia inistitute of technology*.

CAI, T. T., YUAN, M. et al. (2011). Optimal estimation of the mean function based on discretely sampled functional data: Phase transition. *The annals of statistics* **39** 2330–2355.

DRAKE, A., DJORGOVSKI, S., MAHABAL, A., BESHORE, E., LARSON, S., GRAHAM, M., WILLIAMS, R., CHRISTENSEN, E., CATELAN, M., BOATTINI, A. et al. (2009). First results from the catalina real-time transient survey. *The Astrophysical Journal* **696** 870.

DRAKE, A., GRAHAM, M., DJORGOVSKI, S., CATELAN, M., MAHABAL, A., TORREALBA, G., GARCÍA-ÁLVAREZ, D., DONALEK, C., PRIETO, J., WILLIAMS, R. et al. (2014). The catalina surveys periodic variable star catalog. *The Astrophysical Journal Supplement Series* **213** 9.

JAMES, G. M., HASTIE, T. J. and SUGAR, C. A. (2000). Principal component models for sparse functional data. *Biometrika* **87** 587–602.

JIANG, C.-R., WANG, J.-L. et al. (2010). Covariate adjusted functional principal components analysis for longitudinal data. *The Annals of Statistics* **38** 1194–1226.

JIANG, C.-R., WANG, J.-L. et al. (2011). Functional single index models for longitudinal data. *The Annals of Statistics* **39** 362–388.

KAYANO, M. and KONISHI, S. (2009). Functional principal component analysis via regularized Gaussian basis expansions and its application to unbalanced data. *Journal of Statistical Planning and Inference* **139** 2388–2398.

LI, G., SHEN, H. and HUANG, J. Z. (2016). Supervised sparse and functional principal component analysis. *Journal of Computational and Graphical Statistics* **25** 859–878.

LIN, L., ST. THOMAS, B., ZHU, H. and DUNSON, D. B. (2017). Extrinsic local regression on manifold-valued data. *Journal of the American Statistical Association* **112** 1261–1273.

PAUL, D., PENG, J. et al. (2009). Consistency of restricted maximum likelihood estimators of principal components. *The Annals of Statistics* **37** 1229–1271.

PENG, J. and PAUL, D. (2009). A geometric approach to maximum likelihood estimation of the functional principal components from sparse longitudinal data. *Journal of Computational and Graphical Statistics* **18** 995–1015.

PEZZULLI, S. (1993). Some properties of smoothed principal components analysis for functional data. *Computational Statistics* **8** 1–16.

RAMSAY, J. and SILVERMAN, B. (2005). Principal components analysis for functional data. *Functional Data Analysis* 147–172.

SILVERMAN, B. W. et al. (1996). Smoothed functional principal components analysis by choice of norm. *The Annals of Statistics* **24** 1–24.

SUAREZ, A. J., GHOSAL, S. et al. (2017). Bayesian estimation of principal components for functional data. *Bayesian Analysis* **12** 311–333.

VAN DER LINDE, A. (2008). Variational bayesian functional PCA. *Computational Statistics & Data Analysis* **53** 517–533.

WOOD, S. N. (2006). Low-rank scale-invariant tensor product smooths for generalized additive mixed models. *Biometrics* **62** 1025–1036.

YAO, F., MÜLLER, H.-G. and WANG, J.-L. (2005). Functional data analysis for sparse longitudinal data. *Journal of the American Statistical Association* **100** 577–590.

ZHANG, X., PARK, B. U. and WANG, J.-L. (2013). Time-varying additive models for longitudinal data. *Journal of the American Statistical Association* **108** 983–998.

ZHANG, X., WANG, J.-L. et al. (2016). From sparse to dense functional data and beyond. *The Annals of Statistics* **44** 2281–2321.

ZHOU, L., HUANG, J. Z. and CARROLL, R. J. (2008). Joint modelling of paired sparse functional data using principal components. *Biometrika* **95** 601–619.

ZHU, H., CHEN, Y., IBRAHIM, J. G., LI, Y., HALL, C. and LIN, W. (2009). Intrinsic Regression Models for Positive-Definite Matrices With Applications to Diffusion Tensor Imaging. *Journal of the American Statistical Association* **104** 1203-1212. PMID: 20174601.