Multi-dimensional functional principal component analysis

Lu-Hung Chen · Ci-Ren Jiang

Received: 1 November 2015 / Accepted: 18 June 2016 / Published online: 29 June 2016
© Springer Science+Business Media New York 2016

Abstract Functional principal component analysis is one of the most commonly employed approaches in functional and longitudinal data analysis and we extend it to analyze functional/longitudinal data observed on a general $d$-dimensional domain. The computational issues emerging in the extension are fully addressed with our proposed solutions. The local linear smoothing technique is employed to perform estimation because of its capabilities of performing large-scale smoothing and of handling data with different sampling schemes (possibly on irregular domain) in addition to its nice theoretical properties. Besides taking the fast Fourier transform strategy in smoothing, the modern GPGPU (general-purpose computing on graphics processing units) architecture is applied to perform parallel computation to save computation time. To resolve the out-of-memory issue due to large-scale data, the random projection procedure is applied in the eigen-decomposition step. We show that the proposed estimators can achieve the classical nonparametric rates for longitudinal data and the optimal convergence rates for functional data if the number of observations per sample is of the order $(n/\log n)^{d/4}$. Finally, the performance of our approach is demonstrated with simulation studies and the fine particulate matter (PM 2.5) data measured in Taiwan.

Keywords Fast Fourier transform · Functional and longitudinal data · GPU-parallelization · Local linear smoother · PM 2.5 data · Random projection

1 Introduction

Functional principal component analysis (FPCA), inherited from principal component analysis (PCA), has been widely employed in various fields of functional data analysis (FDA) and longitudinal data analysis; e.g., functional regression (Yao et al. 2005b; Gertheiss et al. 2013), functional classification (Leng and Müller 2006; Chiou 2012), functional clustering (Chiou and Li 2007, 2008), outlier detection (Gervini 2009), survival analysis (Chiou and Müller 2009), time series analysis (Hyndman and Shang 2009; Aue et al. 2015), etc. Its applications also cover a wide range of topics (e.g., Leng and Müller 2006; Chiou and Müller 2009; Liu and Müller 2009; Aston et al. 2010; Chiu 2012; Jiang et al. 2016, etc.), and various estimation methods have been proposed for FPCA: Rice and Silverman (1991) and Silverman (1996) represented the latent functions with B-splines and extracted the eigenfunctions by performing generalized eigen-decomposition owing to the roughness penalty; James et al. (2000) and Rice and Wu (2001) adopted the mixed effect models where the mean function and the eigenfunctions were represented with B-splines and the spline coefficients were estimated by the EM algorithm; Yao et al. (2005a) applied the local linear smoothers (Fan and Gijbels 1996) to estimate the mean and the covariance functions and obtained the eigenfunctions by solving the corresponding eigen-equations. The principal component scores were predicted by the conditional expectation approach, named “PACE”. Some computational tricks for densely observed functional data can be found in (Chen et al. 2015) to accelerate the computation.
Even though FDA has received considerable attention over the last decade, most approaches still focus on one-dimensional functional data. Few are developed for general $d$-dimensional functional data. To extend FPCA to its multidimensional version, in practice one has to take good care of the following issues. **Issue A** is that the conventional strategy for obtaining eigenfunctions needs to be modified as the covariance of $d$-dimensional functional data becomes a $2d$-dimensional function instead of a matrix. **Issue B** is the enormous size of an empirical covariance function and applying any computation to an object of this scale is demanding. The last issue, **Issue C**, is that the $d$-dimensional domain may not be rectangle which could cause challenging problems in many smoothing approaches.

Recently, some attention has been drawn to spatial and image data in the Statistics society (e.g., Zhu et al. 2007; Aston and Kirch 2012; Tian et al. 2012; Zhang et al. 2013; Risk et al. 2014, etc.). In particular, several attempts have been made to extend FPCA for spatial and image data. Zipunnikov et al. (2011a,b) vectorized the pre-smoothed images and extracted the eigenfunctions by singular value decomposition (SVD). Wang and Huang (2016) proposed a regularized SVD approach for two-dimensional spatial data. However, SVD could only be applied to functional data observed on a regular grid. To handle bivariate functional observations made on irregular domain, Zhou and Pan (2014) extended the functional mixed effect models (James et al. 2000; Rice and Wu 2001) to bivariate functional data by utilizing the triangulated bivariate splines (Chui and Lai 1987; Lai and Wang 2013). However, the triangularized bivariate splines are designed for two-dimensional functions only. Extending spline basis functions for general $d$-dimensional data observed on an irregular domain is very sophisticated and becomes extremely complex as $d$ increases.

Hung et al. (2012) applied the multilinear PCA (MPCA) (Lu et al. 2008) to analyze two-dimensional facial images. MPCA requires less computational resources and is much faster than general $d$-dimensional FPCA, referred as d-FPCA hereafter. However, the eigentensors obtained by MPCA are usually uninterpretable. Moreover, MPCA is not applicable if the $d$-dimensional functions are not observed on rectangle meshes.

We will employ the local linear smoothing technique to perform estimation in d-FPCA in this paper because of its nice theoretical properties (Hall and Hosseini-Nasab 2006; Hall et al. 2006; Li and Hsing 2010) and its capability of handling data with different sampling schemes over various domains. Thus, **Issue C** is solved. A local linear smoother is also more appropriate for large-scale smoothing as most smoothing techniques, such as spline-based approaches, require the users to solve a global linear system where all the data are involved. However, for large-scale problems loading all the data into a computer’s memory is nearly impossible.

On the contrary, a local linear smoother estimates the function locally and independently, and thus can be performed parallelly and distributively; for example, it can be done through a modern GPGPU (general-purpose computing on graphics processing units) architecture. Another nice aspect of a local linear smoother is that it can easily be incorporated with the random projection procedure (Halko et al. 2011) and block matrix operations, which make the eigendecomposition step feasible when the empirical covariance function is of an enormous size. When the observations are made at the nodes of a regular grid, **Issue C** no longer exists and the local linear smoothing procedure can be further accelerated by using the FFT (fast Fourier transform) based approaches (e.g., Silverman 1982; Breslaw 1992; Wand 1994, etc.). These FFT based approaches can be easily parallelized as well and these arguments regarding to computational issues will be further elaborated in Sect. 3.

The rest of this paper proceeds as follows. Section 2 is comprised of the d-FPCA framework, the proposed estimators and their asymptotic properties. The major computational issues with solutions are provided in Sect. 3. Simulation studies and an illustrative real data analysis are presented in Sects. 4 and 5, respectively. Section 6 contains concluding remarks. The theoretical assumptions are delegated to an Appendix.

## 2 Methodology

We consider the stochastic process, for $t \in \Omega$,

$$X(t) = \mu(t) + \sum_{\ell=1}^{\infty} A_{\ell} \phi_{\ell}(t), \quad (2.1)$$

where $\mu$ is the mean function, $\phi_{\ell}$ is the $\ell$-th eigenfunction of $\Gamma$, the covariance function of $X$, and $A_{\ell}$ is the $\ell$-th principal component score. Without loss of generality, we assume the compact space $\Omega \subset [0, 1]^d$ and thus the domain of $\Gamma$ is $\Omega \times \Omega$. The principal component scores, $A_{\ell}$’s, are uncorrelated random variables with mean 0 and variance $\lambda_{\ell}$, where $\lambda_{\ell}$ is the eigenvalue of $\Gamma$ corresponding to eigenfunction $\phi_{\ell}$. We further assume that $\mu$, $\Gamma$ and $\phi_{\ell}$ are smooth functions. Detailed assumptions can be found in the Appendix.

### 2.1 Estimation

In practice, most data are available at discrete time points and might further be contaminated with measurement errors. We denote the observations of the $i$-th sample made at $t_{ij} = (t_{ij,1}, \ldots, t_{ij,d})^T$ as

$$Y_{ij} = X_i(t_{ij}) + \epsilon_{ij}$$
\[
\hat{\mu}(t) = \hat{b}_0, \\
\hat{b} = \arg \min_{b=(b_0, \ldots, b_d)^T \in \mathbb{R}^{d+1}} \sum_{i=1}^n \frac{1}{N_i} \sum_{j=1}^{N_i} K_h(t-t_{ij}) \\
\times \left[ Y_{ij} - b_0 - \sum_{k=1}^d b_k(t_k - t_{ij,k}) \right]^2, \quad (2.3)
\]

where \( K_h(t-t_{ij}) = \prod_{k=1}^d K_{h_{\mu,k}}(\frac{t-t_{ij,k}}{h_{\mu,k}}) \), \( h_{\mu,k} \) is the bandwidth for the \( k \)-th coordinate and \( K(\cdot) \) is a kernel function. Once the mean function is estimated, the covariance function can be obtained by

\[
\hat{\Gamma}(s, t) = \hat{b}_0 - \hat{\mu}(s) \hat{\mu}(t), \quad \text{where}
\]

\[
\hat{b} = \arg \min_{b=(b_0, \ldots, b_d)^T \in \mathbb{R}^{d+1}} \sum_{i=1}^n \frac{1}{N_i} \sum_{j \neq i \in N_i} K_h(t-t_{ij}) \\
\times \left[ Y_{ij} - Y_{i\ell} - b_0 - \sum_{k=1}^d b_k(s_k - s_{ij,k}) \right. \\
\left. - \sum_{k=1}^d b_{d+k}(t_k - t_{i\ell,k}) \right]^2, \quad (2.4)
\]

\( h_{\Gamma,k}'s \) are the bandwidths and \( K_h(\cdot) \) is defined as (2.3). The eigenvalues and eigenfunctions can be estimated by solving (3.1) with \( \hat{\Gamma}(s, t) \) replaced by \( \hat{\Gamma}(s, t) \). Details could be found in Sect. 3.

When \( Y_{ij}'s \) are densely observed, the principal component scores, \( A_{i\ell}'s \), can be predicted by inner products. However, when the observations are not dense enough, applying inner products might not generate satisfactory results. To solve this issue for irregularly observed longitudinal data, Yao et al. (2005a) proposed PACE. It is known that the PACE is a BLUP (best linear unbiased predictor) even when the normality assumption is violated. To employ PACE, we further need the estimate of \( \sigma^2 \). Since \( \text{cov}(Y_{ij}, Y_{ij'}) = \Gamma(t_{ij}, t_{ij'}) + \sigma^2 \delta_{jj'} \), where \( \delta_{jj'} = 1 \) if \( j = j' \) and 0 otherwise, \( \sigma^2 \) can be estimated by the strategy taken in Yao et al. (2005a). First, we estimate \( \Gamma(t, t) + \mu(t)\mu(t) + \sigma^2 \) by \( \hat{b}_0 \), where

\[
\hat{b}_0 = \arg \min_{b=(b_0, \ldots, b_d)^T \in \mathbb{R}^{d+1}} \frac{1}{N_i} \sum_{j=1}^{N_i} K_{h_\sigma}(t-t_{ij}) \\
\times \left[ Y_{ij} - b_0 - \sum_{k=1}^d b_k(t_k - t_{ij,k}) \right]^2, \quad (2.5)
\]

\( K_{h_\sigma} \) is defined as (2.3) and \( h_{\sigma,k}'s \) are the bandwidths. Then,

\[
\hat{\sigma}^2(t) = \hat{b}_0 - \hat{\Gamma}(t, t) - \hat{\mu}(t)\hat{\mu}(t).
\]

If \( \hat{\sigma}^2 \) does not change over \( t \), we consider to estimate it by

\[
\hat{\sigma}^2 = \frac{1}{|\Omega|} \int_{\Omega} \hat{\sigma}^2(t) dt. \quad (2.6)
\]

Note that more sophisticated approaches could be considered to produce more robust estimates for \( \hat{\sigma}^2 \). With \( \hat{\sigma}^2 \), we can now predict \( A_{i\ell} \) by PACE. Specifically,

\[
\hat{A}_{i\ell} = \hat{\alpha}_{i\ell} \hat{\Phi}_{i\ell}^T \hat{\Sigma}_{Y_i}^{-1} Y_i^C, \quad (2.7)
\]

where

\[
\hat{\Phi}_{i\ell} = (\hat{\phi}_{i\ell}(t_{i1}), \ldots, \hat{\phi}_{i\ell}(t_{iN_i}))^T, \\
\hat{\Sigma}_{Y_i} = \sum_{\ell=1}^L \hat{\alpha}_{i\ell} \hat{\Phi}_{i\ell} \hat{\Phi}_{i\ell}^T + \hat{\sigma}^2 I_{N_i \times N_i},
\]

and \( Y_i^C = (Y_{i1} - \hat{\mu}(t_{i1}), \ldots, Y_{iN_i} - \hat{\mu}(t_{iN_i}))^T \).

By combining the above estimates, \( X_i(t) \) can be reconstructed as

\[
\hat{X}_i(t) = \hat{\mu}(t) + \sum_{\ell=1}^L \hat{\alpha}_{i\ell} \hat{\phi}_{i\ell}(t),
\]

where \( L \) can be selected by AIC, BIC, FVE (fraction of variation explained) or other model selection criteria; FVE is employed in this paper. The product Epanechnikov kernel function is employed in our numerical studies. The bandwidths in the aforementioned estimators can be decided with some data-driven approach, such as the method proposed in Ruppert et al. (1995). We will elaborate our bandwidth selection procedure in Sect. 3.3.

### 2.2 Asymptotical properties

The strategies taken in Li and Hsing (2010) may be employed to show the asymptotical properties of our estimators, and
similar asymptotic properties are obtained. The proofs are provided in the supplement. Below we will provide the results while the assumptions are listed in the Appendix. Two sampling schemes on $N_i$ for $d \geq 2$ will be discussed; one corresponds to functional data, while the other corresponds to longitudinal data. Denote $\gamma_{nk} = \left(n^{-1} \sum_{i=1}^{n} N_i^{-1}\right)^{-1}$ for $k = 1$ and $2$, $\delta_n(h) = \left[1 + 1/(h^d \gamma_{nk})\right]\log n/n^{1/2}$ and $\delta_n(h) = \left[1 + 1/(h^d \gamma_{nk}) + 1/(h^{2d} \gamma_{nk})\right]\log n/n^{1/2}$. We first provide the convergence rate for $\tilde{\mu}$.

**Theorem 1** Assume that A.1–A.4 hold. Then,

$$
sup_{t \in \Omega} |\tilde{\mu}(t) - \mu(t)| = O(h_{\mu}^2 + \delta_n(h_{\mu})) \ a.s. \tag{2.8}
$$

On the right hand side of (2.8), $O(h_{\mu}^2)$ is a bound for bias and the other term is the uniform bound for $|\tilde{\mu}(t) - E(\tilde{\mu}(t))|$. We next investigate the asymptotical results of Theorem 1 under two special sampling schemes.

**Corollary 1** Assume that A.1–A.4 hold.

(a) If $\max_{1 \leq i \leq n} N_i \leq \mathcal{M}$ for some fixed $\mathcal{M}$, then

$$
sup_{t \in \Omega} |\tilde{\mu}(t) - \mu(t)| = O(h_{\mu}^2 + \log n/\{nh_{\mu}^d\})^{1/2} \ a.s.
$$

(b) If $\max_{1 \leq i \leq n} N_i \geq \mathcal{M}_n$, where $\mathcal{M}_n^{-1} \approx h_{\mu}^2 \approx (\log n/n)^{d/4}$ is bounded away from zero, then

$$
sup_{t \in \Omega} |\tilde{\mu}(t) - \mu(t)| = O((\log n/n)^{1/2}) \ a.s.
$$

Corollary 1 indicates that the classical nonparametric rate for estimating a $d$-dimensional function can be achieved for longitudinal data and that the optimal convergence rate can be achieved if $N_i$, the number of observations per sample, is of the order $(n/\log n)^{d/4}$.

The following results are the convergence rates for $\hat{\Gamma}$ and $\hat{\sigma}^2$.

**Theorem 2** Assume that A.1–A.6 hold. Then,

$$
sup_{s, t \in \Omega} |\hat{\Gamma}(s, t) - \Gamma(s, t)| = O\left(h_{\mu}^2 + \delta_n(h_{\mu}) + h_{\Gamma}^2 + \delta_n(h_{\Gamma})\right) \ a.s.
$$

**Theorem 3** Assume that A.1–A.2 and A.5–A.7 hold. Then,

$$
sup_{t \in \Omega} |\hat{\sigma}^2 - \sigma^2| = O\left(h_{\Gamma}^2 + \delta_n(h_{\Gamma}) + \delta_n^2(h_{\sigma}) + h_{\sigma}^2 + \delta_n^2(h_{\sigma})\right) \ a.s.
$$

**Corollary 2** Assume that A.1–A.6 hold.

(a) If $\max_{1 \leq i \leq n} N_i \leq \mathcal{M}$ for some fixed $\mathcal{M}$ and $h_{\Gamma}^{1/2} \lesssim h_{\mu} \lesssim h_{\Gamma}$ then

$$
sup_{s, t \in \Omega} |\hat{\Gamma}(s, t) - \Gamma(s, t)| = O\left(h_{\Gamma}^2 + \log n/\{nh_{\Gamma}^d\}\right)^{1/2} \ a.s.
$$

(b) If $\max_{1 \leq i \leq n} N_i \geq \mathcal{M}_n$, where $\mathcal{M}_n^{-1} \lesssim h_{\Gamma}^d \lesssim h_{\mu}^d \lesssim (\log n/n)^{d/4}$ is bounded away from zero, then

$$
sup_{s, t \in \Omega} |\hat{\Gamma}(s, t) - \Gamma(s, t)| = O\left((\log n/n)^{1/2}\right) \ a.s.
$$

Corollary 2 shows that the classical nonparametric rate for estimating a 2d-dimensional function can be obtained for longitudinal data and the optimal convergence rates can be achieved if $N_i$ is of the order $(n/\log n)^{d/4}$ for functional data which is the same as the condition required for $\tilde{\mu}$ to have optimal convergence rate.

Further, we assume the nonzero eigenvalues are distinct as Li and Hsing (2010) due to the identifiability of eigenfunctions.

**Theorem 4** Assume that A.1–A.6 hold, for $1 \leq j \leq J$,

(i) $|\hat{\lambda}_j - \lambda_j| = O\left((\log n/n)^{1/2} + \xi + \delta_n^2(h_{\mu})\right)$ \ a.s.,

(ii) $\|\hat{\phi}_j(t) - \phi_j(t)\| = O\left((\xi + \delta_n(h_{\mu}) + \delta_n(h_{\Gamma}))\right)$ \ a.s.,

(iii) $sup_{t \in \Omega} |\hat{\phi}_j(t) - \phi_j(t)| = O\left((\xi + \delta_n(h_{\mu}) + \delta_n(h_{\Gamma}))\right)$ \ a.s.,

where $\xi = h_{\mu}^2 + h_{\Gamma}^2 + \delta_n^2(h_{\Gamma})$, and $J$ is an arbitrary fixed constant.

Again, we discuss the above results with two different sampling schemes for $d \geq 2$.

**Corollary 3** Assume that A.1–A.6 hold and $1 \leq j \leq J$ for an arbitrary fixed constant $J$.

(i) Suppose that $\max_{1 \leq i \leq n} N_i \leq \mathcal{M}$ for some fixed $\mathcal{M}$. If $(\log n/n)^{1/2} < h_{\mu} \lesssim h_{\Gamma} \lesssim (\log n/n)^{1/3}$,

(a) $|\hat{\lambda}_j - \lambda_j| = O\left(h_{\Gamma}^2 + \log n/\{nh_{\Gamma}^d\}\right)$ \ a.s.;

(b) both $sup_{t \in \Omega} |\hat{\phi}_j(t) - \phi_j(t)|$ and $\|\hat{\phi}_j(t) - \phi_j(t)\|$ are of the rate $O\left(h_{\Gamma}^2 + \log n/\{nh_{\Gamma}^d\}\right)$.

(ii) Suppose $\max_{1 \leq i \leq n} N_i \geq \mathcal{M}_n$, where $\mathcal{M}_n^{-1} \lesssim h_{\Gamma}^d \lesssim h_{\mu}^d \lesssim (\log n/n)^{d/4}$ is bounded away from zero. \(\hat{\lambda}_j - \lambda_j\),

sup_{t \in \Omega} |\hat{\phi}_j(t) - \phi_j(t)| and $\|\hat{\phi}_j(t) - \phi_j(t)\|$ are of the rate $O\left((\log n/n)^{1/2}\right)$.
3 Computational issues

Now, we provide solutions to Issues A and B and introduce GPU (graphics processing unit) parallelization as well as the FFT calculation for local linear smoothers.

3.1 Matrixization

Issue A can be tackled by properly matrixizing $\Gamma$ as the following eigen-eqarrays

$$\int_{\Omega} \Gamma(s,t)\phi_\ell(s)ds = \lambda_\ell \phi_\ell(t),$$
$$\int_{\Omega} \phi_\ell(t)\phi_\ell(t)dt = 0 \quad \text{for all } \ell \neq i,$$

(3.1)

can be well-approximated by their corresponding Riemann sums, which can be represented with matrix operations. We will illustrate the trick with a simple case where $d = 2$. Suppose that $\Gamma$ is available on a dense regular grid $(s_i,j,j_i,j_i',t_2,t_2')$ for $1 \leq i, i' \leq M_1$ and $1 \leq j,j' \leq M_2$. Let $\Sigma = [\Sigma_{ij'}]_{1 \leq i,i' \leq M_1}$ be a matrix consisting of block matrices

$$\Sigma_{ij'} = \left[ \int_{\Omega_{ij}} \Gamma(s_i,t,j,j',t_2,t_2') \right]_{1 \leq k,l \leq M_2},$$

and $\psi_\ell$ be the eigenvector of $\Sigma$ corresponding to eigenvalue $\lambda_\ell$. The eigenvalues, $\lambda_\ell$, and eigenfunctions, $\phi_\ell$, can be well-approximated by properly rescaling $\hat{\lambda}_\ell$ and $\hat{\psi}_\ell$ when $M_1$ and $M_2$ are sufficiently big. The rescaling is to ensure $\int \phi_\ell^2(t)dt = 1$. This matrixization idea holds for general $d$-dimensional cases, and the construction of $\Sigma$ is fairly simple. It is done by reshaping $\Gamma$ to a 2-dimensional matrix; for example, it can be done by the reshape function in MATLAB. Note that if the domain $\Omega$ is not a rectangle, one could simply find a rectangle compact domain $\bar{\Omega} \subset \Omega$, and performs integration on it by setting $\Gamma(s,t) = 0$ if $s$ or $t$ is not in $\Omega$ (e.g., Stewart 2012, Chap. 12).

3.2 Large-scale local polynomial smoothing

Computing the local linear estimators, (2.3)-(2.5), is time-consuming when $N = \sum_{i=1}^{n} N_i$ is large. Fortunately, a local linear smoother possesses some key characteristics that make GPU parallelization easy. Now, we take (2.3) as an illustrative example. The computations of $\hat{b}$ at any two distinct $t$’s are identical, but independent of each other and do not communicate with each other. A GPU comprises hundreds to thousands of micro computing units, and each of them can conduct simple calculations simultaneously. However, these computing units do not communicate with each other, and hence GPU parallelization may not be suitable for other smoothing approaches, such as smoothing splines. Our numerical experience indicates that the GPU implementation d-FPCA is nearly 100 times faster than the existing MATLAB package on a machine with a cheap GPU (NVIDIA Quadro K600) consisting of 192 threads.

When $N$ is very large, simply considering the GPU paralleled implementation may not be enough since the computational complexities of both (2.3) and (2.5) are $O(N^2)$, and that of (2.4) is $O(N^4)$. Taking the FFT strategy (Wand 1994) could greatly reduce the computational complexities of (2.3)-(2.5) to $O(N \log N)$, $O(N^2 \log N)$ and $O(N \log N)$, respectively. Although the FFT is restricted for regularly observed functional data, the binning strategy (Wand 1994) can be taken for irregularly observed data. Even though binned local linear estimators are asymptotically consistent (Hoti and Holmström 2003), the consistency highly relies on the order of bin widths. In order to achieve it, the orders of bin widths have to be smaller than those of the optimal bandwidths and Hall and Wand (1996) discussed the minimum number of bins so that the binning effect is negligible.

To relax the memory limitation of FFT without sacrificing the computational speed, we partition the domain into several overlapping blocks and apply the FFT-based local linear smoothing to each block. We choose overlapping blocks to avoid additional boundary effects due to smoothing. The block size is decided to achieve a balance between system memory and computational speed.

3.3 Bandwidth selection

Bandwidth selection is crucial in smoothing. The bandwidth minimizing the distance between fitted and true functions, i.e. conditional mean integrated squared error (MISE), is often considered as an optimal bandwidth. To avoid overfitting, one commonly considers a cross-validation (CV) procedure to select a proper bandwidth and a valid scheme in FDA is the leave-one-sample-out CV due to theoretical considerations. Unfortunately, performing the leave-one-sample-out CV is very time-consuming and unlike traditional smoothing problems, the equivalent form in MISE between CV and GCV does not exist. Therefore, a different viewpoint is taken here. The bandwidth selection procedure is to provide some guidance on suitable bandwidths for a given dataset and in practice one may need to further adjust these objective suggestions manually Liu and Müller (2009). Also, GCV generally generates satisfactory results as the MATLAB package PACE (Yao et al. 2005a) considers GCV as a default option. As a consequence, we consider the the leave-one-observation-out CV to perform bandwidth selection for its nice connection to GCV.

Although the conditional MISE is asymptotically a smooth and convex function of bandwidths, its empirical version could be very noisy and thus with multiple local minima.
So, traditional numerical minimization algorithms cannot be directly applied for searching the optimal bandwidths. The CV or GCV procedures in most statistical packages are carried out by grid search; however, grid search is very time-consuming, especially when $d$ is not small. Therefore, we propose a numerical optimization procedure adaptive to the noisy CV or GCV criteria. Our idea originates from the derivative-free trust-region interpolation-based method (Marazzi and Nocedal 2002), a Newton-like algorithm. Here, the gradient and hessian of the objective function are “estimated” by locally quadratic interpolations. To be adaptive for the noisy criteria, we replace the interpolation step in Marazzi and Nocedal (2002) with a quadratic regression. Our algorithm requires $O(d^3)$ iterations to converge in the worst case (Conn et al. 2009), which is a great improvement over grid search.

### 3.4 Random projection

**Issue B** could be vital even with a moderate $d$ as the size of $\Sigma$ grows exponentially. Suppose we have $d$-dimensional functional data with 100 observation grids in each coordinate and $\Sigma$ will consist of $10^{4d}$ elements. It is challenging to perform eigendecomposition on $\Sigma$ due to the memory issue. Take $d = 3$ for example; it requires approximately $4 \times 10^{12} \approx 4$TB to store $\Sigma$ with single precision floating points. One remedy is employing the random projection approximation (Halko et al. 2011), which is based on the following algebraic property. Suppose $\Sigma = \Psi \Lambda \Psi^T$ by eigendecomposition. For any $p \times q$ matrix $\tilde{Q}$ satisfying $\tilde{Q} \tilde{Q}^T \approx I_p$ and $q \ll p$, $\Sigma' = \tilde{Q}^T \Sigma \tilde{Q}$ can be expressed as $\tilde{Q}^T \Psi \Lambda \Psi^T \tilde{Q}$. This implies that one can perform eigendecomposition on a smaller matrix $\Sigma'$ to obtain $\Lambda$ and $\Psi$. Specifically, $\Psi \approx \tilde{Q} (\tilde{Q}^T \Psi)$, where $\tilde{Q}^T \Psi$ are the eigenvectors of $\Sigma'$. To choose a proper $\tilde{Q}$, Halko et al. (2011) proposed to employ a random matrix $\tilde{Q} = [\tilde{q}_{ij}]$, where $\tilde{q}_{ij} \sim N(0,1/q)$, because $E(\tilde{Q} \tilde{Q}^T) = I_p$. The computer codes for the random projection approximation are available on the parallel computing platforms, such as Hadoop and Spark. Note that with our block-wise FFT strategy, $\Sigma'$ can be obtained by block matrix multiplications and thus $\tilde{Q}^T \Psi$ without storing the entire $\Sigma$ in a computer’s memory. However, this random projection strategy is based on the assumption that the rest ($p - q$) eigenvalues are all zeros. If this assumption is violated, we will need to investigate a new algorithm for eigendecomposition and this will be our future project.

### 4 Simulation studies

Two simulation studies are conducted to demonstrate the performance of our approach. In the first study, we compare the computational speed of our implementations on the local linear smoothing procedure with that of existing statistical packages. Among them, the MATLAB package PACE (Yao et al. 2005a) is selected as it was specifically designed for one-dimensional FPCA and generates satisfactory results. We consider two different implementations on the local linear smoothing procedures: the GPU parallelization and the FFT strategy. The GPU parallelization implementation is written in C using the OpenACC toolkit Herdman et al. (2014) and called in MATLAB; the FFT implementation is written in MATLAB using the MATLAB subroutine fftn. In the second study, we consider $d = 3$ to demonstrate the potential of our approach for 3D functional data. We will demonstrate the computational advantages of taking the random projection strategy. The random projection approximation is implemented in MATLAB on our own.

#### 4.1 Simulation I

Following Yao et al. (2005a), we generate the data from

$$Y_i(t) = X_i(t) + \epsilon = \mu(t) + \sum_{\ell=1}^{2} A_{i\ell} \phi_{\ell}(t) + \epsilon,$$

where $\mu(t) = t + \sin(t)$, $\phi_{1}(t) = -\cos(\pi t/10)/\sqrt{5}$, $\phi_{2}(t) = \sin(\pi t/10)/\sqrt{5}$, $A_{11} \sim N(0,4)$, $A_{12} \sim N(0,1)$, and $\epsilon \sim N(0,0.25)$. The simulation consists of 100 runs and each run contains 100 random curves sampled from 10000 equispaced points in $[0,10]$. To have fair comparisons, we select the number of eigenfunctions by setting the FVE to be at least 95%, and the principal component scores are estimated through integration for both PACE and our d-FPCA. While applying PACE, we choose the data type as “dense functional”, and all the other options are set as default. To accelerate the computational speed, PACE bins the original data into 400 equispaced bin grids and selects the bandwidths via GCV. The experiment is performed on a machine with Intel E3-1230V3 CPU, 32GB RAM and NVIDIA Quadro K600 GPU. The computational time as well as the MISE,

$$\frac{1}{100} \sum_{i=1}^{100} \int_{0}^{10} [\hat{X}_i(t) - X_i(t)]^2 dt,$$

are summarized in Table 1, indicating that our d-FPCA provides slightly (but, not significantly) better reconstructed curves than PACE does, while our computational speed is about 100 times and 300 times faster than PACE with GPU and FFT implementations, respectively.
Table 1  Computational time (in seconds) and MISE for PACE and d-FPCA (mean ± std)

|        | PACE      | d-FPCA (GPU) | d-FPCA (FFT) |
|--------|-----------|--------------|--------------|
| Time   | 314.1 ± 11.1 | 3.2 ± 1.0    | 1.0 ± 0.3    |
| MISE   | .015 ± .019 | .014 ± .016  | .014 ± .016  |

4.2 Simulation II

In the second experiment, we consider the following model

\[ Y(t) = X(t) + \epsilon = \mu(t) + \sum_{\ell=1}^{4} A_{\ell} \phi_{\ell}(t) + \epsilon, \]  

(4.2)

where

\[ \mu(t) = \exp((t - 0.5)^T (t - 0.5)), \]

\[ \phi_{\ell}(t) = \prod_{i=1}^{3} (\sqrt{2} \sin(2\ell \pi t_i)) \text{ for } \ell = 1, \ldots, 4, \]

\( t = (t_1, t_2, t_3)^T \in [0, 1]^3, A_{\ell} \sim N(0, 4^{3-\ell}), \) and \( \epsilon \sim N(0, 1/16). \) In each run, we generate 100 random functions on the nodes of a 64^3 equispaced grid. Notice that we need 64^6 × 4 ≈ 256 GB RAM to store the entire covariance function \( \Sigma \) if single precision floating points are used, and so performing eigendecomposition directly is demanding. Two approaches are compared here.

Both approaches adapt the FFT strategy in the step of estimating \( \mu \) and \( \Sigma \); the difference is that the first approach performs conventional eigendecomposition to obtain eigenfunctions while the other employs the random projection strategy and the blockwise FFT to obtain eigenfunctions. Performing conventional eigendecomposition on \( \Sigma \) requires additional 256 GB of RAM to obtain all the eigenfunctions; clearly, it does not appear feasible in practice. To be conservative in memory usage, we only extract the first 99 eigenfunctions, which is much more than the number of true eigenfunctions. The experiment is carried out on a Microsoft Azure G5 instance, a virtual machine with 32 cores of Intel Xeon E5 V3 CPU and 448GB of RAM. The Microsoft Azure G5 instance is the machine with the largest memories that we can access. In the second approach, the random projection, \( \Sigma' = Q^T \Sigma Q \), can be obtained by blockwise matrix multiplications, and hence it does not require 256 GB RAM to secure \( \Sigma' \). To perform blockwise FFT, we partition the covariance function into 8 overlapping blocks. In the random projection step, we use \( q = 99 \). The experiment is carried out on a Microsoft Azure D14 instance, a virtual machine with 16 cores of Intel Xeon E5 V3 CPU and 112GB of RAM.

To save cost, this simulation only consists of 20 runs. To complete one run, the first approach takes approximately 8.5 hours, while the second one takes approximately 5 hours. The integrated squared error (ISE) of the \( t \)th eigenfunction is defined as

\[ \int_{\Omega} (\hat{\phi}_t(t) - \phi_t(t))^2 dt, \]

and the results are summarized in Table 2. The reconstruction errors of both methods are almost equivalent, but the one based on the random projection does introduce very little errors into the estimated eigenfunctions. However, those errors are insignificant and negligible compared to its computational advantage.

5 Data analysis

Recent studies have shown PM2.5 is a potential risk factor for lung cancer (Raaschou-Nielsen et al. 2013) and heart attack (Cesaroni et al. 2014); thus, monitoring the level of PM2.5 becomes urgent. The dataset consisting of PM2.5 measurements (in \( \mu g/m^3 \)) made at 25 irregularly located monitor sites on western Taiwan from Nov. 29, 2012 to Mar. 1, 2015 (available at http://taqm.epa.gov.tw/pm25/tw/) is considered in this section. The measurements were carried out manually every three days; therefore, the sample size is 275. Due to experimental failures, machine malfunction and that all the monitor sites were not established simultaneously, the measurements are not complete. One might consider to perform one-dimensional FPCA to this dataset, i.e., the measurements made at each monitor site represent one sample curve. However, doing so leads to serious problems; the sample size becomes very small (i.e., \( n = 25 \)) and the samples are highly correlated due to their spatial locations. Moreover, because

Table 2  The reconstruction errors (mean ± std) and the ISE of estimated eigenfunctions (mean ± std) for d-FPCA with/without random projection (RP)

|        | \( \hat{\phi}_1 \)     | \( \hat{\phi}_2 \)     | \( \hat{\phi}_3 \)     | \( \hat{\phi}_4 \)     |
|--------|----------------------|----------------------|----------------------|----------------------|
| RP     | .003 ± 6.2 × 10^{-6} | .1122 ± .0035       | .1137 ± .0038        | .1137 ± .0038        | .1140 ± .0037        |
| w.o. RP| .003 ± 6 × 10^{-6}   | .1125 ± .0036       | .1133 ± .0038        | .1133 ± .0038        | .1137 ± .0036        |
the measurements were made every three days and they are highly dependent on the weather conditions (e.g., winds and rains), the true temporal correlation is difficult to estimate consistently with current available data resolution. All these are problematic in performing FPCA. To the best of our knowledge, no existing functional approach is capable of analyzing such type of data. So, only d-FPCA is applied to this dataset and the following model is considered

\[ Y_i(s) = \mu(s) + \sum_{k=1}^{L} A_{ik} \phi_k(s) + \epsilon, \]

where \( s \) is a given spatial location, \( L \) is the number of eigenfunctions and \( i = 1, \ldots, 275 \).

Figure 1 shows the estimated mean function, \( \hat{\mu} \), and the estimates of the first three eigenfunctions, \( \hat{\phi}_k \) for \( k = 1, 2, 3 \).
The fractions of variation explained by the first three eigenfunctions are 78.6, 13.0, and 7.1 %, respectively. Figure 1a indicates that the level of PM 2.5 on southwestern Taiwan is higher than that on northern Taiwan on average. This may be because most of the coal thermal power stations and heavy industry factories (e.g., steel mills and petrochemical factories) are on southwestern Taiwan. Note that |\( \phi_k \)| represents the variation magnitude of PM 2.5 level at location \( s \). A positive \( \phi_k(s) \) with a positive \( A_{ik} \) implies that the PM 2.5 level at location \( s \) at time \( i \) is higher than \( \mu(s) \) given that other variables remain unchanged. This may be due to the PM 2.5 accumulation at location \( s \). On the contrary, a negative \( A_{ik} \) implies that PM 2.5 level at time \( i \) is lower than its average, which may be because of rains or winds. Figure 1b indicates that the issue of PM 2.5 is very severe around Yunlin County, where the largest petrochemical complex in Taiwan is located, as the first eigenfunction explains 78.6 % variation of the PM 2.5 data and the PM 2.5 level could become very high in this area. Figure 1c indicates that the PM 2.5 levels in Kaohsiung and Tainan (two consecutive cities in southern Taiwan) could become relatively high. One possible explanation is that most of the petrochemical factories in Taiwan are located in Kaohsiung. Figure 1d suggests that the PM 2.5 level in Taipei could also get high even though its average is relatively low and this might result from the air pollutant in China carried by the northeast monsoon in winter. Figure 2 further confirms this conjecture as the principal component scores are all very close to zero from April to September while they fluctuate a lot from October to March. The explanation for negative principal component scores is that it rains very often on northern Taiwan in winter. Further, all three estimated eigenfunctions show that a small area in Nantou county could have relatively high level of PM2.5 and this may be due to the accumulation of fine particulate matter blocked by the Central Range.

Two experiments are conducted to demonstrate the reconstruction ability of d-FPCA. In the first experiment, each run we randomly split the data into training and validation sets with 175 and 100 samples, respectively. It consists of 50 runs. Again, MISE is employed to evaluate the performance of d-FPCA; the average and standard error are 29.15 and 4.46, respectively. In the second experiment, we evaluate d-FPCA’s ability to predict missing values by means of the leave-one-location-out CV strategy. The performance is measured with the squared prediction error,

\[
\sum_{j \in I_i} \left[ Y_j(s_i) - \hat{Y}^{(-i)}_j(s_i) \right]^2,
\]

where \( Y_j(s_i) \) is the measurement of location \( s_i \) made at time \( j \), \( \hat{Y}^{(-i)}_j(s_i) \) is the prediction of \( Y_j(s_i) \) and \( I_i \) is the set of observation times for location \( s_i \). The average and standard error of (5.1) are 31.17 and 4.52, respectively. Figure 3 shows the reconstructions at two specific locations, the closest stations to where the two authors live. Figure 3 indicates that d-FPCA works very well since distributions of the observations and the corresponding predictions are very close to the straight line (\( x = y \)).

6 Conclusions

The idea of extending FPCA to analyze multidimensional functional data is very natural and the extension is quite straightforward conceptually and theoretically; however, the emerging computational issues for practical data analyses are not so easy. To tackle these issues, we employed the estimators based on local linear smoothers because of their capabilities of performing large-scale smoothing and handling data with different sampling schemes over various domains in addition to their nice theoretical properties. To accelerate the computational speed, we employed both the FFT strategy and the GPU parallel computing in the smoothing step. The out-of-memory problem in the eigen-decomposition step due to large-scale data has been properly addressed by using the random projection approximation and block matrix operations. Our strategies on conducting large-scale smoothing and performing eigendecomposition on huge covariance matrices are completely applicable to other FDA approaches for multi-dimensional functional data, such as partial least-squares (Preda and Saporta 2005; Delaigle and Hall 2012), functional sliced inverse regression Ferré and Yao (2003, 2005), etc.

We have also investigated the asymptotic properties of the proposed estimators under two different sampling schemes.
Specifically, we have shown that the classical nonparametric rates can be achieved for longitudinal data and the optimal convergence rates can be achieved if $N_i$, the number of observations per sample, is of the order $(n / \log n)^{d/4}$ for functional data. The finite sample performance of our approach has been demonstrated with simulation studies and the fine particulate matter (PM 2.5) data measured in Taiwan. Although only functional data cases have been considered in our numerical experiments, the proposed approach is definitely applicable for longitudinal data by employing “PACE” to predict the principal component scores.

Acknowledgements The authors would like to thank the Editor, the Associate Editor and the referees for their constructive comments which improve the presentation of the paper substantially. The research of Lu-Hung Chen is partially supported by the Ministry of Science and Technology of Taiwan under grant MOST 104-2118-M-005-003; the research of Ci-Ren Jiang is partially supported by the Ministry of Science and Technology of Taiwan under grant MOST 103-2118-M-001-001.

Appendix: Assumptions

Since the estimators, $\hat{\mu}(t)$ and $\hat{\Gamma}(s, t)$, are obtained by applying the local linear smoothing approaches, it is natural to make the standard smoothness assumptions on the second derivatives of $\mu$ and $\Gamma$. Assumed that the data $(T_i, Y_i), i = 1, \ldots, n$, are from the same distribution, where $T_i = (t_{i1}, \ldots, t_{iN_i})$ and $Y_i = (Y_{i1}, \ldots, Y_{iN_i})$. Notice that $t_{ij}$’s are random observational points with density $g_T(\cdot)$. Additional assumptions and conditions are listed below.

The following assumptions of $Y(t)$ were also made in Li and Hsing (2010). Suppose the observation of the $i$th subject at time $t_{ij}$ is $Y_{ij} = \mu(t_{ij}) + U_{ij}$, where $\text{cov}(U_i(s), U_i(t)) = \Gamma(s, t) + \sigma^2 I(s = t)$ and $\Gamma(s, t) = \sum_\ell \lambda_\ell \phi_\ell(s)\phi_\ell(t)$.

A.1 For some constant $m_T > 0$ and $M_T < \infty$, $m_T \leq g_T(t) \leq M_T$ for all $t \in \Omega$. Further, $g_T$ is differentiable with a bounded derivative.
A.2 The kernel function $K(\cdot)$ is a symmetric probability density function on $[-1, 1]$ and is of bounded variation on $[-1, 1]$.
A.3 $\mu$ is twice differentiable and the second derivative is bounded on $\Omega$.
A.4 $E(|U_{ij}|^\lambda) < \infty$ and $E(\sup_{t \in \Omega} |X(t)|^\lambda) < \infty$ for some $\lambda \in (2, \infty)$; $h_\mu \to 0$ and $(h_\mu^{2d} + h_\mu/\gamma_1)^{-1}(\log n/n)^{1-2/\lambda} \to 0$ as $n \to \infty$.
A.5 All second-order partial derivatives of $\Gamma$ exist and are bounded on $\Omega \times \Omega$.
A.6 $E(|U_{ij}|^2\lambda) < \infty$ and $E(\sup_{t \in \Omega} |X(t)|^{2\lambda}) < \infty$ for some $\lambda \in (2, \infty)$; $h_\Gamma \to 0$ and $(h_\Gamma^{2d} + h_\Gamma^{2d}/\gamma_1 + h_\Gamma^{2d}/\gamma_2)^{-1}(\log n/n)^{1-2/\lambda} \to 0$ as $n \to \infty$.
A.7 $E(|U_{ij}|^4) < \infty$ and $E(\sup_{t \in \Omega} |X(t)|^{4\lambda}) < \infty$ for some $\lambda \in (2, \infty)$; $h_{\sigma} \to 0$ and $(h_{\sigma}^{4d} + h_{\sigma}^{4d}/\gamma_1)^{-1}(\log n/n)^{1-2/\lambda} \to 0$ as $n \to \infty$.

References

Aston, J.A., Kirch, C.: Estimation of the distribution of change-points with application to fmri data. Ann. Appl. Stat. 6, 1906–1948 (2012)
Aston, J.A., Chiou, J.M., Evans, J.: Linguistic pitch analysis using functional principal component mixed effect models. J. R. Stat. Soc. Ser. C 59, 297–317 (2010)
Aue, A., Norinho, D.D., Hörmann, S.: On the prediction of stationary functional time series. J. Am. Stat. Assoc. 110, 378–392 (2015)
Breslau, J.A.: Kernel estimation with cross-validation using the fast fourier transform. Econ. Lett. 38, 285–289 (1992)
Yao, F., Müller, H.G., Wang, J.L.: Functional linear regression analysis for longitudinal data. Ann. Stat. 33, 2873–2903 (2005b)
Zhang, T., Li, F., Beckes, L., Coan, J.A.: A semi-parametric model of the hemodynamic response for multi-subject fmri data. NeuroImage 75, 136–145 (2013)
Zhou, L., Pan, H.: Principal component analysis of two-dimensional functional data. J. Comput. Graph. Stat. 23, 779–801 (2014)
Zhu, H., Zhang, H., Ibrahim, J., Peterson, B.: Statistical analysis of diffusion tensors in diffusion-weighted magnetic resonance image data (with discussion). J. Am. Stat. Assoc. 102, 1081–1110 (2007)

Zipunnikov, V., Caffo, B., Yousem, D.M., Davatzikos, C., Schwartz, B.S., Crainiceanu, C.: Functional principal component model for high-dimensional brain imaging. NeuroImage 58, 772–784 (2011a)
Zipunnikov, V., Caffo, B., Yousem, D.M., Davatzikos, C., Schwartz, B.S., Crainiceanu, C.: Multilevel functional principal component analysis for high-dimensional data. J. Comput. Graph. Stat. 20, 852–873 (2011b)