Spline estimation of functional principal components via manifold conjugate gradient algorithm

Shiyuan He1,2 · Hanxuan Ye1 · Kejun He1

Received: 8 December 2021 / Accepted: 22 October 2022 / Published online: 9 November 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract

Functional principal component analysis has become the most important dimension reduction technique in functional data analysis. Based on B-spline approximation, functional principal components (FPCs) can be efficiently estimated by the expectation-maximization (EM) and the geometric restricted maximum likelihood (REML) algorithms under the strong assumption of Gaussianity on the principal component scores and observational errors. When computing the solution, the EM algorithm does not exploit the underlying geometric manifold structure, while the performance of REML is known to be unstable. In this article, we propose a conjugate gradient algorithm over the product manifold to estimate FPCs. This algorithm exploits the manifold geometry structure of the overall parameter space, thus improving its search efficiency and estimation accuracy. In addition, a distribution-free interpretation of the loss function is provided from the viewpoint of matrix Bregman divergence, which explains why the proposed method works well under general distribution settings. We also show that a roughness penalization can be easily incorporated into our algorithm with a potentially better fit. The appealing numerical performance of the proposed method is demonstrated by simulation studies and the analysis of a Type Ia supernova light curve dataset.

Keywords Functional data analysis · Stiefel manifold · Positive definite cone · Product manifold · Matrix Bregman divergence

1 Introduction

The statistical analysis of data in the form of curves, commonly referred to as functional data analysis (FDA, Ramsay and Silverman 2002), has gained increasing research interest due to the improved capability of recording a vast amount of data in many scientific fields. Among the developed methodology, functional principal component analysis (FPCA) is a key technique and fundamental tool for analyzing functional data. It provides an informative way to explore the infinite-dimensional functional data through its variance-covariance structure and discovers features to characterize the curves (Ramsay and Silverman 2005). Based on the computed functional principal components (FPCs), various multivariate data analysis methods can then be applied on the principal component scores, such as regression (Ramsay and Dalzell 1991; Yao et al. 2005), clustering (James and Sugar 2003; Chiu and Li 2007), and classification (James and Hastie 2001).

Consider a random function \( x(u) \) with a time index \( u \in \mathcal{U} \), where \( \mathcal{U} \) is a compact set in \( \mathbb{R} \). Without loss of generality, we assume \( \mathcal{U} = [0,1] \). The function \( x(\cdot) \) belongs to a Hilbert space \( \mathcal{H} \subseteq L_2(\mathcal{U}) \) equipped with the inner product \( \langle x, \tilde{x} \rangle_{\mathcal{H}} = \int_{\mathcal{U}} x(u) \tilde{x}(u) \, du \) for \( x, \tilde{x} \in \mathcal{H} \). Suppose the random element \( x \) has zero mean and covariance function \( K(u,v) = E[x(u)x(v)] \). Under mild conditions, the Mercer’s lemma states there exists an orthonormal sequence \( \{\psi_r(u)\}_r \) of eigenfunctions in \( \mathcal{H} \), and a decreasing non-negative sequence \( \{\lambda_r\}_r \) of eigenvalues, such that the covariance function can be expanded as
The non-negative sequence \( \{ \lambda_0r \} \) is assumed to satisfy the trace-class condition \( \sum_{r=1}^{\infty} \lambda_0r < \infty \). Using the Karhunen-Loève expansion (Karhunen 1946; Loève 1946), the random functions \( x \in \mathcal{H} \) can be represented as \( x(u) = \sum_{r=1}^{\infty} \lambda_0r^{1/2} \psi_0r(u) \), where \( \psi_r \)'s are uncorrelated random variables with zero mean and unit variance. Though functional data reside in infinite dimensional space, the leading \( R \) eigenfunctions usually have high quality approximation power such that \( K(u, v) \approx \sum_{r=1}^{R} \lambda_0r \psi_0r(u) \psi_0r(v) \) and \( x(u) \approx \sum_{r=1}^{R} \lambda_0r^{1/2} \psi_0r(u) \). It is equivalent to assuming the sequence of eigenvalues \( \{ \lambda_0r \} \) decays fast. The eigenfunctions \( \psi_01, \cdots, \psi_0R \) are called FPCs, and the term \( R \) is usually chosen such that the leading components explain a desired proportion of variability in the dataset.

In practice, we have \( N \) realized functions \( x_1, \cdots, x_N \) of \( x \). Each function \( x_n \) is discretely observed at \( M_n \) time points \( u_{n1}, \cdots, u_{nM_n} \in \mathcal{U} \). The observed function value \( y_{nj} \) at \( u_{nj} \) is corrupted by noise as

\[
y_{nj} = x_n(u_{nj}) + \epsilon_{nj}, \quad n = 1, \cdots, N, \quad j = 1, 2, \cdots, M_n, \tag{2}
\]

where the noise \( \epsilon_{nj} \)'s are independent with zero mean and finite variance \( \sigma^2 = \mathbb{E} \epsilon_{nj}^2 \). The goal of FPCA is to recover \( \psi_01, \cdots, \psi_0R \) from these \( N \) sparse samples \( \{ u_{nj}, y_{nj} \}_{n,j} \).

In the past decades, significant interests have been paid to the methodological and theoretical properties of FPCA. Some early works (Rice and Silverman 1991; Silverman 1996) constructed the estimation based on densely sampled curves. For sparsely observed curves, Yao et al. (2005) estimated the FPCs through learning the mean curve and the covariance surface via weighted local regression, but its convergence result was suboptimal. For the local smoothing method, the optimal rate was later established by Hall et al. (2006). More extensive theoretical investigation of the local smoothing methods has been carried out by Li and Hsing (2010) and Zhang and Wang (2016). By assuming that the sample path belongs to a reproducing kernel Hilbert space (RKHS), Cai and Yuan (2010) proposed an estimation based on RKHS kernels and proved its consistency.

The above non-parametric approaches (Yao et al. 2005; Cai and Yuan 2010) are generally computational intensive. This is because either the local regression has to be computed at a dense grid of time points (Yao et al. 2005) or the number of parameters grow with the sample size (Cai and Yuan 2010). Based on spline expansion, one can establish optimization problems with a smaller number of unknown parameters (James et al. 2000; Peng and Paul 2009). When the scores and the noises follow Gaussian distribution, the vector \( y_n = (y_{n1}, y_{n2}, \cdots, y_{nM_n})^T \) is also multivariate Gaussian distributed. The estimates of the principal component functions can be obtained by maximizing the log-likelihood with spline coefficients as parameters. To solve the optimization problem, James et al. (2000) augmented the parameter space with the unobserved scores and used the expectation-maximization (EM) algorithm. In this optimization, the parameter space is inherently endowed with a geometric structure due to the orthonormality of the eigenfunctions and the positive semi-definiteness of the covariance function. However, the EM algorithm does not fully exploit the geometric structure during its update, except for an ad hoc post-processing step of orthogonalization at the end of each iteration to respect this structure. The work of Peng and Paul (2009) explored another approach based on a geometric restricted maximum likelihood (REML). Though REML considers the manifold structure and is shown to improve accuracy over EM in Peng and Paul (2009), REML is found not stable and fails to converge in some replicates. This issue becomes worse when its initial value is not close enough to the optimal one. Figure 1 presents the result of one replicate in the pracSin simulation setting with sample size \( N = 100 \). The pracSin setting is similar the practical setup in Peng and Paul (2009) such that there are 5 eigenfunctions generated from a random combination of 10 sinusoid basis functions (see Section 5.1 for more details). The figure depicts the estimation error versus computational time for EM and REML (along with our proposed method, named CG). The three panels from left to right correspond to the leading three eigenfunctions. The \( L_2 \) distance between the estimated and true eigenfunctions are shown in the vertical axis. In Fig. 1, the unconvergence behavior of REML (the blue dashed curve) is evident, while EM (the green dashed curve) appears not to search very efficiently.

The purpose of this work is threefold. First, though both EM and REML are derived from the negative Gaussian log-likelihood, the estimates are consistent under broader distribution settings. In Paul and Peng (2009), the authors remarked that Gaussianity is more of a working assumption and can be relaxed by assuming an appropriate tail behavior of the observations. In this work, we show the corresponding loss function can be explained from the perspective of matrix Bregman divergence (Pitrik and Virostek 2015; Dhillon and Tropp 2008), which is free of distribution assumption. This distribution-free motivation helps to explain why solving the loss function of EM and REML sometimes works well under more general distributions.

Second, the derived objective function with respect to spline coefficients is constrained on a rank-\( R \) matrix. Though the geometric structure of the parameter space is exploited in REML, its numerical results turn out to be unstable and heavily rely on the choice of initial value (Peng and Paul 2009). Through looking insight into the manifold geometry...
(Absil et al. 2009; Hu et al. 2020), we propose a conjugate gradient algorithm for FPC estimation according to the recent advances in matrix manifold optimization. In particular, we factorize the fixed rank-\( R \) symmetric matrix as a product of two matrices, such that one belongs to the Stiefel manifold (Edelman et al. 1998) and the other is within the positive definite cone (Sra and Hosseini 2015). In contrast to Peng and Paul (2009), we view the two matrices together as an element of the corresponding product manifold, and the conjugate gradient algorithm is deployed over this product structure by taking into account its intrinsic geometry. Figure 1 also plots the convergence trace of our proposed method (CG) in solid red curve. The improvement over REML and EM is substantial.

Our third task is to compare two approaches of tuning empirically. One is using regression splines where the tuning parameter is the number of knots; the other is using penalized splines with a roughness penalty parameter to be tuned. Both REML and EM are based on regression splines with equally spaced knots within the domain, and the number of knots needs to be tuned. One drawback of this approach is the discrete tuning in the sense that the number of knots is selected from a set of discrete integers, which may result in unstable estimation. We suggest using penalized splines (O’Sullivan 1986, 1988; Huang and Su 2021) by fixing the number of knots at a relatively large value and incorporating the roughness penalization to achieve continuous tuning. Our optimization algorithm allows us to directly integrate the roughness penalty, and we find that the penalized spline estimator potentially provides more accuracy in estimation and prediction compared with the one using regression splines.

The rest of the article is organized as follows. Our spline estimator is motivated from the matrix Bregman divergence, and the corresponding optimization problem is formulated in Sect. 2. We review basic concepts of product manifold geometry in Sect. 3. Based on these concepts, a matrix manifold conjugate gradient algorithm is proposed in Section 4. In Sect. 5, our method is compared with some existing alternatives using both simulated and real datasets. This paper is summarized in Sect. 6 with a brief concluding remark.

2 Methods

This section formulates the optimization problem to estimate the leading \( R \) FPCs from sparsely sampled data. Our method is built upon a rank-\( R \) covariance function \( C \), defined in a tensor product spline space, to approximate the true covariance function \( \mathcal{K} \) in (1). In particular, over the domain \( \mathcal{U} \), we let \( \mathbf{b}(\cdot) \in \mathbb{R}^K \) be an orthonormalized spline basis
with degrees of freedom $K$. The orthonormality means that \( \int b(u)b^\top(u) \, du = I \in \mathbb{R}^{K \times K} \). This orthonormality requirement facilitates the interpretation and computation of the functional principal component by transforming the $L_2$ norms (or inner products) of eigenfunctions to the Euclidean norms (or inner products respectively) of their basis coefficients vectors. Meanwhile, the choice of spline basis order is closely related to the smoothness of the underlying eigenfunctions. We suggest using the cubic splines (4th order) as in James et al. (2000); Huang and Su (2021) when the prior information about the eigenfunction smoothness is unknown. Given the vector of basis functions, we model the covariance function as \( C(u, v) = b^\top(u)Db(v) \) using a matrix $D \in \mathbb{R}^{K \times K}$ which is positive semi-definite with rank $R$. Suppose its eigen-decomposition is $D = UWU^\top$, where $U \in \mathbb{R}^{K \times R}$ has orthonormal columns and $W = \text{diag}(\lambda_1 , \ldots, \lambda_K) \in \mathbb{R}^{R \times R}$ is a diagonal matrix. Let $u_r$ be the $r$-th column of $U$ and denote $\psi_r(u) = u_r^\top b(u)$. It can be shown straightforwardly that

\[
\int \psi_r(u)\psi_r'(u) \, du = u_r^\top \left\{ \int b(u)b^\top(u) \, du \right\} u_r' = u_r^\top u_r' = \delta_{r,r'},
\]

where $\delta_{r,r'}$ is the Dirac delta, and that

\[
\int C(u, v)\psi_r(v) \, dv = b(u)^\top UWU^\top \left\{ \int b(u)b^\top(v) \, dv \right\} u_r = b(u)^\top UWU^\top u_r = \lambda_r b(u)^\top u_r = \lambda_r \psi_r(u).
\]

Thus, written in this way, the modeled covariance function $C$ has $\lambda_r$ as its eigenvalue, and the corresponding eigenfunction is $\psi_r(u) = u_r^\top b(u)$, for $r = 1, \ldots, R$.

For the $n$-th sample $y_n = (y_{n1}, \ldots, y_{nM_n})^\top$ with $y_{nj}$ in (2), $n = 1, \ldots, N$ and $j = 1, 2, \ldots, M_n$, its covariance matrix is

\[
\mathcal{K}_n = [K(u_{nj}, u_{nj'})]_{j,j'} + \sigma_e^2 I.
\]

Due to our spline representation of the covariance function $C(u, v) = b^\top(u)Db(v)$ with $D = UWU^\top$, $\mathcal{K}_n$ in (4) will then be approximated by

\[
\Sigma_n = [\mathcal{C}(u_{nj}, u_{nj'})]_{j,j'} + \sigma_e^2 I,
\]

with parameters $U$, $W$, and $\sigma_e^2$. To estimate the unknown parameters $U$, $W$ and $\sigma_e^2$, we propose a loss function based on the matrix Bregman divergence (Pitrik and Virotszek 2015; Dhillon and Tropp 2008) to measure the discrepancy between the covariance matrices $\mathcal{K}_n$ in (4) and $\Sigma_n$ in (5).

A matrix Bregman divergence is constructed with a strictly convex and twice continuously differentiable function $\varphi(\cdot) : \mathbb{R} \rightarrow \mathbb{R}$. This function can be applied to the positive definite matrix $\varphi(\Sigma_n) \in \mathbb{R}^{M \times M}$ via its eigendecomposition $\Sigma_n = F_nG_nF_n^\top$, where the columns of $F_n$ contain the eigenvectors of $\Sigma_n$ and $G_n = (g_{n1}, \ldots, g_{nM_n})$ is the diagonal matrix of its eigenvalues. We denote $\varphi(\Sigma_n) = F_n\varphi(G_n)F_n^\top$, where $\varphi(G_n) = \text{diag}(\varphi(g_{n1}), \ldots, \varphi(g_{nM_n}))$ such that $\varphi$ is applied elementwisely to the diagonal. The function $\varphi$ induces a matrix Bregman divergence for the pair of positive definite matrices $\mathcal{K}_n$ and $\Sigma_n$ as

\[
D_\varphi(\mathcal{K}_n || \Sigma_n) = tr \left\{ \varphi(\mathcal{K}_n) - \varphi(\Sigma_n) - \varphi'(\Sigma_n)(\mathcal{K}_n - \Sigma_n) \right\},
\]

and we call $\varphi$ the seed function. This divergence gauges the similarity between the true covariance matrix $\mathcal{K}_n$ and our model covariance $\Sigma_n$, and (6) is zero if and only if $\mathcal{K}_n = \Sigma_n$.

In this work, we choose the seed function $\varphi(x) = -\log(x)$, and the divergence loss becomes the LogDet divergence

\[
D_\varphi(\mathcal{K}_n || \Sigma_n) = tr \left\{ -\log(\mathcal{K}_n) + \log(\Sigma_n) + \Sigma_n^{-1}(\mathcal{K}_n - \Sigma_n) \right\}.
\]

Note also the true covariance matrix $\mathcal{K}_n$ is not observed in practice. Fortunately, we have its one-sample estimate $S_n = y_ny_n^\top$, which is an unbiased estimate of $\mathcal{K}_n$ at the same observational points $\{u_{nj} : j = 1, \ldots, M_n\}$ of $y_n$, $n = 1, \ldots, N$. Notice that the functional data are assumed to be sparsely observed, and the observational time points $\{u_{nj} : j = 1, \ldots, M_n\}$ are randomly drawn from the domain $\mathcal{U}$. This means, for the $n$-th sample, the random observational time points $\{u_{nj} : j = 1, \ldots, M_n\}$ are unique to this particular sample. The one-sample estimate $S_n$ therefore provides the only available information for the corresponding true covariance matrix $\mathcal{K}_n$ in (4) at the observational time points of $y_n$. Furthermore, the right hand side of (7) indicates that the first term $\log(\mathcal{K}_n)$ is fixed with respect to the parameters. Therefore, discarding the first term $\log(\mathcal{K}_n)$ and replacing the last $\mathcal{K}_n$ by its one-sample estimate $S_n$, we get the loss function over the $N$ samples

\[
\mathcal{L}(U, W, \sigma_e^2) = \frac{1}{N} \sum_{n=1}^{N} \left\{ \log \det S_n + \langle S_n^{-1}, S_n - \Sigma_n \rangle \right\} = \frac{1}{N} \sum_{n=1}^{N} \left\{ \log \det S_n + \langle S_n^{-1}, S_n \rangle \right\} + C,
\]

where $C$ is a relative constant. From the above, we can see this loss corresponds to the negative log-likelihood of $y_n$ when the
data is assumed to follow multivariate Gaussian distribution. The same loss has been used in James et al. (2000) and Peng and Paul (2009) where Gaussian distribution was assumed. The above derivation motivates that the loss function works under more general settings without assuming Gaussianity. Given $N$ sparse samples and the loss function (8), the estimation can be obtained via solving

$$\arg\min_{U,W,\sigma^2_x} \mathcal{L}(U, W, \sigma^2_x),$$
subject to $U \in \text{St}(R, K)$, $W = \text{diag}(\lambda_1, \cdots, \lambda_R) > 0$, and $\sigma^2_x > 0,$

where $\text{St}(R, K) = \{U \in \mathbb{R}^{K \times R} : U^TU = I\}$ is the Stiefel manifold (Edelman et al. 1998) and all the eigenvalues $\lambda_r$ are required to be non-negative. In (9), the noise variance $\sigma^2_x$ is also a parameter to be estimated. Denote $(\hat{U}, \hat{W})$ and $\hat{\sigma}^2_x$ as the optimal solution to problem (9). The $r$-th eigenfunction is then estimated by $\hat{\psi}_r(u) = b^T(u)\hat{u}_r$, where $\hat{u}_r$ is the $r$-th column of $\hat{U}$. Requiring $U$ to belong to the Stiefel manifold $\text{St}(R, K)$ makes the estimates $\hat{\psi}_r(u)$ compatible with the eigenfunctions orthonormality property (3).

3 Geometry of the product manifold

Before presenting the algorithm to solve (9), we first note that $U$ and $W$ belong to the Stiefel manifold $\text{St}(R, K)$ (Edelman et al. 1998) and the cone of positive definite matrices $\mathcal{S}_{++}(R)$ (Smith 2005; Sra and Hosseini 2015), respectively. The factored matrices $(U, W)$ are identifiable up to a column/row permutation. The diagonal constraint on $W$ in (9) can be too restricted for an efficient search of $(U, W)$. We consider an enlarged parameter space for the possibility of a shorter path via which the optimal value is reached faster. In particular, we let $\mathcal{M}$ denote the product manifold (Absil et al. 2009) of $\text{St}(R, K)$ and $\mathcal{S}_{++}(R)$, and treat $(U, W)$ as a point on $\mathcal{M}$, i.e.,

$$x = (U, W) \in \mathcal{M} = \text{St}(R, K) \times \mathcal{S}_{++}(R).$$

The above $\mathcal{M}$ is a submanifold of $\mathbb{R}^{K \times R} \times \mathbb{R}^{R \times R}$ and its manifold topology is equivalent to the product topology (Absil et al. 2009) of $\text{St}(R, K)$ and $\mathcal{S}_{++}(R)$. To improve the search efficiency of parameter $x = (U, W)$, the geometrical aspects of $\mathcal{M}$ need to be taken into account, and we present the geometric structures of the product manifold in this section.

The tangent space.

As a product manifold, the tangent space at $x = (U, W) \in \mathcal{M}$ is denoted as

$$\mathcal{T}_x\mathcal{M} = \mathcal{T}_U\text{St}(R, K) \times \mathcal{T}_W\mathcal{S}_{++}(R),$$

where $\mathcal{T}_U\text{St}(R, K)$ and $\mathcal{T}_W\mathcal{S}_{++}(R)$ are the tangent spaces of $\text{St}(R, K)$ at $U$ and $\mathcal{S}_{++}(R)$ at $W$, respectively. In other words, each tangent vector $\xi_x = (\xi_U, \xi_W)$ has two components: one is $\xi_U$ in the tangent space of Stiefel manifold $\mathcal{T}_U\text{St}(R, K)$, and the other is $\xi_W$ in the tangent space of positive definite manifold $\mathcal{T}_W\mathcal{S}_{++}(R)$. Both tangent spaces have explicit forms, i.e.,

$$\mathcal{T}_U\text{St}(R, K) = \{U\Omega + U^\perp K : \Omega^T = -\Omega, K \in \mathbb{R}^{(K-R) \times R}\}$$

$$\mathcal{T}_W\mathcal{S}_{++}(R) = \{K \in \mathbb{R}^{R \times R} : K = K^T\}.$$
4 Conjugate gradient algorithm over the product manifold

We now present how to solve (9) by alternatively updating $(U, W)$ and $\sigma_v^2$ in a cyclic manner. In other words, with fixed $\sigma_v^2$, the pair of matrices $(U, W)$ gets updated; then, with fixed $(U, W)$, the noise variance $\sigma_v^2$ gets updated. The latter step is relatively straightforward, as $\sigma_v^2$ can be updated by a classical gradient method. For updating $(U, W)$, we propose to use conjugate gradient algorithm (Nocedal and Wright 2006) over the product manifold $\mathcal{M}$ and will present it afterward. Algorithm 1 summarizes the main steam.

**Algorithm 1: The Main Algorithm.**

Input: Initial value for $U, W, \sigma_v^2$.
Output: The estimates of $U, W, \sigma_v^2$.

1 repeat
2 For fixed $\sigma_v^2$, update $(U, W) \in \mathcal{M}$ by the conjugate gradient algorithm in Algorithm 2.
3 For fixed $(U, W)$, update $\sigma_v^2$ by a classical gradient method.
4 until convergence.

The conjugate gradient algorithm to update $(U, W)$ is presented in Algorithm 2 aiming to minimize (9) efficiently. Compared with the steepest descent, the conjugate gradient algorithm accelerates the convergence rate (Bertsekas 1999; Nocedal and Wright 2006) by adding an additional term to ensure conjugacy. Meanwhile, unlike the Newton’s method, the conjugate gradient algorithm also reduces the cost of numerical iterations by avoiding evaluating the inverse of the Hessian operator (Bertsekas 1999). Algorithm 2 works on the product manifold and relies on two major components. The first is to exploit the manifold structure to update $x$, which has been discussed in Sect. 3. The second is computing the partial derivatives (13) and (14) of $F$ in the Euclidean space. In particular, set $F(U, W) := L(U, W, \sigma_v^2)$ for the loss function (8) in the Euclidean space with $\sigma_v^2$ fixed. By a direct application of the chain rule, the partial derivative of $F$ w.r.t. $U$ and $W$ are

$$\frac{\partial F(U, W)}{\partial U} = \frac{1}{N} \sum_{n=1}^{N} 2B_n^T \frac{\partial L_n(S_n)}{\partial \Sigma_n} B_n U W$$

and

$$\frac{\partial F(U, W)}{\partial W} = \frac{1}{N} \sum_{n=1}^{N} U^TB_n^T \frac{\partial L_n(S_n)}{\partial \Sigma_n} B_n U$$

where

$$\frac{\partial L_n(S_n)}{\partial \Sigma_n} = \Sigma_n^{-1} - \Sigma_n^{-1} \Sigma_n \Sigma_n^{-1}.$$
Algorithm 2: The Conjugate Gradient Algorithm over $\mathcal{M}$.

**Input:** Initial iterate $x_0 = (U_0, W_0) \in \mathcal{M}$.

**Output:** Computed estimations for $U$ and $W$ when $\sigma^2$ is fixed.

1. Set $k = 0$.
2. Compute $\frac{\partial F}{\partial U}$ and $\frac{\partial F}{\partial W}$ at the current $x_0$.
3. Compute the gradient $\nabla f(x_0) = \nabla \log L(x_0) (\frac{\partial F}{\partial U}, \frac{\partial F}{\partial W})$ via (11).
4. Set the descent direction $\eta_0 = -\nabla f(x_0)$.
5. repeat
   6. Select the step size $\alpha_k \in \mathbb{R}$ by the Wolfe condition and set $x_{k+1} = \mathcal{R}_{x_k} (x_k + \alpha_k \eta_k)$.
   7. Compute $\xi_{k+1} = \mathcal{R}_{x_k} (x_{k+1} - x_k + \alpha_k \eta_k)$.
   8. Compute $\frac{\partial F}{\partial U}$ and $\frac{\partial F}{\partial W}$ at the current $x_k = (U_k, W_k)$.
   9. Compute the gradient $\nabla f(x_{k+1}) = \nabla \log L(x_{k+1}) (\frac{\partial F}{\partial U}, \frac{\partial F}{\partial W})$ via (11).
   10. Set $\eta_{k+1} = -\nabla f(x_{k+1}) + \beta_k \xi_{k+1}$ with $\beta$ being specified by (15) or (16).
   11. Set $k = k + 1$.
   until convergence.
13. Compute matrix product $D = U_k W_k U_k^T$.
14. Compute the eigendecomposition of $D = \text{VEV} \text{T}$. Let the estimator of $U$ be the first $R$ columns of $V$. Let the estimator of $W$ be the first $R$ diagonal elements of $E$.

To be consistent with Section 3, we use $\tilde{f}$ to denote $F$ when the domain is restricted to the product manifold $\mathcal{M}$.

In Algorithm 2, there are five main steps (Line 6–10). In Line 6, we move the current point $x_k = (U_k, W_k)$ along the search direction $\eta_k$ and with the retraction operation $\mathcal{R}_{x_k}$ in (12). The step size $\alpha_k$ is selected by the Wolfe condition (see Nocedal and Wright (2006) and its manifold counterpart which can be found in, for example, in (Ring and Wirth 2012)). In Line 7, the search direction $\eta_k$ is transported onto the tangent space of the updated point $x_{k+1}$. We calculate the Euclidean derivatives of the objective function $F$ in Line 8, and then project them onto the manifold tangent space in Line 9. After that, in Line 10, a new search direction $\eta_{k+1}$ is computed as a combination of the translated search direction $\xi_{k+1}$ and the calculated negative manifold gradient $-\nabla f(x_{k+1})$. The weights of this combination is based on the commonly used Fletcher-Reeves (Fletcher and Reeves 1971) rules. To be specific, we have

1. Fletcher-Reeves:
   
   $$\beta_{k+1} = \frac{\langle \nabla f(x_{k+1}), \nabla f(x_{k+1}) \rangle}{\langle \nabla f(x_k), \nabla f(x_k) \rangle};$$  

2. Polak-Ribière:
   
   $$\beta_{k+1} = \frac{\langle \nabla f(x_{k+1}), \nabla f(x_{k+1}) - \xi_{k} \rangle}{\langle \nabla f(x_k), \nabla f(x_k) \rangle};$$  

where $\xi_k = \mathcal{P}_{x_k \to x_{k+1}} (\nabla f(x_k)$).

The sequence of directions $\eta_0, \eta_1, \eta_2, \ldots$ for updating $x = (U, W)$ resembles the classical conjugate gradient direction (Nocedal and Wright 2006). For the classical conjugate gradient algorithm with the variable $x \in \mathbb{R}^p$ in a Euclidean space, suppose the (local) objective function $f(x)$ has a quadratic form like $f(x) = x^T A x + b^T x$ for some positive definite matrix $A$ and vector $b$ of proper size. In this case, the classical conjugate gradient algorithm ensures the updating directions are conjugate to each other, i.e., $\eta_i^T A \eta_j = 0$ for $i \neq j$. Moreover, the classical conjugate gradient method ensures that $x_{k+1}$ minimizes $f(x)$ inside the subspace $x_0 + \text{span}(\eta_0, \eta_1, \ldots, \eta_k)$, where $x_0$ is the initial value. As a notable feature of the classical conjugate gradient algorithm, the next direction $\xi_{k+1}$ can be computed as a linear combination of the gradient at the current point $-\nabla f(x_{k+1})$ and the previous search direction $\eta_k$. This is the same as Line 10 of Algorithm 2, except that the previous search direction $\tilde{\xi}_{k+1}$ is vector transported to the current point $x_{k+1}$ in Line 7.

Finally, note that the working domain $\mathcal{M} = \text{St}(R, K) \times S_{+}(R)$ of Algorithm 2 is larger than the constraint of (9). In (9), the matrix $W$ is restricted to be a diagonal matrix with positive elements. At the end (Line 13–14) of Algorithm 2, we recover this structure. More precisely, let $(U_k, W_k) \in \mathcal{M}$ be the output of the iterative steps (Line 5–12) in Algorithm 2. We then compute the matrix product $D := U_k W_k U_k^T$ and its eigendecomposition $D = \text{VEV} \text{T}$. The algorithm will output the first $R$ columns of $V$ and the first $R$ diagonal elements of $E$, respectively. The outputs correspond to the updated values of $U$ and $W$ when $\sigma^2$ is fixed in Algorithm 1.

We conclude this section by remarking the computational complexities of the proposed CG method and the alternatives of EM and REML. Recall in our setting, $K$ is the number of basis functions, $R$ is the number of principal component functions (with $R \leq K$), and $M_n$ is the number of observation points for the $n$-th sample, $n = 1, \ldots, N$. For presentation simplicity, we assume $M_1, \ldots, M_N$ are all equal to $M$. The computational complexity of our proposed conjugate gradient algorithm (CG) includes two parts:

C1 Computing the objective function and its Euclidean gradient requires a computational complexity of order $O(N(R^3 + MKR))$, after using the Cholesky decomposition, the matrix determinant lemma, and the Woodbury matrix identity for acceleration.

C2 For manifold-related operations, the computational complexity of calculating the gradient projection, vector transport, and retraction is $O(K^2 R)$.

The computational complexity in C1 also applies to EM (for evaluating the log-likelihood) and REML. The overhead in
C2 for manifold-related computation is minor when the sample size $N$ is relatively large.

5 Numerical studies

In this section, our method is compared with several existing methods both on simulated datasets and a set of real astronomical data. The competing methods have the same loss function (8) as ours, but distinct optimization algorithms are employed. Our method minimizes the loss via the conjugate gradient algorithm over the product manifold in Sect. 4, which will be denoted as CG in the following. The alternative methods include the reduced rank model (James et al. 2000) using the EM algorithm (denoted as EM) and the restricted maximal likelihood method (REML, Peng and Paul 2009). The implementation of the alternative methods is based on the R package fPCA (Peng and Paul 2011). The tuning parameters of all these methods are the number of spline knots, and they are tuned via cross-validation (CV).

5.1 Simulation

In the simulation, the random function are generated by a rank-$R_0$ model

$$x(u) = \sum_{r=1}^{R_0} \lambda_{0r}^{1/2} \theta_r \psi_{0r}(u), \quad (17)$$

where $\lambda_{0r}$’s are eigenvalues and $\theta_r$’s are random scores following standard normal distribution. The true eigenfunctions $\psi_{0r}$’s in (17) are constructed from sinusoid functions over $[0,1]$ as in Peng and Paul (2009).

Similar to Peng and Paul (2009), we consider two settings called easySin and pracSin. Generally, the easySin setting has fewer number of eigenfunctions and each eigenfunction (as random combination of 5 sinusoid bases) has less variation within its domain. The eigenfunctions in this setting are relatively easy to be estimated. On the other hand, in the pracSin setting, each eigenfunction (as a random combination of 10 sinusoid bases) has more variation in the domain. The pracSin setting also has a slightly larger number of eigenfunctions and thus is closer to the real cases in practice. More precisely, in the easySin setting, the true rank is $R_0 = 3$ with eigenvalues: 1, 0.66, and 0.517. For the eigenfunctions, we consider 5 sinusoid functions denoted as $f_1(u) = (\sin(u), \sin(2u), \ldots, \sin(5u))$. Though any three of these sinusoidal functions can be directly set as the true eigenfunctions, we decide to make the estimation task more interesting by creating linear combinations of them. To be specific, in each replication, we generate a $5 \times 3$ matrix $T^{(1)}$ whose elements are drawn from the standard normal distribution. Let $Q^{(1)}$ be the Q-factor of the QR decomposition of $T^{(1)}$, so that $Q^{(1)}$ is an orthonormal matrix. For $r = 1, 2, 3$, we set the $r$-th true eigenfunction as $\psi_{0r}(u) = f_1(u) Q^{(1)}_r$ with $Q^{(1)}_r$ being the $r$-th column of $Q^{(1)}$.

The pracSin setting is designed to be a more challenging setting, for which the rank is higher and the eigenfunctions have more complex shapes. Specifically, the true rank is set as $R_0 = 5$ with eigenvalues: 1, 0.66, 0.517, 0.435, and 0.381. Let $f_2(u) = (\sin(u), \sin(2u), \ldots, \sin(10u))$ and consider their linear combinations for eigenfunctions. We similarly denote $Q^{(2)}$ as the orthonormal Q-factor of the QR decomposition of some $10 \times 5$ random matrix $T^{(2)}$, whose elements are independently drawn from the standard normal distribution for different replications. We set the $r$-th true function as $\psi_{0r}(u) = f_2(u) Q^{(2)}_r$ with $Q^{(2)}_r$ being the $r$-th column of $Q^{(2)}$, $r = 1, \ldots, 5$.

To generate $N$ sparse and noisy observations, we begin with the sparsely sampled curves $x_1, \ldots, x_N$. In particular, the $n$-th sample curve $x_n$ has $M_n$ sparse observations, where $M_n$ is drawn from the discrete uniform distribution over $\{2, 3, \ldots, 10\}$. For $j = 1, \ldots, M_n$, the observation time points $u_{nj}$’s are independently drawn from the standard uniform distribution $\mathcal{U} = [0, 1]$. The observed value at $u_{nj}$ is $y_{nj} = x(u_{nj}) + \sigma_e \epsilon_{nj}$ with $\sigma_e = 1/4$. The noise $\epsilon_{nj}$ follows one of the three distributions: the standard normal $\mathcal{N}(0,1)$, the rescaled student t distribution $t_3/\sqrt{3}$ with 3 degrees of freedom, and the uniform distribution $U[-\sqrt{3}, \sqrt{3}]$. In this way, a collection of sparse and noisy observations $\{(u_{nj}, y_{nj})\}_{j=1}^{M_n}$ are generated under each setting, $n = 1, \ldots, N$. For easySin, $N = 50$ samples are generated; as for pracSin, we generate 100 and 500 samples. In each setting, 500 replications are carried out.

The functional principal components are then estimated from the simulated datasets by the proposed CG method and competitive EM and REML. For fitting, cubic spline basis (with order 4) is used. The number of knots is selected among $\{4, 5, \ldots, 12\}$ for the easySin setting, and from $\{8, 9, \ldots, 15\}$ for the pracSin setting. The knot selection is conducted by ten-fold CV, and the loss function used in CV is the same as (8). As in Peng and Paul (2009), we fix the number of principal functions at the true one $R = R_0$ for all methods in the simulation studies. Lastly, to evaluate the performance of various methods, we compute the average estimation errors in $L_2$ norm, i.e., $\|\hat{\psi}_r - \psi_{0r}\|_{L_2}$, for each eigenfunction.

To compare the stability of various methods, we also consider three different types of initial values: LOC, LS, and EM. Specifically, (1) LOC: these initial values are the estimation results of the local polynomial method (LOC, Yao et al. 2005). LOC fits the mean and covariance functions through nonparametric local regression and thus itself does not require initial values. (2) LS: these initial values are obtained by a least squares procedure. It fits the linear model $y_n = B_u y_n + \epsilon_n$ to each sample independently. The ini-
To compute the estimation errors, the performance of REML deteriorates considerably in the sense that its estimation error is much larger when LS or LOC is used for initialization. Moreover, REML has 4%–30% un-converged replicates out of the total 500, especially under the initialization method of LS or LOC.

Note that in this study, we use the converged replicates to compute the estimation errors; the performance of REML may be much worse when a practitioner uses the last iteration as the output in real applications.

This phenomenon is consistent with statements in Peng and Paul (2009) that REML is very sensitive to the choice of initial values and the unconvergence of REML mainly comes from unsuitable initial estimates. Unlike REML, the performance of our proposed CG method is stable across different initials with the highest accuracy for eigenfunction estimation in almost all cases. The improvement of CG over REML illustrates the benefit of the proposed conjugate gradient algorithm. The results in Table 2 are generally similar as those in Table 1. Since the true eigenfunctions have complex shapes under the pracSin setting, the sample size \( N = 100 \) is not sufficient for REML to work well. To be specific, with LS and LOC as initial values, REML has less than 100 converged replicates out of the total 500, and its estimation error is much larger than others. Our proposed CG outperforms the competitors and is more stable than REML and EM with different initial values. When the sample size increases to 500, the performance of all methods gets improved. However, CG remains superior to the others.

Table 1 shows that the performance of the proposed CG is stable and reasonably better than the competitors. Given EM as initial values, the performances of REML and CG are very close generally. However, if the initial values are taken other than EM, the performance of REML deteriorates considerably in the sense that its estimation error is much larger when LS or LOC is used for initialization. Moreover, REML has 4%–30% un-converged replicates out of the total 500, especially under the initialization method of LS or LOC.

Table 1: Simulation result for the easySin setting. The reported values (\( \times 10^{-1} \)) are the average estimation errors for each eigenfunction \( \| \hat{\psi}_r - \psi_0 \|_{L_2} \) based on 500 replications. The reported numbers are the actual values multiplied by 10. The numbers in the parentheses are the standard errors. The first row indicates the initialization methods of EM, LS, and LOC. The row ‘# Conv’ is the number of converged replicates.

| \( N = 50 \) | \( \epsilon \sim \text{Normal} \) | \( \epsilon \sim \text{t}_3 \) | \( \epsilon \sim \text{Unif} \) |
|-------------|-----------------|-----------------|-----------------|
|             | EM (\( \times 10^{-1} \)) | LS (\( \times 10^{-1} \)) | LOC (\( \times 10^{-1} \)) |
| \( \psi_1 \) | 4.57(0.14) | 4.42(0.12) | 6.23(0.15) |
| \( \psi_2 \) | 7.20(0.17) | 7.09(0.16) | 7.06(0.16) |
| \( \psi_3 \) | 6.27(0.16) | 6.24(0.15) | 6.23(0.15) |
| # Conv      | 482 500 362 500 | 399 500 399 500 | 500 500 500 |
| \( \psi_1 \) | 4.87(0.14) | 4.82(0.13) | 4.81(0.13) |
| \( \psi_2 \) | 7.01(0.16) | 7.05(0.15) | 7.03(0.15) |
| \( \psi_3 \) | 6.03(0.15) | 5.91(0.15) | 5.91(0.15) |
| # Conv      | 474 500 365 500 | 406 500 406 500 | 500 500 500 |
| \( \psi_1 \) | 4.86(0.15) | 4.74(0.14) | 4.75(0.14) |
| \( \psi_2 \) | 7.30(0.17) | 7.22(0.16) | 7.22(0.16) |
| \( \psi_3 \) | 6.03(0.16) | 6.01(0.15) | 6.01(0.15) |
| # Conv      | 480 500 398 500 | 378 500 378 500 | 500 500 500 |

The initial value of \( U \) is then extracted from the left singular vector matrix of \( \Gamma = (\hat{\psi}_1, \ldots, \hat{\psi}_n) \) where \( \hat{\psi}_n = (B_n^I B_n)^{-1} B_n^I y_n \).

The phenomenon is consistent with statements in Peng and Paul (2009) that REML is very sensitive to the choice of initial values and the unconvergence of REML mainly comes from unsuitable initial estimates. Unlike REML, the performance of our proposed CG method is stable across different initials with the highest accuracy for eigenfunction estimation in almost all cases. The improvement of CG over REML illustrates the benefit of the proposed conjugate gradient algorithm. The results in Table 2 are generally similar as those in Table 1. Since the true eigenfunctions have complex shapes under the pracSin setting, the sample size \( N = 100 \) is not sufficient for REML to work well. To be specific, with LS and LOC as initial values, REML has less than 100 converged replicates out of the total 500, and its estimation error is much larger than others. Our proposed CG outperforms the competitors and is more stable than REML and EM with different initial values. When the sample size increases to 500, the performance of all methods gets improved. However, CG remains superior to the others.

To compare the running efficiency between our algorithm with the traditional methods of EM and REML, we run all methods on the same computation platform with 2.40 GHz Intel (R) Xeon (R) E5-2680 v4 28-Core CPU using R. We record the CPU running time of each method for all settings to compare their computational efficiency. We depict the boxplots of the running time (in seconds) of each method.
Table 2  Simulation result for the pracSin setting. The reported values ($\times 10^{-1}$) are the average estimation errors for each eigenfunction $\| \hat{\psi}_r - \psi_0 \|_{L_2}$ based on 500 replications. The reported numbers are the actual values multiplied by 10. The numbers in the parentheses are the standard errors. The first row indicates the initialization methods of EM, LS, and LOC. The row ‘# Conv’ is the number of converged replicates.

| $N = 100$ | EM ($\times 10^{-1}$) | LS ($\times 10^{-1}$) | LOC ($\times 10^{-1}$) |
|-----------|-----------------------|-----------------------|-----------------------|
| $\nu \sim $ Normal | REML | CG | REML | EM | CG | REML | EM | CG |
| $\hat{\psi}_1$ | 6.80(0.15) | 4.53(0.10) | 11.39(0.11) | 5.00(0.12) | 4.56(0.10) | 11.72(0.11) | 5.17(0.12) | 4.60(0.10) |
| $\hat{\psi}_2$ | 9.32(0.14) | 7.59(0.13) | 12.19(0.08) | 8.06(0.14) | 7.62(0.13) | 12.16(0.08) | 8.12(0.14) | 7.67(0.13) |
| $\hat{\psi}_3$ | 10.23(0.12) | 9.20(0.14) | 12.51(0.06) | 9.64(0.13) | 9.18(0.13) | 12.28(0.07) | 9.59(0.13) | 9.24(0.13) |
| $\hat{\psi}_4$ | 10.37(0.12) | 9.61(0.13) | 12.65(0.06) | 9.76(0.13) | 9.67(0.13) | 12.54(0.07) | 9.92(0.13) | 9.65(0.13) |
| $\hat{\psi}_5$ | 10.24(0.13) | 8.17(0.14) | 12.50(0.06) | 8.69(0.15) | 8.24(0.14) | 12.51(0.07) | 8.96(0.15) | 8.28(0.14) |
| # Conv | 434 | 500 | 62 | 500 | 500 | 66 | 499 | 500 |
| $\nu \sim t_3$ | REML | CG | REML | EM | CG | REML | EM | CG |
| $\hat{\psi}_1$ | 6.90(0.15) | 4.74(0.10) | 11.41(0.12) | 5.26(0.12) | 4.77(0.10) | 11.42(0.11) | 5.34(0.13) | 4.82(0.11) |
| $\hat{\psi}_2$ | 9.41(0.13) | 7.60(0.14) | 12.23(0.08) | 7.93(0.14) | 7.67(0.14) | 12.14(0.08) | 7.94(0.14) | 7.68(0.14) |
| $\hat{\psi}_3$ | 10.39(0.12) | 9.33(0.13) | 12.37(0.07) | 9.54(0.13) | 9.41(0.13) | 12.29(0.07) | 9.55(0.13) | 9.44(0.13) |
| $\hat{\psi}_4$ | 10.52(0.12) | 9.51(0.13) | 12.56(0.06) | 9.83(0.13) | 9.53(0.13) | 12.48(0.06) | 9.90(0.13) | 9.62(0.13) |
| $\hat{\psi}_5$ | 10.43(0.13) | 8.48(0.14) | 12.54(0.06) | 8.82(0.14) | 8.52(0.14) | 12.54(0.06) | 8.94(0.15) | 8.56(0.14) |
| # Conv | 437 | 500 | 57 | 500 | 500 | 77 | 500 | 500 |
| $\nu \sim $ Unif | REML | CG | REML | EM | CG | REML | EM | CG |
| $\hat{\psi}_1$ | 6.70(0.15) | 4.67(0.11) | 11.46(0.11) | 5.15(0.13) | 4.64(0.11) | 11.50(0.11) | 5.29(0.13) | 4.66(0.11) |
| $\hat{\psi}_2$ | 9.07(0.14) | 7.60(0.14) | 12.18(0.08) | 7.91(0.14) | 7.59(0.14) | 12.18(0.08) | 7.99(0.13) | 7.62(0.13) |
| $\hat{\psi}_3$ | 10.16(0.12) | 9.20(0.14) | 12.47(0.07) | 9.54(0.13) | 9.20(0.13) | 12.20(0.08) | 9.69(0.13) | 9.20(0.13) |
| $\hat{\psi}_4$ | 10.17(0.13) | 9.56(0.14) | 12.54(0.06) | 9.85(0.13) | 9.54(0.14) | 12.35(0.07) | 9.89(0.13) | 9.56(0.14) |
| $\hat{\psi}_5$ | 10.35(0.13) | 8.41(0.14) | 12.42(0.06) | 8.86(0.14) | 8.40(0.14) | 12.52(0.06) | 8.90(0.14) | 8.47(0.14) |
| # Conv | 451 | 500 | 59 | 500 | 500 | 73 | 496 | 500 |

Fig. 2  The boxplot of running time of each method with different initial values for the easySin setting under Gaussian noises. The results are based on 500 simulation replications, and the $y$-axis is plotted in log scale among 500 repetitions with normal distributed noises. Figures 2–4 correspond to the easySin setting with $N = 50$, the pracSin setting with $N = 100$, and the pracSin setting with $N = 500$, respectively. The results of the running time when the noises are rescaled student $t$ distributed and uniform distributed are similar. These plots evidently show that the running time of the proposed CG method is less than REML and EM on average. For example, the average running time of the REML is above 30 seconds for the pracSin setting.
Table 2 continued

| $\psi$ | $N = 500$ | $\varepsilon \sim $ Normal | $\varepsilon \sim $ Normal | $\varepsilon \sim $ Normal | $\varepsilon \sim $ Normal |
|-------|-----------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|
| $\psi_1$ | 1.88(0.03) | 1.90(0.03) | 6.71(0.22) | 2.67(0.10) | 1.90(0.03) | 8.04(0.22) | 2.93(0.10) | 1.90(0.03) |
| $\psi_2$ | 3.27(0.07) | 3.28(0.07) | 7.95(0.19) | 4.20(0.13) | 3.28(0.07) | 9.17(0.19) | 4.48(0.13) | 3.28(0.07) |
| $\psi_3$ | 5.03(0.12) | 5.01(0.12) | 9.06(0.18) | 5.88(0.15) | 5.01(0.12) | 9.86(0.16) | 6.01(0.14) | 5.01(0.11) |
| $\psi_4$ | 6.12(0.14) | 6.17(0.14) | 9.48(0.17) | 6.98(0.16) | 6.18(0.14) | 10.35(0.15) | 7.03(0.15) | 6.18(0.14) |
| $\psi_5$ | 4.89(0.13) | 4.94(0.13) | 9.07(0.18) | 6.05(0.17) | 4.95(0.13) | 9.88(0.17) | 6.07(0.17) | 4.96(0.13) |
| $\# Conv$ | 486 | 500 | 248 | 500 | 500 | 221 | 500 | 500 |

$\varepsilon \sim $ Normal

| $\psi$ | $N = 100$ | $\varepsilon \sim $ Normal | $\varepsilon \sim $ Normal | $\varepsilon \sim $ Normal |
|-------|-----------|-----------------------------|-----------------------------|-----------------------------|
| $\psi_1$ | 1.87(0.03) | 1.89(0.03) | 7.26(0.22) | 2.60(0.10) | 1.89(0.03) | 7.98(0.21) | 2.85(0.10) | 1.89(0.03) |
| $\psi_2$ | 3.38(0.08) | 3.36(0.07) | 8.43(0.20) | 4.22(0.12) | 3.36(0.07) | 9.23(0.19) | 4.54(0.12) | 3.36(0.07) |
| $\psi_3$ | 5.14(0.12) | 5.11(0.12) | 9.30(0.18) | 5.85(0.14) | 5.07(0.12) | 10.35(0.15) | 7.03(0.15) | 5.08(0.12) |
| $\psi_4$ | 6.21(0.14) | 6.10(0.14) | 9.97(0.16) | 6.81(0.15) | 6.06(0.14) | 10.39(0.15) | 7.02(0.16) | 6.07(0.14) |
| $\psi_5$ | 5.01(0.13) | 4.94(0.13) | 9.33(0.18) | 5.86(0.17) | 4.92(0.13) | 10.15(0.16) | 5.99(0.17) | 4.93(0.13) |
| $\# Conv$ | 491 | 500 | 245 | 500 | 500 | 224 | 500 | 500 |

$\varepsilon \sim $ Normal

| $\psi$ | $N = 100$ | $\varepsilon \sim $ Normal | $\varepsilon \sim $ Normal | $\varepsilon \sim $ Normal |
|-------|-----------|-----------------------------|-----------------------------|-----------------------------|
| $\psi_1$ | 1.86(0.03) | 1.88(0.03) | 7.68(0.22) | 2.67(0.10) | 1.88(0.03) | 8.08(0.22) | 2.89(0.10) | 1.88(0.03) |
| $\psi_2$ | 3.31(0.08) | 3.32(0.07) | 8.12(0.20) | 4.18(0.13) | 3.32(0.07) | 9.13(0.19) | 4.47(0.13) | 3.32(0.07) |
| $\psi_3$ | 5.09(0.12) | 5.01(0.12) | 8.99(0.18) | 5.77(0.15) | 5.01(0.12) | 9.90(0.16) | 6.03(0.14) | 5.02(0.12) |
| $\psi_4$ | 6.21(0.14) | 6.18(0.14) | 9.53(0.17) | 6.81(0.16) | 6.17(0.14) | 10.31(0.15) | 6.98(0.16) | 6.18(0.14) |
| $\psi_5$ | 4.91(0.13) | 5.00(0.13) | 8.81(0.19) | 5.94(0.17) | 4.99(0.13) | 10.07(0.17) | 6.04(0.17) | 5.00(0.13) |
| $\# Conv$ | 478 | 500 | 260 | 500 | 500 | 217 | 500 | 500 |

Fig. 4 The boxplot of running time of each method with different initial values for the pracSin setting of sample size $N = 500$ under Gaussian noises. The results are based on 500 simulation replications, and the $y$-axis is plotted in log 10 scale.

5.2 Comparison between knots selection and roughness penalty

All the above methods are tuned by selecting the number of knots $K$ in the domain. In this subsection, we consider an alternative approach of penalized spline (O’Sullivan 1986, 1988; Huang and Su 2021), which fixes the number of knots at a relatively large value and a roughness penalization is introduced to trade off bias and variance. A large value of $K$ allows us to approximate more complex functions; meanwhile, the variance is controlled by the roughness penalty.

with $N = 500$; whereas the proposed CG takes less than 3 seconds on average for the same setting.

We also explore two more settings used in Peng and Paul (2009) which particularly favor REML. The two settings are denoted as easy and prac. For the former, there are 3 eigenvalues: 1, 0.66, and 0.517; and the eigenfunctions are directly generated by the cubic B-spline with 5 equally spaced knots. The latter is more complex with 10 equally spaced knots and 5 eigenvalues: 1, 0.66, 0.517, 0.435, and 0.381. The noises also follow Gaussian, rescaled student t, and uniform distributions. For both settings, we fix $R = R_0$ for all methods. The results for these two settings are presented in Tables 3 and 4, respectively. It shows that the performance of REML becomes competitive under these settings when the output of EM is used as the initial value, and the sample size is large. However, even in these cases, the performance of the proposed CG is close to that of REML. For other cases, the CG remains to outperform REML and EM in terms of estimation accuracy and stability of initials.
Simulation result for the *easy* setting. The reported values ($\times 10^{-1}$) are the average estimation errors for each eigenfunction $\|\hat{\psi} - \psi_0\|_{L_2}$ based on 500 replications. The reported numbers are the actual values multiplied by 10. The numbers in the parentheses are the standard errors. The first row indicates the initialization methods of EM, LS, and LOC. The row '# Conv' is the number of converged replicates.

| $\varepsilon \sim \text{Normal}$ | EM ($\times 10^{-1}$) | LS ($\times 10^{-1}$) | LOC ($\times 10^{-1}$) |
|-------------------------------|-----------------|-----------------|-----------------|
| $\hat{\psi}_1$                | 5.06(0.14)      | 5.07(0.14)      | 5.07(0.14)      |
| $\hat{\psi}_2$                | 6.96(0.16)      | 7.06(0.15)      | 7.05(0.16)      |
| $\hat{\psi}_3$                | 5.56(0.14)      | 5.81(0.14)      | 5.80(0.14)      |
| # Conv                         | 494             | 500             | 500             |

Simulation result for the *prac* setting. The reported values ($\times 10^{-1}$) are the average estimation errors for each eigenfunction $\|\hat{\psi} - \psi_0\|_{L_2}$ based on 500 replications. The reported values are the actual values multiplied by 10. The numbers in the parentheses are the standard errors. The first row indicates the initialization methods of EM, LS, and LOC. The row '# Conv' is the number of converged replicates.

| $\varepsilon \sim \text{Normal}$ | EM ($\times 10^{-1}$) | LS ($\times 10^{-1}$) | LOC ($\times 10^{-1}$) |
|-------------------------------|-----------------|-----------------|-----------------|
| $\hat{\psi}_1$                | 7.37(0.19)      | 4.88(0.12)      | 4.83(0.11)      |
| $\hat{\psi}_2$                | 9.20(0.15)      | 7.73(0.14)      | 7.69(0.13)      |
| $\hat{\psi}_3$                | 10.28(0.13)     | 9.51(0.13)      | 9.50(0.13)      |
| # Conv                         | 403             | 500             | 500             |

| $\varepsilon \sim \text{Unif}$ | EM ($\times 10^{-1}$) | LS ($\times 10^{-1}$) | LOC ($\times 10^{-1}$) |
|-------------------------------|-----------------|-----------------|-----------------|
| $\hat{\psi}_1$                | 7.40(0.19)      | 4.85(0.12)      | 4.82(0.12)      |
| $\hat{\psi}_2$                | 9.46(0.16)      | 7.82(0.14)      | 7.78(0.14)      |
| $\hat{\psi}_3$                | 10.12(0.13)     | 9.45(0.13)      | 9.49(0.13)      |
| # Conv                         | 403             | 500             | 500             |

| $\varepsilon \sim \text{Unif}$ | EM ($\times 10^{-1}$) | LS ($\times 10^{-1}$) | LOC ($\times 10^{-1}$) |
|-------------------------------|-----------------|-----------------|-----------------|
| $\hat{\psi}_1$                | 7.02(0.19)      | 4.74(0.11)      | 4.73(0.11)      |
| $\hat{\psi}_2$                | 9.10(0.15)      | 7.66(0.13)      | 7.67(0.13)      |
| $\hat{\psi}_3$                | 9.93(0.13)      | 9.47(0.13)      | 9.56(0.13)      |
| # Conv                         | 413             | 500             | 500             |
The penalty measures the roughness of a function through the integral of its squared $d$-th ($1 \leq d \leq O + 1$) derivative, i.e.,

$$
P_\eta(U) := \eta \sum_{r=1}^{R} \int \left\{ \psi_r^{(d)}(u) \right\}^2 du = \eta \text{tr}(U^T \Gamma U),\tag{18}
$$

where $\Gamma = \int b^{(d)}(u) (b^{(d)}(u))^T du$ is a $K \times K$ matrix solely depending on the basis functions and $\eta$ is the penalty parameter. Note this penalty on the eigenfunctions can be transferred to the optimization parameter $U$ and thus can be incorporated into our method directly. More precisely, combining the loss (8) and the penalty (18), the objective function becomes

$$
\ell_\eta(U, W, \sigma_r^2) := L(U, W, \sigma_r^2) + P_\eta(U, W).\nonumber
$$

We get our estimator from the $N$ sparse samples via solving

$$
\min_{U, W, \sigma_r^2} \ell_\eta(U, W, \sigma_r^2),
$$

subject to $U \in \text{St}(R, K), \ W = \text{diag}(\lambda_1, \cdots, \lambda_K) > 0,$

and $\sigma_r^2 > 0$.

The above optimization problem can still be solved by the proposed conjugate gradient algorithm over the product manifold, and we denote the corresponding estimator as CG-p in the following. In this numerical experiment, we fix the number of knots as 15 and select optimal $\eta$ by CV.

The two approaches (CG and CG-p) are compared over a simulation setting named pracNU, where a non-linear transformation is applied to the eigenfunctions $\psi_r$ in the prac setting of Peng and Paul (2009). Specifically, our experiment uses the eigenfunction $\tilde{\psi}_r(u) = \psi_r(F(u; \alpha, \beta))$ for $r = 1, \ldots, 5$, where $F(u; \alpha, \beta)$ is the cumulative distribution function of beta distribution (the regularized incomplete beta function) and the parameters are specified as $\alpha = \beta = 2$. After the non-linear transformation, the shape of eigenfunction becomes more complicated, and it’s interesting to compare the performance of discrete tuning (the number of knots) and continuous tuning (roughness penalization). The results of this simulation study under various error distributions and sample sizes are summarized in Table 5.

Table 5 shows some advantages of CG-p over CG. When the sample size is 100, the estimation errors of functional principal components have been reduced when the roughness penalization is used. However, when the sample size is relatively large ($N = 500$), the two methods have competitive performance. We thus recommend using CG-p when the sample size is relatively small. Note that CG-p could possibly
Table 5  Estimation result for the *pracNU* setting. Our proposed method is tuned by selecting the number of knots (CG) or selecting the roughness tuning parameter (CG-p) with 15 fixed number of knots. The reported values (×10⁻¹) are the average estimation errors for each eigenfunction $\| \hat{\psi}_i - \psi_0 \|_{L_2}$ based on 500 replications. The reported numbers are the actual values multiplied by 10. The numbers in the parentheses are the standard errors. The first row indicates the initialization methods of EM, LS, and LOC.

| $N = 100$ | EM (×10⁻¹) | LS (×10⁻¹) | LOC (×10⁻¹) |
|------------|-------------|-------------|-------------|
| $\varepsilon \sim \text{Normal}$ | | | |
| $\hat{\psi}_1$ | 5.91(0.14) | 5.76(0.14) | 5.64(0.13) |
| $\hat{\psi}_2$ | 8.77(0.13) | 7.67(0.13) | 7.83(0.14) |
| $\hat{\psi}_3$ | 9.66(0.13) | 8.77(0.14) | 8.96(0.14) |
| $\hat{\psi}_4$ | 9.72(0.13) | 9.30(0.14) | 9.28(0.13) |
| $\hat{\psi}_5$ | 8.86(0.14) | 8.40(0.14) | 8.39(0.14) |
| $\varepsilon \sim t_3$ | | | |
| $\hat{\psi}_1$ | 5.70(0.12) | 5.29(0.12) | 5.37(0.12) |
| $\hat{\psi}_2$ | 8.91(0.13) | 7.52(0.13) | 7.86(0.13) |
| $\hat{\psi}_3$ | 10.06(0.12) | 9.15(0.13) | 9.19(0.13) |
| $\hat{\psi}_4$ | 9.84(0.13) | 9.44(0.13) | 9.55(0.13) |
| $\hat{\psi}_5$ | 8.95(0.14) | 8.65(0.14) | 8.74(0.15) |
| $\varepsilon \sim \text{Unif}$ | | | |
| $\hat{\psi}_1$ | 5.66(0.13) | 5.44(0.13) | 5.31(0.12) |
| $\hat{\psi}_2$ | 8.90(0.13) | 7.89(0.13) | 7.84(0.13) |
| $\hat{\psi}_3$ | 10.02(0.13) | 9.05(0.14) | 9.10(0.14) |
| $\hat{\psi}_4$ | 9.84(0.13) | 9.40(0.13) | 9.37(0.14) |
| $\hat{\psi}_5$ | 9.19(0.14) | 8.88(0.15) | 8.80(0.14) |

$N = 500$

| $\varepsilon \sim \text{Normal}$ | | | |
| $\hat{\psi}_1$ | 2.12(0.05) | 1.99(0.04) | 2.10(0.05) |
| $\hat{\psi}_2$ | 3.87(0.09) | 3.55(0.08) | 3.85(0.09) |
| $\hat{\psi}_3$ | 5.34(0.12) | 5.19(0.12) | 5.39(0.12) |
| $\hat{\psi}_4$ | 6.16(0.13) | 6.30(0.14) | 6.20(0.14) |
| $\hat{\psi}_5$ | 4.82(0.13) | 4.99(0.13) | 4.83(0.13) |
| $\varepsilon \sim t_3$ | | | |
| $\hat{\psi}_1$ | 1.99(0.04) | 1.95(0.04) | 1.96(0.04) |
| $\hat{\psi}_2$ | 3.77(0.08) | 3.59(0.08) | 3.76(0.08) |
| $\hat{\psi}_3$ | 5.46(0.12) | 5.46(0.12) | 5.53(0.12) |
| $\hat{\psi}_4$ | 6.29(0.14) | 6.35(0.14) | 6.30(0.14) |
| $\hat{\psi}_5$ | 4.96(0.13) | 5.11(0.13) | 4.96(0.13) |
| $\varepsilon \sim \text{Unif}$ | | | |
| $\hat{\psi}_1$ | 1.98(0.03) | 1.92(0.04) | 1.97(0.03) |
| $\hat{\psi}_2$ | 3.78(0.08) | 3.52(0.07) | 3.80(0.08) |
| $\hat{\psi}_3$ | 5.20(0.12) | 5.13(0.12) | 5.29(0.12) |
| $\hat{\psi}_4$ | 6.13(0.14) | 6.20(0.14) | 6.18(0.14) |
| $\hat{\psi}_5$ | 4.94(0.13) | 5.03(0.13) | 4.96(0.13) |
be improved further by assigning different tuning parameters for each principal component function.

5.3 Real data analysis

We further compare these methods on the Type Ia supernova light curve data obtained from the Carnegie Supernova Project (CSP, Stritzinger et al. 2011). Type Ia supernova (SNe) is an important “standardizing” candle for us to probe the universe and measure the distance. In the CSP dataset, each supernova is observed by nine bands (or filters), which only allow the light within a certain wavelength range to pass through. Different bands are usually denoted by distinct letters (u, g, r, B, V, etc.). Astronomers typically use the peak magnitude (brightness) to measure distances.

We only selected the SNe samples which had measurement before the peak B-band magnitude. This allowed us to perform curve registration and align the peak magnitude to the origin point of the plane. Specifically, a smooth spline function was fitted to the B-band observation of each sample, and the peak magnitude time was identified from the fitted function. The observation time was then subtracted by the peak magnitude time such that the peak magnitude got registered to time 0. After the curve registration, we only kept the observation points within −10 to +40 days around the peak. The range of observational time points was then scaled to the [0, 1]. We treated the light curves from different bands as independent samples. The total number of resulting sample curves is \( N = 258 \). The registered and scaled light curves of one supernova are depicted in Fig. 5, where various colored points represent sparse observations from distinct bands. Before using FPCA to fit the data, a common mean curve was firstly estimated and subtracted from the data. The fitted mean curve, denoted by \( \mu_0 \), is also shown in Fig. 5 as a solid curve.

The proposed CG and the alternative methods discussed in the previous section, i.e., EM and REML, were applied to the pre-processed data to estimate the first three principal component functions. The methods were all tuned by CV to select the number of knots. The estimation results of the first 3 eigenfunctions by CG, EM, and REML are depicted in Fig. 6. It shows that the estimated eigenfunctions by all methods roughly follow the same pattern.

Since the true eigenfunctions are unknown in the real data, we accessed the prediction accuracy to compare different methods. In this part, we also included CG-p, the conjugate gradient method with roughness penalty, as a competitor. The dataset got randomly split into two subsets of curves, i.e., a training set and a test set. They account for 80% and 20% of the total curves, respectively. On the training dataset, \( R(= 2, 3, 4) \) principal components were estimated by each FPCA method. Let \( \hat{\psi}_r \)'s and \( \hat{\lambda}_r \)'s be respectively the estimated eigenfunctions and eigenvalues, \( r = 1, \ldots, 4 \), and \( \hat{\sigma}_e^2 \) be the estimated noise variance.

The prediction accuracy was evaluated on the test dataset as follows. For each light curve in the test set, we randomly selected 50% of its observational points to predict its scores and fitted the light curve. The remaining points were set as hold-out data. In other words, for the \( n \)-th sample with \( M_n \) points, the index set \( \{ 1, \ldots, M_n \} = O_n \cup H_n \) was randomly split into two disjoint subsets \( O_n \) and \( H_n \) of roughly the same size. The two sets correspond to the observed part and the held-out part, respectively. Suppose we have observed \( y_{ns} = (y_{nj})_{j \in O_n} \) at the time points \( \{ u_{nj} \}_{j \in O_n} \). The corresponding mean vector is denoted as \( \hat{\mu}_{ns} = \{ \hat{\mu}(u_{nj}) \}_{j \in O_n}^T \), with \( \hat{\mu} \) being the estimated mean function \( \mu \) in the data pre-processing step. Define \( \Psi_{ns} = [\psi_r(u_{nj})]_{j \in O_n,r} \) as a matrix of size \( |O_n| \times R \). The \( r \)-th column of \( \Psi_{ns} \) contains the evaluation of the \( r \)-th eigenfunction at the observed time points, \( r = 1, \ldots, R \).

Let \( \theta_n \) be the PC scores of \( n \)-th sample in the \( r \)-th component. Denote \( \theta_n = (\theta_{n1}, \ldots, \theta_{nR})^T \). We can see the joint vector \( (y_{ns}^T, \theta_n^T)^T \) of the observations and the scores has the following first and second moments

\[
E \left( \begin{array}{c} y_{ns} \\ \theta_n \end{array} \right) = \left( \begin{array}{c} \mu_{ns} \\ 0 \end{array} \right), \quad \text{Var} \left( \begin{array}{c} y_{ns} \\ \theta_n \end{array} \right) = \begin{pmatrix} \Psi_{ns} W \Psi_{ns}^T + \sigma_e^2 I & \Psi_{ns} W \\ W \Psi_{ns}^T & W \end{pmatrix}.
\]

(19)
Fig. 6 The three panels from left to right correspond to the leading three estimated eigenfunctions by CG, EM, and REML.

Fig. 7 Two examples of curve fitting by the three methods. The red dots are the observed points in the test curves that we used to predict the PC scores. The yellow dots are held-out data points for which we test the prediction performance. The predicted light curves by the proposed CG method are shown as blue solid lines in the center. The corresponding 95% predictive intervals by CG are shown as grey areas. The predicted curves by EM and REML are shown in green short dashed and purple long dashed, respectively.
Plug-in the estimated $\hat{\mu}_{n*}$, $\hat{\sigma}^2_n$, $\hat{W}$, and $\hat{\Psi}_{n*}$, (19) leads us to the best linear unbiased prediction (BLUP) of the score $\theta_n$ as

$$\hat{\theta}_n = \hat{W}^{\dagger} (\hat{\Psi}_{n*} \hat{W}^{\dagger} T + \hat{\sigma}^2_n I)^{-1} (y_{n*} - \hat{\mu}_{n*}).$$

Given the predicted score $\hat{\theta}_n$, the mean squared prediction errors (MSPE) over the held-out observations indexed by $H_n$ is evaluated as

$$\text{MSPE} = \frac{1}{|H_n|} \sum_{j \in H_n} \left\{ y_{nj} - \hat{\mu}(u_{nj}) - \sum_{r=1}^R \lambda_r^{1/2} \hat{\theta}_{nr} \psi_r(u_{nj}) \right\}^2.$$  

Figure 7 illustrates this predicting procedure for two curves in the test dataset. It plots the fitted curve by using $R = 4$ PPCs estimated by CG over the training dataset in one random splitting. The red and orange points correspond to the observed (indexed by $O_n$) and held-out (indexed by $H_n$) data points, respectively. The solid blue lines are the fitted curves based on the red (observed) points, and then MSPE is evaluated for the held-out data points in orange. The gray area represents the 95% point-wise predictive interval. The predicted curves by EM (green short dashed) and REML (purple long dashed) are included for comparison.

We carried out 500 random splitting of the above procedure. In each replication, the average MSPE using each method under consideration (among EM, REML, CG, and CG-p) was computed for all the held-out samples in the test dataset. We tested various numbers $R(=2, 3, 4)$ of fitted functional principal components. Table 6 reports the average MSPE under these random splittings with the corresponding standard errors. The results in Table 6 are consistent with those in the simulation studies. In particular, Table 6 shows that REML leads to a large prediction error when the initial value is not good enough. Our methods (CG and CG-p) enjoy the best prediction accuracy compared with the other methods in consideration. CG-p with a penalty also shows some improvement over CG.

### 6 Discussion

This paper has proposed a manifold conjugate gradient algorithm to estimate the functional principal components when the sample curves are sparse and irregularly observed. Using basis expansion, we have formulated the objective function on the product space of the Stiefel manifold and the cone of positive definite matrices. The proposed method improves the search efficiency and estimation accuracy by taking into account the intrinsic geometric structure of the underlying product manifold. Our objective function is directly derived from a matrix Bregman divergence, and thus no distribution assumption is needed to apply the proposed method. We also have shown that a roughness penalization can be easily incorporated into our algorithm with a better fit. The superior performance and numerical stability of the proposed method against the existing ones are demonstrated by simulation studies and a real application on Type Ia supernova light curve data.

Our manifold conjugate gradient algorithm can be extended to general loss functions by choosing different seed functions in the matrix Bregman divergence. A comprehensive study on the comparison of different loss functions for functional principal component analysis is a potential future research topic. Furthermore, investigating the asymptotic properties of these estimators is also of interest.

### Acknowledgements

The authors thank the editor, the associate editor, and the anonymous reviewers for their comments that helped significantly improve this work. This research was supported by the National Key R&D Program of China (Grant No. 2018YFC2000302) and by Public Computing Cloud, Renmin University of China. The research of Shiyuan He was partially supported by National Natural Science Foundation of China (No.11801561). The research of Kejun He was
References

Absil, P.-A., Mahony, R., Sepulchre, R.: Optimization Algorithms on Matrix Manifolds. Princeton University Press, Princeton, New Jersey (2009)

Bertsekas, D.P.: Nonlinear Programming. Athena Scientific, Belmont (1999)

Cai, T.T., Yuan, M.: Nonparametric Covariance Function Estimation for Functional and Longitudinal Data. University of Pennsylvania, Philadelphia (2010)

Chiou, J., Li, P.: Functional clustering and identifying substructures of longitudinal data. J. Royal Stat. Soc. Series B (Stat. Methodol.) 69(4), 679–699 (2007)

Dhillon, I.S., Tropp, J.A.: Matrix nearness problems with bregman divergences. SIAM J. Matrix Anal. Appl. 29(4), 1120–1146 (2008)

Edelman, A., Arias, T.A., Smith, S.T.: The geometry of algorithms with orthogonality constraints. SIAM J. Matrix Anal. Appl. 20(2), 303–353 (1998)

Fletcher, R., Reeves, C.M.: Function minimization by conjugate gradients. Comput. J. 7(2), 149–154 (1964)

Hall, P., Müller, H.-G., Wang, J.-L.: Properties of principal component methods for functional and longitudinal data analysis. Ann. Stat. 34(3), 1493–1517 (2006)

Hu, J., Liu, X., Wen, Z.-W., Yuan, Y.-X.: A brief introduction to manifold optimization. J. Operat. Res. Soc. Chin. 8(2), 199–248 (2020)

Huang, J.Z., Su, Y.: Asymptotic properties of penalized spline estimators in concave extended linear models: Rates of convergence. Ann. Stat. 49(6), 3383–3407 (2021)

James, G.M., Hastie, T.J.: Functional linear discriminant analysis for irregularly sampled curves. J. Royal Stat. Soc. Series B (Stat. Methodol.) 63(3), 533–550 (2001)

James, G.M., Sugar, C.A.: Clustering for sparsely sampled functional data. J. Am. Stat. Assoc. 98(462), 397–408 (2003)

James, G.M., Hastie, T.J., Sugar, C.A.: Principal component models for sparse functional data. Biometrika 87(3), 587–602 (2000)

Karhunen, K.: Zur spektraltheorie stochastischer prozesse. Ann. Academiae Scientiarum Fennicae, Series A I Math. 34, 1–7 (1946)

Li, Y., Hsing, T.: Uniform convergence rates for nonparametric regression and principal component analysis in functional/longitudinal data. Ann. Stat. 38(6), 3321–3351 (2010)

Loève, M.: Fonctions aléatoires à décomposition orthogonale exponentielle. La Revue Scientifique 84, 159–162 (1946)

Meyer, G.: Geometric optimization algorithms for linear regression on fixed-rank matrices. PhD thesis, Universite de Liege, Belgique (2011)

Nocedal, J., Wright, S.: Numerical Optimization. Springer, New York (2006)

O’Sullivan, F.: A statistical perspective on ill-posed inverse problems. Stat. Sci. 1(4), 502–518 (1986)

O’Sullivan, F.: Fast computation of fully automated log-density and log-hazard estimators. SIAM J. Sci. Stat. Comput. 9(2), 363–379 (1988)

Paul, D., Peng, J.: Consistency of restricted maximum likelihood estimators of principal components. Ann. Stat. 37(3), 1229–1271 (2009)

Peng, J., Paul, D.: Fpca: Restricted MLE for Functional Principal Components Analysis. (2011). R package version 0.2-1. https://CRAN.R-project.org/package=fpca

Peng, J., Paul, D.: A geometric approach to maximum likelihood estimation of the functional principal components from sparse longitudinal data. J. Comput. Graph. Stat. 18(4), 995–1015 (2009)

Polak, E.: Computational Methods in Optimization: a Unified Approach, vol. 77. Academic press, New York (1971)

Ramsay, J.O., Dalzell, C.J.: Some tools for functional data analysis. J. Royal Stat. Soc. Series B Methodol. 53(3), 539–572 (1991)

Ramsay, J.O., Silverman, B.W.: Applied Functional Data Analysis?: Methods and Case Studies. Springer, New York (2002)

Ramsay, J.O., Silverman, B.W.: Functional Data Analysis, 2nd edn. Springer, New York (2005)

Rice, J.A., Silverman, B.W.: Estimating the mean and covariance structure nonparametrically when the data are curves. J. Roy. Stat. Soc. Ser. B (Methodol.) 51(3), 233–243 (1991)

Ring, W., Wirth, B.: Optimization methods on riemannian manifolds and their application to shape space. SIAM J. Optim. 22(2), 596–627 (2012)

Silverman, B.W.: Smoothed functional principal components analysis by choice of norm. Ann. Stat. 24(1), 1–24 (1996)

Smith, S.T.: Covariance, subspace, and intrinsic cramer/spl acute/r-rao bounds. IEEE Trans. Signal Process. 53(5), 1610–1630 (2005)

Sra, S., Hosseini, R.: Conic geometric optimization on the manifold of positive definite matrices. SIAM J. Optim. 25(1), 713–739 (2015)

Stritzerger, M.D., Phillips, M., Boldt, L.N., Burns, C., Campillay, A., Contreras, C., Gonzalez, S., Fortelini, G., Morrell, N., Krzeminski, W.: The carnegie supernova project: second photometry data release of low-redshift type ia supernovae. Astron. J. 142(5), 156 (2011)

Yao, F., Müller, H.-G., Wang, J.-L.: Functional data analysis for sparse longitudinal data. J. Am. Stat. Assoc. 100(470), 577–590 (2005)

Yao, F., Mueller, H.G., Wang, J.L.: Functional linear regression analysis for longitudinal data. Ann. Stat. 33(6), 2873–2903 (2005)

Zhang, X., Wang, J.-L.: From sparse to dense functional data and beyond. Ann. Stat. 44(5), 2281–2321 (2016)

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.