# Fast Covariance Estimation for High-dimensional Functional Data

Luo Xiao[*], David Ruppert [†], Vadim Zipunnikov[*], and Ciprian Crainiceanu[*]

June 24, 2013

## Abstract

For smoothing covariance functions, we propose two fast algorithms that scale linearly with the number of observations per function. Most available methods and software cannot smooth covariance matrices of dimension $J \times J$ with $J > 500$; the recently introduced sandwich smoother is an exception, but it is not adapted to smooth covariance matrices of large dimensions such as $J \geq 10,000$. Covariance matrices of order $J = 10,000$, and even $J = 100,000$, are becoming increasingly common, e.g., in 2- and 3-dimensional medical imaging and high-density wearable sensor data. We introduce two new algorithms that can handle very large covariance matrices: 1) FACE: a fast implementation of the sandwich smoother and 2) SVDS: a two-step procedure that first applies singular value decomposition to the data matrix and then smoothes the eigenvectors. Compared to existing techniques, these new algorithms are at least an order of magnitude faster in high dimensions and drastically reduce memory requirements. The new algorithms provide instantaneous (few seconds) smoothing for matrices of dimension $J = 10,000$ and very fast ($< 10$ minutes) smoothing for $J = 100,000$. Although SVDS is simpler than FACE, we provide ready to use, scalable R software for

[*]Department of Biostatistics, Johns Hopkins University, Baltimore, MD

[†]Department of Statistical Science and School of Operations Research and Information Engineering, Cornell University, Ithaca, NY

FACE. When incorporated into R package *refund*, FACE improves the speed of penalized functional regression by an order of magnitude, even for data of normal size ($J < 500$). We recommend that FACE be used in practice for the analysis of noisy and high-dimensional functional data.

**Keywords**: FACE; fPCA; penalized splines; sandwich smoother; smoothing; singular value decomposition.

## 1 Introduction

The covariance function plays an important role in many areas of functional data analysis (FDA), for example, functional principal component analysis (fPCA), functional linear regression, and functional canonical correlation analysis (see, e.g., Ramsay and Silverman, 2002; and Ramsay and Silverman, 2005). There are several major differences between functional and multivariate data. Functional data are measured on the same scale, with sizable noise, and possibly sampled at an irregular grid. Ordering of functional observations is also important, but it can easily be handled by careful indexing. Thus, functional data require new methods for covariance estimation. In particular, in FDA we usually need to smooth the estimated covariance function. There are a number of existing smoothing methods for covariance functions, but none of them are sufficiently fast to work with the high-dimensional functional data that are becoming increasingly common. The dimension $J$ of functional data is often greater than 10,000, which implies that the dimension of the covariance function, $J^2$, is greater than 100 million. We introduce two new smoothing algorithms that are practical for $J$ as large as 100,000.

It has become common practice in functional data analysis to estimate functional principal components by diagonalizing a smoothed estimator of the covariance function; see, e.g., (Besse and Ramsay, 1986; Ramsay and Dalzell, 1991; Kneip, 1994; Besse et al., 1997; Staniswalis and Lee, 1998; Yao et al., 2003, 2005). The sample covariance function, its eigenvalues, and its eigenvectors converge to their population counterparts at the optimal rate when the sample paths are completely observed without measurement error (Dauxois et al., 1982). However, in practice, data are measured at a finite number of locations and often with sizable measurement error. For such data, the eigenvectors of the sample covariance matrix tend to be noisy, which can substantially reduce interpretability. Therefore, smoothing is often used to estimate the functional principal components; see, e.g., Besse and Ramsay (1986); Ramsay and Dalzell (1991); Rice and Silverman (1991); Kneip (1994); Capra and Müller (1997); Besse et al. (1997); Staniswalis and Lee (1998); Cardot (2000); Yao et al. (2003, 2005).

There are three main approaches to estimating smooth functional principal components. The first approach is to smooth the functional principal components of the sample covariance function; for a detailed discussion see, for example, Rice and Silverman (1991); Capra and Müller (1997); Ramsay and Silverman (2005). The second is to smooth the covariance function and then diagonalize it; see, e.g., Besse and Ramsay (1986); Staniswalis and Lee (1998); Yao et al. (2003). The third is to smooth each curve and diagonalize the sample covariance function of the smoothed curves; see Ramsay and Silverman (2005) and the references therein.

Our first approach is a fast bivariate smoothing algorithm for the covariance operator which connects the latter two approaches. This algorithm is a fast and new implementation of the ‘sandwich smoother’ in Xiao et al. (2013), with a completely different and specialized computational approach that improves the original algorithm’s computational efficiency by at least an order of magnitude. The sandwich smoother with the new implementation will be referred to as Fast Covariance Estimation, or FACE. Our second approach is to use penalized spline smoothing of the eigenvectors obtained from a high-dimensional singular value decomposition of the raw data matrix and will be referred to as SVD plus smoothing, or SVDS. To the best of our knowledge, this approach has not been used in the literature for low- or high-dimensional data. Given the simplicity of SVDS, we will focus more on FACE, though simulations and data analysis will be based on both approaches.

The sandwich smoother provides the next level of computational scalability for bivariate smoothers and has significant computational advantages over bivariate $P$-splines (Eilers and Marx, 2003; Marx and Eilers, 2005) and thin plate regression splines (Wood, 2003). These advantages are achieved, essentially, by transforming the technical problem of bivariate smoothing into a short sequence of univariate smoothing steps. For covariance matrix smoothing, the sandwich smoother was shown to be much faster than local linear smoothers. However, adapting the sandwich smoother to fast covariance matrix smoothing in the ultrahigh dimensions of, for example, modern medical imaging or high-density wearable sensor data (Bai et al., 2012; Shou et al., 2013), is not straightforward. For instance, the sandwich smoother requires the sample covariance matrix which can be hard to calculate and impractical to store for ultrahigh dimensions. While the sandwich smoother is the only available fast covariance smoother, it was never tested for dimensions $J > 5,000$ and becomes computationally impractical for $J > 50,000$ on current standard computers; $J > 50,000$ is well within the range of current high-dimensional data.

In contrast, FACE is linear in the number of functional observations per subject, provides instantaneous ($<$ 1 minutes) smoothing for matrices of dimension $J = 10,000$ and fast ($<$ 10 minutes) smoothing for $J = 100,000$. This is done by carefully exploiting the low-rank structure of the sample covariance, which allows smoothing and spectral decomposition of the smooth estimator of the covariance *without calculating or storing the empirical covariance operator*. The new approach is at least an order of magnitude faster in high dimensions and drastically reduces memory requirements; see Table 3 in Section 6 for a comparison of computation times. Unlike the sandwich smoother, FACE also efficiently estimates the covariance function, eigenfunctions, and scores.

The code for FACE will be available in R package *refund* (Crainiceanu et al., 2013) that is scheduled for an update in July and a developer version can be downloaded from the web link: `https://sites.google.com/site/xiaoyuesixi/publications/code`. We tested the speed of FACE and the default choice of bivariate smoothing in *refund*: thin plate regression splines (Wood, 2003), under the same simulation settings in Section 6.1 except that we let $J = 100$ and the number of curves be 200. Here the number of knots for FACE was 35 whereas the basis dimension of thin plate regression splines was 35. We found that FACE was at least an order of magnitude faster than thin plate splines: FACE gave instantaneous ($< 0.1$ second) results whereas the default method took more than 6 minutes on average. More importantly, after we incorporated FACE into penalized functional regression models (Goldsmith et al., 2011, 2012), the speed of these models has also been significantly improved. For instance, we used both FACE and thin plate regression splines (with a reduced basis dimension of 20) in running these models on the examples provided in *refund*: FACE took less than 20 seconds while the default method needed 7 minutes.

The remainder of the paper is organized as follows. Section 2 provides the model and data structure. Section 3 introduces FACE and provides the basic fast algorithm. Section 4 extends FACE to structured high-dimensional functional data and incomplete data. Section 5 introduces SVDS, the penalized spline smoothing of eigenvectors obtained from SVD. Section 6 provides simulation results and shows that FACE is faster and more accurate than SVDS. Section 7 shows how FACE and SVDS work in a large study of EEG signals recorded during sleep. Section 8 provides concluding remarks.

## 2 Model and Data Structure

Suppose that $\{X_i, i = 1, \dots, I\}$ is a collection of independent realizations of a random functional process $X$ with covariance function $K(s,t), s,t \in [0,1]$. The observed data, $Y_{ij} = X_i(t_j) + \epsilon_{ij}$, are noisy proxies of $X_i$ at the sampling points $\{t_1, \dots, t_J\}$. We assume that $\epsilon_{ij}$ are i.i.d. errors with mean zero and variance $\sigma^2$, and are mutually independent of the processes $X_i$. For simplicity we assume initially that $t_j$'s are the same across subjects; the case when sampling points vary across subjects is discussed in Section 3.3. For ease of presentation, we also assume that $Y_{ij}$ have been centered across subjects.

Let $\mathbf{Y}_i = (Y_{i1}, \dots, Y_{iJ})^T, i = 1, \dots, I$. The sample covariance matrix, $\widehat{\mathbf{K}}$, is the $J \times J$ dimensional matrix $\widehat{\mathbf{K}} = I^{-1} \sum_{i=1}^{I} \mathbf{Y}_i \mathbf{Y}_i^T = I^{-1}\mathbf{Y}\mathbf{Y}^T$, where $\mathbf{Y} = [\mathbf{Y}_1, \dots, \mathbf{Y}_I]$ is a $J \times I$ dimensional matrix with the $i$th column equal to $\mathbf{Y}_i$. Covariance smoothing typically refers to applying bivariate smoothers to $\widehat{\mathbf{K}}$. When $I$ is much smaller than $J$, $\widehat{\mathbf{K}}$ is of low rank; this low-rank structure of $\widehat{\mathbf{K}}$ will be particularly useful for deriving fast methods for smoothing $\widehat{\mathbf{K}}$.

## 3 FACE

The FACE estimator of the covariance matrix has the form

$$\widetilde{\mathbf{K}} = \mathbf{S}\widehat{\mathbf{K}}\mathbf{S}, \tag{1}$$

where $\mathbf{S}$ is a symmetric smoother matrix of dimension $J \times J$. Because of (1), we say FACE has a sandwich form. We use $P$-splines (Eilers and Marx, 1996) to construct $\mathbf{S}$ so that $\mathbf{S} = \mathbf{B}\left(\mathbf{B}^T\mathbf{B} + \lambda\mathbf{P}\right)^{-1}\mathbf{B}^T$. Here $\mathbf{B}$ is the $J \times c$ matrix $\{B_k(t_j)\}_{1\le j\le J, 1\le k\le c}$, $\mathbf{P}$ is a symmetric penalty matrix of size $c \times c$, $\lambda$ is the smoothing parameter, $\{B_1(\cdot), \dots, B_c(\cdot)\}$ is the collection of B-spline basis functions, $c$ is the number of interior knots plus the order (degree plus 1) of B-splines. We assume that the knots are equally spaced and use a difference penalty as in Eilers and Marx (1996) for the construction of $\mathbf{P}$. The sandwich smoother in Xiao et al.

(2013) has form $\mathbf{S}_1\mathbf{Y}\mathbf{S}_2$ where $\mathbf{Y}$ is a data matrix and $\mathbf{S}_1$ and $\mathbf{S}_2$ are smoother matrices. Therefore, the FACE estimator (1) is the special case of the sandwich smoother where $\mathbf{S}_1$ and $\mathbf{S}_2$ are identical and $\mathbf{Y}$ is a sample covariance matrix. Although the FACE *estimator* is not new, the fast FACE *algorithm* introduced in this paper is novel. Our new algorithm takes advantage of certain features of the FACE estimator not in general possessed by the sandwich smoother, e.g., that is specialized to smoothing covariance matrices.

Notice that $\widetilde{\mathbf{K}}$ inherits the symmetry and positive semi-definiteness of $\widehat{\mathbf{K}}$ because $\mathbf{S}$ is symmetric. The sandwich form of the smoother and the low-rank structure (assuming $I \ll J$) of the sample covariance matrix can be exploited to scale FACE to high and ultra high dimensional data ($J > 10,000$). For instance, the eigendecomposition of $\widetilde{\mathbf{K}}$ provides the estimates of the eigenfunctions associated with the covariance function. However, when $J$ is large, both the smoother matrix and the sample covariance matrix are high dimensional and even storing them may become impractical. FACE, unlike the sandwich smoother, is designed to obtain the eigendecomposition of $\widetilde{\mathbf{K}}$ *without computing the smoother matrix or the sample covariance matrix.*

FACE depends on a single smoothing parameter. The algorithm for selecting $\lambda$ in Xiao et al. (2013) requires $O(J^2I)$ computations and can be hard to compute when $J$ is large. We propose a new and efficient selection method for the smoothing parameter that requires only $O(JI + cI)$ computations; see Section 3.2 for details.

## 3.1 Estimation of Eigenfunctions

Assuming that the covariance function $K$ is square integrable, i.e., is in $L_2([0,1]^2)$, Mercer's theorem states that $K$ admits an eigendecomposition $K(s,t) = \sum_k \lambda_k \psi_k(s)\psi_k(t)$ where $\{\psi_k(\cdot) : k \geq 1\}$ are eigenfunctions of $K$ that form an orthonormal basis for $L_2([0,1])$ and $\lambda_1 \geq \lambda_2 \geq \cdots$ are the corresponding eigenvalues. Estimating the functional principal components/eigenfunctions $\psi_k$'s is one of the most fundamental tasks in FDA and has attracted much attention in the literature (see, e.g., Ramsay and Silverman, 2005). Typically, interest lies in finding a few eigenfunctions that explain a large proportion, e..g, 95% or 99%, of the observed variation. This is equivalent to finding the first few eigenfunctions whose linear combination can well approximate the random functions $X_i$. Computing the eigenfunctions of a symmetric bivariate function is generally not trivial. The common practice is to discretize the estimated covariance function and approximate its eigenfunctions by the corresponding eigenvectors (see, e.g., Yao et al., 2003). In this section, we show that by using FACE we easily obtain the eigendecomposition of the smoothed covariance matrix $\widetilde{\mathbf{K}}$ in equation (1).

We start with the spectral decomposition $(\mathbf{B}^T\mathbf{B})^{-1/2}\mathbf{P}(\mathbf{B}^T\mathbf{B})^{-1/2} = \mathbf{U}\text{diag}(\mathbf{s})\mathbf{U}^T$, where $\mathbf{U}$ is the matrix of eigenvectors and $\mathbf{s}$ is the vector of eigenvalues. Let $\mathbf{A}_S = \mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1/2}\mathbf{U}$. Then $\mathbf{A}_S^T\mathbf{A}_S = \mathbf{I}_c$ which implies that $\mathbf{A}_S$ has orthonormal columns. It follows that $\mathbf{S} = \mathbf{A}_S\mathbf{\Sigma}_S\mathbf{A}_S^T$ with $\mathbf{\Sigma}_S = \{\mathbf{I}_c + \lambda\,\text{diag}(\mathbf{s})\}^{-1}$. Let $\widetilde{\mathbf{Y}} = \mathbf{A}_S^T\mathbf{Y}$ be a $c \times I$ matrix. Then $\widetilde{\mathbf{K}} = \mathbf{A}_S\left(I^{-1}\mathbf{\Sigma}_S\widetilde{\mathbf{Y}}\widetilde{\mathbf{Y}}^T\mathbf{\Sigma}_S\right)\mathbf{A}_S^T$. Thus, only the $c\times c$ dimensional matrix in the parenthesis depends on the smoothing parameter; this observation will lead to a simple spectral decomposition of $\widetilde{\mathbf{K}}$. Indeed, consider the spectral decomposition $I^{-1}\mathbf{\Sigma}_S\widetilde{\mathbf{Y}}\widetilde{\mathbf{Y}}^T\mathbf{\Sigma}_S = \mathbf{A}\mathbf{\Sigma}\mathbf{A}^T$, where $\mathbf{A}$ is the $c \times c$ matrix of eigenvectors and $\mathbf{\Sigma}$ is the $c \times c$ diagonal matrix of eigenvalues. It follows that $\widetilde{\mathbf{K}} = (\mathbf{A}_S\mathbf{A})\mathbf{\Sigma}(\mathbf{A}_S\mathbf{A})^T$ which is the eigendecomposition of $\widetilde{\mathbf{K}}$ and shows that $\widetilde{\mathbf{K}}$ has no more than $c$ nonzero eigenvalues (Proposition 1). Because of the dimension reduction of matrices ($c \times c$ versus $J \times J$), this eigenanalysis of the smoothed covariance matrix is fast. The derivation reveals that through smoothing we obtain a smoothed covariance operator and its associated eigenfunctions. An important consequence is that the number of elements stored in memory is only $O(Jc)$ for FACE, while using other bivariate smoothers requires storing the $J\times J$ dimensional covariance operators. This makes a dramatic difference, allows non-compromised smoothing of covariance matrices, and provides a transparent, easy to use method.

## 3.2 Selection of the Smoothing Parameter

We start with the following result.

**Proposition 1.** *Assume that* $c = o(J)$*. Then the rank of the smoothed covariance matrix* $\widetilde{\mathbf{K}}$ *is at most* $\min(c, I)$*.*

This indicates that the number of knots controls the maximal rank of the smoothed covariance matrix, $\widetilde{\mathbf{K}}$, or equivalently, the number of eigenfunctions that can be extracted from $\widetilde{\mathbf{K}}$. This implies that using an insufficient number of knots may result in severely biased estimates of eigenfunctions and number of eigenfunctions. We propose to use a relatively large number of knots, e.g., 100 knots, to reduce the estimation bias and control overfitting by an appropriate penalty. Note that for high-dimensional data, $J$ can be thousands or more and the dimension reduction by FACE is sizeable. Moreover, as only a small number of functional principal components is typically used in practice, FACE with 100 knots seems adequate for most applications. When the covariance function has a more complex structure or a larger number of functional principal components are needed, one may use a larger number of knots. The simulations and theory in Ruppert (2002) and Wang et al. (2011) show that the number of knots of a P-spline smoother is not important, provided it is above some lower bound. Using more knots than this bound does not degrade the smoother, since the the penalty prevents overfitting.

We select the smoothing parameter by minimizing the pooled generalized cross validation (PGCV), a functional extension of the GCV (Craven and Wahba, 1979),

$$\sum_{i=1}^{I} \|\mathbf{Y}_i - \mathbf{S}\mathbf{Y}_i\|^2 / \{1 - \mathrm{t}r(\mathbf{S})/J\}^2. \tag{2}$$

Here $\|\cdot\|$ is the Euclidean norm of a vector. Criterion (2) was also used in Zhang and Chen

(2007) and could be interpreted as smoothing each sample, $\mathbf{Y}_i$, using the same smoothing parameter. We argue that using criterion (2) is a reasonable practice for covariance estimation. An alternative but computationally demanding method for selecting the smoothing parameter is the leave-one-curve-out cross validation (Yao et al., 2005). The following result indicates that PGCV can be easily calculated in high dimensions.

**Proposition 2.** *The PGCV in expression (2) equals*

$$\frac{\sum_{k=1}^{c} C_{kk}(\lambda s_k)^2/(1+\lambda s_k)^2 - \|\widetilde{\mathbf{Y}}\|_F^2 + \|\mathbf{Y}\|_F^2}{\{1 - J^{-1}\sum_{k=1}^{c}(1+\lambda s_k)^{-1}\}^2},$$

*where $s_k$ is the kth element of $\mathbf{s}$, $C_{kk}$ is the kth diagonal element of $\widetilde{\mathbf{Y}}\widetilde{\mathbf{Y}}^T$, and $\|\cdot\|_F$ is the Frobenius norm.*

The result shows that $\|\mathbf{Y}\|_F^2$, $\|\widetilde{\mathbf{Y}}\|_F^2$, and the diagonal elements of $\widetilde{\mathbf{Y}}\widetilde{\mathbf{Y}}^T$ need to be calculated only once, which requires $O(IJ + cI)$ calculations, and which, *inter alia*, makes the FACE algorithm that we now present fast.

*FACE algorithm:*

*Step 1. Obtain the spectral decomposition* $(\mathbf{B}^T\mathbf{B})^{-1/2}\mathbf{P}(\mathbf{B}^T\mathbf{B})^{-1/2} = \mathbf{U}\text{diag}(\mathbf{s})\mathbf{U}^T$.

*Step 2. Specify* $\mathbf{S} = \mathbf{A}_S\mathbf{\Sigma}_S\mathbf{A}_S^T$ *by calculating and storing* $\mathbf{A}_S = \mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1/2}\mathbf{U}$ *and* $\mathbf{s}$ such that $\mathbf{\Sigma}_S = \{\mathbf{I}_c + \lambda\,\text{diag}(\mathbf{s})\}^{-1}$.

*Step 3. Calculate and store* $\widetilde{\mathbf{Y}} = \mathbf{A}_S^T\mathbf{Y}$.

*Step 4. Select $\lambda$ by minimizing PGCV in expression (2).*

*Step 5. Calculate* $\mathbf{\Sigma}_S = \{\mathbf{I}_c + \lambda\text{diag}(\mathbf{s})\}^{-1}$.

*Step 6. Construct the eigendecomposition* $I^{-1}\mathbf{\Sigma}_S\widetilde{\mathbf{Y}}\widetilde{\mathbf{Y}}^T\mathbf{\Sigma}_S = \mathbf{A}\mathbf{\Sigma}\mathbf{A}^T$.

*Step 7. Construct the eigendecomposition* $\widetilde{\mathbf{K}} = (\mathbf{A}_S\mathbf{A})\mathbf{\Sigma}(\mathbf{A}_S\mathbf{A})^T$.

By exploiting the sparsity of the design matrix $\mathbf{B}$ and the penalty matrix $\mathbf{P}$ which are banded matrices, FACE can be computed with $O\{JI + Jc + \min(I,c)Ic + c^3 + ck_0\}$ computations, where $k_0$ is the number of iterations needed for selecting the smoothing parameter, and the total required memory is $O\left(JI + I^2 + Jc + c^2 + k_0\right)$. See Proposition 3 in the appendix for details. When $c = O(I)$ and $k_0 = o(IJ)$, the computation time of FACE is $O(JI + I^3)$ and $O(JI + I^2)$ memory units are required. As a comparison, if we smooth the covariance operator using other bivariate smoothers, then at least $O(J^2 + JI)$ memory units are required, which dramatically reduces the computational efficiency of those smoothers.

## 3.3 Subject-specific Sampling Points

So far we have only considered the case when the sampling points are the same for all subjects. Assume now for the $i$th sample that we observe $\mathbf{Y}_i = \{Y_i(t_{i1}), \ldots, Y_i(t_{iJ_i})\}^T$, where $t_{ij}$, $j = 1, \ldots, J_i$ can be different across subjects. In this case the empirical estimator of the covariance operator does not have a decomposable form. Consider the scenario when subjects are densely sampled and all $J_i$'s are large. Using the idea from Di et al. (2009), we can undersmooth each $\mathbf{Y}_i$ using, for example, a kernel smoother with a small bandwidth or a regression spline. FACE can then be applied on the under-smoothed estimates evaluated at an equally spaced grid, $\{\widehat{\mathbf{Y}}_1, \ldots, \widehat{\mathbf{Y}}_I\}$. Extension of FACE to the sparse design scenario is beyond the scope of this paper.

## 3.4 Estimating the Scores

Under standard regularity conditions (Karhunen, 1947), $X_i(t)$ can be written as $\sum_{k\geq 1} \xi_{ik}\psi_k(t)$ where $\{\psi_k : k \geq 1\}$ is the set of eigenfunctions of $K$ and $\xi_{ik} = \int_0^1 X_i(s)\psi_k(s)ds$ are the principals scores of $X_i$. It follows that $Y_i(t_j) = \sum_{k\geq 1} \xi_{ik}\psi_k(t_j) + \epsilon_{ij}$. In practice, we may be interested in only the first $N$ eigenfunctions and approximate $Y_i(t_j)$ by $\sum_{k=1}^N \xi_{ik}\psi_k(t_j) + \epsilon_{ij}$. Using the estimated eigenfunctions $\widehat{\psi}_k$'s and eigenvalues $\widehat{\lambda}_k$'s from FACE, the scores of each $X_i$ can be obtained by either numerical integration or as best linear unbiased predictors

(BLUPs). FACE provides fast calculations of scores for both approaches.

Let $\widetilde{\mathbf{Y}}_i$ denote the $i$th column of $\widetilde{\mathbf{Y}}$. Let $\boldsymbol{\xi}_i = (\xi_{i1}, \ldots, \xi_{iN})^T$ and let $\widehat{\mathbf{A}}_N$ denote the first $N$ columns of $\mathbf{A}$ defined in Section 3.1. Let $\boldsymbol{\psi}_k = \{\psi_k(t_1), \ldots, \psi_k(t_J)\}^T$ and $\boldsymbol{\Psi} = [\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_N]$. The matrix $J^{-1/2}\boldsymbol{\Psi}$ is estimated by $\mathbf{A}_S\widehat{\mathbf{A}}_N$. The method of numerical integration estimates $\xi_{ik}$ by $\widehat{\xi}_{ik} = \int_0^1 Y_i(t)\widehat{\psi}_k(t)dt \approx J^{-1}\sum_{j=1}^J Y_i(t_j)\widehat{\psi}_k(t_j)$.

**Theorem 1.** *The estimated principal scores* $\widehat{\boldsymbol{\xi}}_i = (\widehat{\xi}_{i1}, \ldots, \widehat{\xi}_{iN})^T$ *using numerical integration are* $\widehat{\boldsymbol{\xi}}_i = J^{-1/2}\widehat{\mathbf{A}}_N^T\widetilde{\mathbf{Y}}_i, 1 \le i \le I$.

We now show how to obtain the estimated BLUPs for the scores. Let $\epsilon_{ij} = Y_i(t_j) - \sum_{k=1}^N \psi_k(t_j)\xi_{ik}$ and $\boldsymbol{\epsilon}_i = (\epsilon_{i1}, \ldots, \epsilon_{iJ})^T$. Then $\mathbf{Y}_i = \boldsymbol{\Psi}\boldsymbol{\xi}_i + \boldsymbol{\epsilon}_i$. The covariance matrix $\mathrm{var}(\boldsymbol{\xi}_i) = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$ can be estimated by $J^{-1}\widehat{\boldsymbol{\Sigma}}_N = J^{-1}\mathrm{diag}(\widehat{\lambda}_1, \ldots, \widehat{\lambda}_N)$. The variance of $\epsilon_{ij}$ can be estimated by

$$\widehat{\sigma}^2 = I^{-1}J^{-1}\|\mathbf{Y}\|_F^2 - J^{-1}\sum_k \hat{\lambda}_k. \tag{3}$$

**Theorem 2.** *Suppose* $\boldsymbol{\Psi}$ *is estimated by* $J^{1/2}\mathbf{A}_S\widehat{\mathbf{A}}_N$, $\mathrm{var}(\boldsymbol{\xi}_i) = diag(\lambda_1, \ldots, \lambda_N)$ *is estimated by* $\widehat{\boldsymbol{\Sigma}}_N = diag(\widehat{\lambda}_1, \ldots, \widehat{\lambda}_N)$, *and* $\sigma^2$ *is estimated by* $\widehat{\sigma}^2$ *in equation (3). Then the estimated BLUPs of* $\boldsymbol{\xi}_i$ *are given by* $\widehat{\boldsymbol{\xi}}_i = J^{-1/2}\widehat{\boldsymbol{\Sigma}}_N(\widehat{\boldsymbol{\Sigma}}_N + J^{-1}\widehat{\sigma}^2\mathbf{I}_N)^{-1}\widehat{\mathbf{A}}_N^T\widetilde{\mathbf{Y}}_i$, *for* $1 \le i \le I$.

Theorems 1 and 2 provide fast approaches for calculating the principal scores using either numerical integration or BLUPs. These approaches combined with FACE are much faster because they make use of the calculations already done for estimating the eigenfunctions and eigenvalues. When $J$ is large, the scores by BLUPs tend to be very close to those obtained by numerical integration; in the paper we only use numerical integration.

## 4 Extensions of FACE

### 4.1 Structured Functional Data

When analyzing structured functional data such as multilevel, longitudinal, and crossed functional data (Di et al., 2009; Greven et al., 2010; Zipunnikov et al., 2011, 2012; Shou et al., 2013), the covariance matrices have been shown to be of the form $\mathbf{YHY}^T$, where $\mathbf{H}$ is a symmetric matrix; see Shou et al. (2013) for more details. We assume $\mathbf{H}$ is positive semi-definite because otherwise we can replace $\mathbf{H}$ by its positive counterpart. Note that if $\mathbf{H}_1$ is a matrix such that $\mathbf{H}_1\mathbf{H}_1^T = \mathbf{H}$, smoothing $\mathbf{YHY}^T$ can be done by using FACE for the transformed functional data $\mathbf{YH}_1$. This insight is particularly useful for the sleep EEG data, which has two visits and requires multilevel decomposition.

### 4.2 Incomplete Data

To handle incomplete data, such as the EEG sleep data where long portions of the functions are unavailable because a subject is awake, we use an algorithm that iterates between smoothing by FACE and missing data imputation. Initially we substitute 0's for the missing data. Then we apply FACE to obtain predictions of scores and functions. We only obtain scores of the first $N$ components and $N$ is selected by the criteria

$$N = \min\left\{k : \frac{\sum_{j=1}^{k}\lambda_j}{\sum_{j=1}^{\infty}\lambda_j} \geq 0.95\right\}.$$

We replace the missing data with their predictions and re-apply FACE to the updated complete data. We repeat the procedure until convergence is reached. Our experience is that it usually takes less than 20 iterations to reach convergence.

## 5 The SVDS Estimator

A second approach for estimating the eigenfunctions and eigenvalues is to decompose the sample covariance matrix $\widehat{\mathbf{K}}$ and then smooth the eigenvectors using smoothing splines (Craven and Wahba, 1979). First let $\mathbf{U}_y\mathbf{D}_y\mathbf{V}_y^T$ be the singular value decomposition (SVD) of the data matrix $\mathbf{Y}$. Here $\mathbf{U}_y$ is a $J \times I$ matrix with orthonormal columns, $\mathbf{V}_y$ is an $I$-dimensional orthogonal matrix, and $\mathbf{D}_y$ is an $I$-dimensional diagonal matrix. The columns of $\mathbf{U}_y$ contain all the eigenvectors of $\widehat{\mathbf{K}}$ that are associated with non-zero eigenvalues and the set of diagonal elements of $I^{-1}\mathbf{D}_y^2$ contain all the non-zero eigenvalues of $\widehat{\mathbf{K}}$. Thus, obtaining $\mathbf{U}_y$ and $\mathbf{D}_y$ is equivalent to the eigendecomposition of $\widehat{\mathbf{K}}$. Then, we smooth the retained eigenvectors by penalized splines. The SVDS approach avoids the direct decomposition of the sample covariance matrix and is computationally simpler when $I = o(J)$. If we smooth $N$ eigenvectors, SVDS requires $O(I^2J + NJ)$ computations, where $O(I^2J)$ computations are for the SVD step and $O(NJ)$ computations are for the smoothing step. Note that, because $N < I$, the computation time for the smoothing step is negligible compared to that of the SVD step. We will compare SVDS and FACE in terms of performance and computation time in the simulation study.

## 6 Simulation

We consider three simulation studies. In the first study we use moderately high-dimensional data contaminated with noise. We let $J = 3,000$ and $I = 50$, which are roughly the dimensions of the EEG data in Section 7. We use both FACE and SVDS; we did not evaluate other bivariate smoothers because we were unable to run them on such dimensions in a reasonably short time. In the second study we consider functional data where portions of the observed functions are missing at random. This simulation is directly inspired by our EEG data where long portions of the functions are missing. In the last study we assess the computation time of FACE and compare it with that of SVDS. We also provide the computation time of the sandwich smoother (Xiao et al., 2013). We use R code that is made available with this paper. All simulations are run on modest, widely available computational resources: a duo core 2.4 GHz Mac (2008 model) with 4 gigabytes of random access memory.

## 6.1 Complete Data

We generate data from the model $Y_i(t_j) = \sum_{k=1}^{N} \xi_{ik}\psi_k(t_j) + \epsilon_{ij}$, where $\xi_{ik} \sim \mathcal{N}(0, \lambda_k)$ and $\epsilon_{ij} \sim \mathcal{N}(0, \sigma^2)$ are mutually independent for $1 \leq i \leq I, 1 \leq j \leq J$. We let $I = 50$, $N = 4$, $\lambda_k = 0.5^{k-1}$ for $k = 1, \dots, 4$. We consider two different sets of eigenfunctions

$$\begin{aligned}
&\text{Case 1:} \quad \left\{\sqrt{2}\sin(2\pi t), \sqrt{2}\cos(2\pi t), \sqrt{2}\sin(4\pi t), \sqrt{2}\cos(4\pi t)\right\}, \\
&\text{Case 2:} \quad \left\{1, \sqrt{3}(2t-1), \sqrt{5}(6t^2-6t+1), \sqrt{7}(20t^3-30t^2+12t-1)\right\},
\end{aligned}$$

which are measured at $\{1/J, 2/J, \dots, 1\}$ and $J = 3,000$. The above two sets of eigenfunctions were also used in the simulation studies in Di et al. (2009), Greven et al. (2010), and Zipunnikov et al. (2011). We let $\sigma = 2$. In all simulations, we use cubic B-splines and a second-order difference penalty (Eilers and Marx, 1996) to construct the univariate smoother matrix.

To make the comparison simple, we assume both FACE and SVDS select 4 eigenfunctions/eigenvectors. We use 100 knots for FACE. Figure 1 displays, for two simulated data sets for each case, the true and estimated eigenfunctions using SVDS and FACE.

We see from Figure 1 that the estimated eigenfunctions are very similar for both methods. The results are expected as both methods are designed to account for the noise in the data and the discrepancy between the estimated eigenfunctions and the true ones is mainly due to the variation in the random functions. Table 1 provides the mean integrated squared errors (MISE) of the estimated eigenfunctions indicating a slightly better performance of FACE.

Figure 2 shows boxplots of estimated eigenvalues that are centered and standardized, $(\widehat{\lambda}_k - \lambda_k)/\lambda_k$. Estimates of the eigenvalues by SVDS tend to overestimate, especially for small eigenvalues. In case 1, on average, SVDS overestimates the four eigenvalues by 11%, 15%, 28% and 55%, respectively. In contrast, the FACE estimators are much closer to the true eigenvalues. Table 2 below provides the average mean squared errors (AMSEs) of the estimated eigenvalues and indicates that the estimates of eigenvalues using FACE have smaller AMSEs.

## 6.2 Incomplete data

In Section 4.2 we extended FACE to incomplete data, and here we illustrate the extension with a simulation. We use the same simulation setting in Section 6.1 except that for each subject we allow for portions of observations missing at random. For simplicity we fix the length of each portion so that $0.065J$ consecutive observations are missing. We allow one subject to miss either 1, 2, or 3 portions with equal probabilities so that in expectation 13% of the data are missing. Note that the real data we will consider later also has about 13% measurements missing and they occur as missing segments as here.

In Figure 1, the green dotted lines are the resulting estimates for incomplete data; in Figure 2, boxplots of the estimated eigenvalues are also shown. The MISEs of the estimated eigenfunctions and the AMSEs of the estimated eigenvalues appear in Tables 1 and 2, respectively. The simulation results show that the performance of FACE degrades only marginally.

## 6.3 Computation Time

We record the computation time of FACE for various combinations of $J$ and $I$. All other settings remain the same as in the first simulation study and we use the eigenfunctions from case 1. For comparison the computation times of SVDS and the sandwich smoother

(Xiao et al., 2013) are also given. Table 3 summarizes the results and shows that FACE is fast even with high-dimensional data while the computation time of the sandwich smoother increases dramatically with $J$, the dimension of the problem. For example it took FACE about 5 seconds to smooth a 10,000 by 10,000 dimensional matrix for 500 subjects, while the sandwich smoother did not run on our computer.

FACE is faster than SVDS in most cases, which is expected because, when $c = O(I) = o(J)$, FACE requires $O(JI + I^3)$ computations, much less than the $O(JI^2)$ computations needed by SVDS. The only exception where SVDS is faster is when $I = 50$ and $c = 500$, which is also reasonable because in this case FACE requires $O(c^3)$ computations, more than the $O(JI^2)$ computations for SVDS.

Although we did not run FACE on ultrahigh-dimensional data, we can obtain a rough estimate of the computation time by the formula $O(JI + Ic^2)$. Table 3 shows that FACE with 500 knots takes 6 seconds on data with $(J, I) = (10000, 500)$. For data with $J$ equal to 100,000 and $I$ equal to 1,000, FACE with 1000 knots should take about two minutes to compute, without taking into account the time for loading data into the computer memory. Note that the dominant term in the formula $O(JI + Ic^2)$ is $Ic^2$. It turned out FACE took 2.5 minutes; as a comparison, SVDS needed 8 minutes.

## 7 Example

The Sleep Heart Health Study (SHHS) is a large-scale study of sleep and its association with health-related outcomes. Thousands of subjects enrolled in the SHHS underwent two in-home polysomnograms (PSGs) at multiple visits. Two-channel electroencephalographs (EEG), part of the PSG, were collected at a frequency of 125Hz, or 125 observations per second for each subject, visit and channel. We model the proportion of $\delta$-power which is a summary measure of the spectrum of the EEG signal. More details on $\delta$-power can be found in Crainiceanu et al. (2009) and Di et al. (2009). The data contain 51 subjects with sleep-disordered breathing (SDB) and 51 matched controls; see Crainiceanu et al. (2012) and Swihart et al. (2012) for details on how the pairs were matched. An important feature of the EEG data is that long consecutive portions of observations, which indicate wake periods, are missing. In total about 13% of the data is missing.

Similar to Crainiceanu et al. (2012), we consider the following statistical model. The data for proportion of $\delta$-power are pairs of curves $\{Y_{iA}(t), Y_{iC}(t)\}$, where $i$ denotes subject, $t = t_1, \dots, t_J$ $(J = 2,880)$ denotes the time measured in 5-second intervals in a 4-hour sleep interval from sleep onset, $A$ stands for apneic and $C$ stands for control. The model is

$$\begin{cases} Y_{iA}(t) = \mu_A(t) + X_i(t) + U_{iA}(t) + \epsilon_{iA}(t) \\ Y_{iC}(t) = \mu_C(t) + X_i(t) + U_{iC}(t) + \epsilon_{iC}(t) \end{cases} \tag{4}$$

where $\mu_A(t)$ and $\mu_C(t)$ are mean functions of proportions of $\delta$-power, $X_i(t)$ is a functional process with mean 0 and continuous covariance operator $K_X(\cdot,\cdot)$, $U_{iA}(t)$ and $U_{iC}(t)$ are functional processes with mean 0 and continuous covariance operator $K_U(\cdot,\cdot)$, and $\epsilon_{iA}(t), \epsilon_{iC}(t)$ are measurement errors with mean 0 and variance $\sigma^2$. The random processes $X_i, U_{iA}, U_{iC}, \epsilon_{iA}$ and $\epsilon_{iC}$ are assumed to be mutually independent. Here $X_i$ accounts for the between-pair correlation of the data while $U_{iA}$ and $U_{iC}$ model the within-pair correlation. The Multilevel Functional Principal Component Analysis (MFPCA) (Di et al., 2009) can be used to analyze data with model (4). One crucial step of MFPCA is to smooth two estimated covariance operators which in this example are $2880 \times 2880$ matrices.

Smoothing large covariance operators of dimension $2880 \times 2880$ can be computationally expensive. We tried bivariate thin plate regression splines and used the R function '*bam*' in the *mgcv* package (Wood, 2013) with 35 equally-spaced knots for each axis. The smoothing parameter was automatically selected by '*bam*' with the option 'GCV.cp'. Running time for thin plate regression splines was three hours. Because the two covariance operators take the form in Section 4.1 (see the details in Appendix B), we applied FACE, which ran in less than 10 seconds with 100 knots. Note that we also tried thin plate splines with 100 knots in *mgcv*, which was still running after 10 hours. Figure 3 displays the first four eigenfunctions for $K_X$ and $K_U$, using both methods. As a comparison, the eigenfunctions using SVDS are also shown. For the SVDS method, to handle incomplete data the SVD step was replaced by a brute-force decomposition of the two $2880 \times 2880$ covariance operators. Figure 3 shows that the top eigenfunctions obtained from the two bivariate smoothing methods are quite different, except for the first eigenfunctions on the top row. The estimated eigenfunctions using FACE in general resemble those by SVDS with some subtle differences, while thin plate splines in this example seem to over-smooth the data, probably because we were forced to use a smaller number of knots.

Table 4 provides estimated eigenvalues of $K_X$ and $K_U$. Compared to FACE, thin plate splines over-shrink significantly the eigenvalues, especially those of the between pair covariance. The results from FACE in Table 4 show that the proportion of variability explained by $K_X$, the between-pair variation, is $15.81/(15.81 + 28.67) \approx 35.5\%$.

## 8 Discussion

In this paper we developed a fast covariance estimation (FACE) method that could significantly alleviate the computational difficulty of bivariate smoothing and eigendecomposition of large covariance matrices in FPCA for high-dimensional data. Because bivariate smoothing and eigendecomposition of covariance matrices are integral parts of FPCA, our method could increase the scope and applicability of FPCA for high-dimensional data. For instance, with FACE, one may consider incorporating high-dimensional functional predictors into the penalized functional regression model of Goldsmith et al. (2011).

The proposed FACE method can be regarded as a two-step procedure (see, e.g., Besse and Ramsay, 1986; Ramsay and Dalzell, 1991; Besse et al., 1997; Cardot, 2000; and Zhang and Chen, 2007). Indeed, if we first smooth data at the subject level $\widehat{\mathbf{Y}}_i = \mathbf{S}\mathbf{Y}_i, i = 1, \ldots, I$, then it is easy to show that the empirical covariance estimator of the $\widehat{\mathbf{Y}}_i$ is equal to $\widetilde{\mathbf{K}}$. There are, however, important computational differences between FACE and the current two-step procedures. First, the fast algorithm in Section 3.2 enables FACE to select efficiently the smoothing parameter. Second, FACE provides the eigendecomposition as described in Section 3.1, whereas a two-step approach might require sub-sampling of the smooth paths when $J$ is large. Third, FACE can be easily extended for incomplete data where long consecutive portions of data are missing while it is unclear how a two-step procedure could be used for such data.

The second approach, SVDS, is very simple and reasonable, and we would recommend it if FACE were not available. However, in our simulation study FACE proved to be faster and more accurate than SVDS, and SVDS becomes computationally impractical for high-dimensional functional observations with missing data.

## Acknowledgement

This work was supported by Grant Number R01EB012547 from the National Institute of Biomedical Imaging And Bioengineering and Grant Number R01NS060910 from the National Institute of Neurological Disorders and Stroke. This work represents the opinions of the researchers and not necessarily that of the granting organizations.

## A Appendix: Proofs

*Proof of Proposition 1*: The design matrix $\mathbf{B}$ is of full rank (Xiao et al., 2012). Hence $\mathbf{B}^T\mathbf{B}$ is invertible and $\mathbf{A}_S$ is of rank $c$. $\mathbf{\Sigma}_S$ is a diagonal matrix with all elements greater than 0 and $\widetilde{\mathbf{Y}}$ is of rank at most $\min(c, I)$. Hence $\widetilde{\mathbf{K}} = \mathbf{A}_S \left( I^{-1}\mathbf{\Sigma}_S\widetilde{\mathbf{Y}}\widetilde{\mathbf{Y}}^T\mathbf{\Sigma}_S \right) \mathbf{A}_S^T$ has a rank at most

$\min(c, I)$ and the proposition follows.

*Proof of Proposition 2*: First of all, $tr(\mathbf{S}) = tr(\mathbf{\Sigma}_S)$ which is easy to calculate. We now compute $\sum_{i=1}^I \|\mathbf{Y}_i - \mathbf{S}\mathbf{Y}_i\|^2$. Because $\|\mathbf{Y}_i - \mathbf{S}\mathbf{Y}_i\|^2 = \mathbf{Y}_i^T(\mathbf{S} - \mathbf{I}_J)^2\mathbf{Y}_i = tr\{(\mathbf{S} - \mathbf{I}_J)^2\mathbf{Y}_i\mathbf{Y}_i^T\}$,

$$\sum_{i=1}^I \|\mathbf{Y}_i - \mathbf{S}\mathbf{Y}_i\|^2 = tr\left\{(\mathbf{S} - \mathbf{I}_J)^2 \sum_{i=1}^I \mathbf{Y}_i\mathbf{Y}_i^T\right\} = tr\left\{(\mathbf{S} - \mathbf{I}_J)^2\mathbf{Y}\mathbf{Y}^T\right\}.$$

It can be shown that $\mathbf{S}^2 = \mathbf{A}_S\mathbf{\Sigma}_S^2\mathbf{A}_S^T$. Hence $tr(\mathbf{S}^2\mathbf{Y}\mathbf{Y}^T) = tr(\mathbf{Y}^T\mathbf{S}^2\mathbf{Y}) = tr(\widetilde{\mathbf{Y}}^T\mathbf{\Sigma}_S^2\widetilde{\mathbf{Y}}) = tr(\mathbf{\Sigma}_S^2\widetilde{\mathbf{Y}}\widetilde{\mathbf{Y}}^T)$. Similarly, we derive $tr(\mathbf{S}\mathbf{Y}\mathbf{Y}^T) = tr(\mathbf{\Sigma}_S\widetilde{\mathbf{Y}}\widetilde{\mathbf{Y}}^T)$. We have $tr(\mathbf{Y}\mathbf{Y}^T) = \|\mathbf{Y}\|_F^2$. It follows that

$$\sum_{i=1}^I \|\mathbf{Y}_i - \mathbf{S}\mathbf{Y}_i\|^2 = tr\left\{(\mathbf{\Sigma}_S - \mathbf{I}_c)^2\widetilde{\mathbf{Y}}\widetilde{\mathbf{Y}}^T\right\} - \|\widetilde{\mathbf{Y}}\|_F^2 + \|\mathbf{Y}\|_F^2.$$

**Proposition 3.** *The computation time of FACE is $O\left\{JI + Jc + \min(I, c)Ic + c^3 + ck_0\right\}$, where $k_0$ is the number of iterations needed for selecting the smoothing parameter (see Section 3.2), and the total required computer memory is $O\left(JI + I^2 + Jc + c^2 + k_0\right)$ memory units.*

*Proof of Proposition 3*: We need to compute or store the following quantities: $\mathbf{Y}$, $\mathbf{B}$, $\mathbf{B}^T\mathbf{B}$, $(\mathbf{B}^T\mathbf{B})^{-1/2}$, $\mathbf{P}$, $(\mathbf{B}^T\mathbf{B})^{-1/2}\mathbf{P}(\mathbf{B}^T\mathbf{B})^{-1/2}$, $\mathbf{A}_S$, $\widetilde{\mathbf{Y}}$, $\widetilde{\mathbf{Y}}\widetilde{\mathbf{Y}}^T$, $\mathbf{A}$, $\mathbf{U}$, $\mathbf{A}_S\mathbf{A}$, We will use the fact that both $\mathbf{B}$ and $\mathbf{P}$ are banded matrices. For the computational complexity, $\mathbf{B}^T\mathbf{B}$ requires $O(J)$ computations, $\mathbf{P}$ requires $O(c)$ computations; $\widetilde{\mathbf{Y}}\widetilde{\mathbf{Y}}^T$ requires $O\left\{\min(I, c)Ic\right\}$ computations; $(\mathbf{B}^T\mathbf{B})^{-1/2}$, $(\mathbf{B}^T\mathbf{B})^{-1/2}\mathbf{P}(\mathbf{B}^T\mathbf{B})^{-1/2}$, $\mathbf{A}$ (through the eigendecomposition of $\mathbf{\Sigma}_S\widetilde{\mathbf{Y}}\widetilde{\mathbf{Y}}^T\mathbf{\Sigma}_S$), and $\mathbf{U}$ (through the eigendecomposition of $(\mathbf{B}^T\mathbf{B})^{-1/2}\mathbf{P}(\mathbf{B}^T\mathbf{B})^{-1/2}$) require $O(c^3)$ computations; $\mathbf{A}_S = \mathbf{B}\left\{(\mathbf{B}^T\mathbf{B})^{-1/2}\mathbf{U}\right\}$, and $\mathbf{A}_S\mathbf{A} = \mathbf{B}\left\{(\mathbf{B}^T\mathbf{B})^{-1/2}\mathbf{U}\mathbf{A}\right\}$ require $O(Jc + c^3)$ computations; $\widetilde{\mathbf{Y}} = \left\{(\mathbf{B}^T\mathbf{B})^{-1/2}\mathbf{U}\right\}^T\left(\mathbf{B}^T\mathbf{Y}\right)$ require $O(JI + c^3)$ computations. So in total, $O(JI + c^3)$ computations are required. For the memory burden, the loading of $\mathbf{Y}$ requires $O(JI)$ memory units, objects $\mathbf{B}$ and $\mathbf{A}_S\mathbf{A}$ requires $O(Jc)$ memory units, and other objects require $O(c^2)$ memory units.

*Proof of Theorem 1*: We have $\widehat{\boldsymbol{\xi}}_i = J^{-1/2}(\mathbf{A}_S\widehat{\mathbf{A}}_N)^T\mathbf{Y}_i = J^{-1/2}\widehat{\mathbf{A}}_N^T(\mathbf{A}_S^T\mathbf{Y}_i) = J^{-1/2}\widehat{\mathbf{A}}_N^T\widetilde{\mathbf{Y}}_i$.

*Proof of Theorem 2*: Let $\widetilde{\mathbf{A}}_N$ denote the first $N$ columns of $\mathbf{A}_S\mathbf{A}$, then $\widetilde{\mathbf{A}}_N = \mathbf{A}_S\widehat{\mathbf{A}}$. The estimated BLUPs for $\boldsymbol{\xi}_i$ (Ruppert et al., 2003) is

$$\widehat{\boldsymbol{\xi}}_i = J^{-1/2}\widehat{\boldsymbol{\Sigma}}_N\widetilde{\mathbf{A}}_N^T\left(\widetilde{\mathbf{A}}_N\widehat{\boldsymbol{\Sigma}}_N\widetilde{\mathbf{A}}_N^T + J^{-1}\widehat{\sigma}^2\mathbf{I}_J\right)^{-1}\mathbf{Y}_i.$$

The inverse matrix in the above equality can be replaced by the following (Seber (2007), page 309, equality b(i)),

$$\left(\widehat{\mathbf{A}}_N\widehat{\boldsymbol{\Sigma}}_N\widetilde{\mathbf{A}}_N^T + J^{-1}\widehat{\sigma}^2\mathbf{I}_J\right)^{-1} = \frac{J}{\widehat{\sigma}^2}\left\{\mathbf{I}_N - \frac{J}{\widehat{\sigma}^2}\widetilde{\mathbf{A}}_N\left(\widehat{\boldsymbol{\Sigma}}_N^{-1} + \frac{J}{\widehat{\sigma}^2}\mathbf{I}_N\right)^{-1}\widetilde{\mathbf{A}}_N^T\right\}.$$

It follows that

$$\begin{aligned}\widehat{\boldsymbol{\xi}} &= J^{-1/2}\frac{J}{\widehat{\sigma}^2}\widehat{\boldsymbol{\Sigma}}\left\{\mathbf{I}_N - \frac{J}{\widehat{\sigma}^2}\left(\widehat{\boldsymbol{\Sigma}}_N^{-1} + \frac{J}{\widehat{\sigma}^2}\mathbf{I}_N\right)^{-1}\right\}\widehat{\mathbf{A}}_N^T\widetilde{\mathbf{Y}}_i \\ &= J^{-1/2}\widehat{\boldsymbol{\Sigma}}_N\left(\widehat{\boldsymbol{\Sigma}}_N + J^{-1}\widehat{\sigma}^2\mathbf{I}_N\right)^{-1}\widehat{\mathbf{A}}_N^T\widetilde{\mathbf{Y}}_i.\end{aligned}$$

## B Appendix: Empirical covariance operators for $K_X$ and $K_U$

Let $I$ denote the number of pairs of cases and controls. For simplicity, we assume estimates of $\mu_A(t)$ and $\mu_C(t)$ have been subtracted from $Y_{iA}$ and $Y_{iC}$, respectively. Let $\mathbf{Y}_{iA} = (Y_{iA}(t_1), \ldots, Y_{iA}(t_T))^T$ and $\mathbf{Y}_{iC} = (Y_{iC}(t_1), \ldots, Y_{iC}(t_J))^T$. By Zipunnikov et al. (2011), we have estimates of the covariance operators,

$$\widehat{\mathbf{K}}_X = \frac{1}{2I}\sum_{i=1}^{I}\left(\mathbf{Y}_{iA}\mathbf{Y}_{iC}^T + \mathbf{Y}_{iC}\mathbf{Y}_{iA}^T\right),$$

and

$$\widehat{\mathbf{K}}_U = \frac{1}{2I}\sum_{i=1}^{I}\left(\mathbf{Y}_{iA} - \mathbf{Y}_{iC}\right)\left(\mathbf{Y}_{iA} - \mathbf{Y}_{iC}\right)^T.$$

Let $\mathbf{Y}_A = [\mathbf{Y}_{1A}, \ldots, \mathbf{Y}_{nA}]$, $\mathbf{Y}_C = [\mathbf{Y}_{1C}, \ldots, \mathbf{Y}_{nC}]$ and $\mathbf{Y} = [\mathbf{Y}_A, \mathbf{Y}_C]$. Then $\mathbf{Y}$ is of dimension $J \times 2I$. It can be shown that $\widehat{\mathbf{K}}_X = \mathbf{Y}\mathbf{H}_X\mathbf{Y}^T$ and $\widehat{\mathbf{K}}_U = \mathbf{Y}\mathbf{H}_U\mathbf{Y}^T$, where

$$\mathbf{H}_X = \frac{1}{2I}\begin{pmatrix} \mathbf{0}_I & \mathbf{I}_I \\ \mathbf{I}_I & \mathbf{0}_I \end{pmatrix},\ \mathbf{H}_U = \frac{1}{2I}\begin{pmatrix} \mathbf{I}_I & -\mathbf{I}_I \\ -\mathbf{I}_I & \mathbf{I}_I \end{pmatrix}.$$

## References

Bai, J., B. Caffo, T. Glass, and C. Crainiceanu (2012). Movelets: a dictionary of movement. *Electronic J. Statist. 6*, 559–578.

Besse, P., H. Cardot, and F. Ferraty (1997). Simultaneous nonparametric regressions of unbalanced longitudinal data. *Comput. Statist. Data Anal. 24*, 255–270.

Besse, P. and J. O. Ramsay (1986). Principal components analysis of sampled functions. *Psychometrika 51*, 285–311.

Capra, W. and H. Müller (1997). An accelerated-time model for response curves. *J. Amer. Statist. Assoc. 92*, 72–83.

Cardot, H. (2000). Nonparametric estimation of smoothed principal components analysis of sampled noisy functions. *J. Nonparametr. Statist. 12*, 503–538.

Crainiceanu, C., P. Reiss, J. Goldsmith, L. Huang, L. Huo, F. Scheipl, S. Greven, J. Harezlak, M. Kundu, Y. Zhao, and M. Mclean (2013). R package *mgcv*: Methodology for regretssion with functional data (version 0.1-7). URL:`http://cran.r-project.org/web/packages/refund/index.html`.

Crainiceanu, C., A. Staicu, and C. Di (2009). Generalized Multilevel Functional Regression. *J. Amer. Statist. Assoc. 104*, 1550–1561.

Crainiceanu, C., A. Staicu, S. Ray, and N. Punjabi (2012). Bootstrap-based inference on the difference in the means of two correlated functional processes. *Statist. Med. 31*, 3223–3240.

Craven, P. and G. Wahba (1979). Smoothing noisy data with spline functions. *Numer. Math. 31*, 377–403.

Dauxois, J., A. Pousse, and Y. Romain (1982). Simultaneous nonparametric regressions of unbalanced longitudinal data. *J. Multivariate Anal. 12*, 136–154.

Di, C., C. M. Crainiceanu, B. S. Caffo, and N. Punjabi (2009). Multilevel functional principal component analysis. *Ann. Appl. Statist. 3*, 458–488.

Eilers, P. and B. Marx (1996). Flexible smoothing with B-splines and penalties (with Discussion). *Statist. Sci. 11*, 89–121.

Eilers, P. and B. Marx (2003). Multivariate calibration with temperature interaction using two-dimensional penalized signal regression. *Chemometrics and Intelligent Laboratory Systems 66*, 159–174.

Goldsmith, J., J. Bobb, C. Crainiceanu, B. Caffo, and D. Reich (2011). Penalized functional regression. *J. Comput. Graph. Statist. 20*, 830–851.

Goldsmith, J., C. Crainiceanu, B. Caffo, and D. Reich (2012). Longitudinal penalized functional regression for cognitive outcomes on neuronal tract measurements. *J. R. Statist. Soc. C 61*, 453–469.

Greven, S., C. Crainiceanu, B. Caffo, and D. Reich (2010). Longitudinal functional principal component. *Electronic J. Statist. 4*, 1022–1054.

Karhunen, K. (1947). Uber lineare methoden in der wahrscheinlichkeitsrechnung. *Annales Academie Scientiarum Fennicae 37*, 1–79.

Kneip, A. (1994). Nonparametric estimation of common regressors for similar curve data. *Ann. Statist. 22*, 1386–1427.

Marx, B. and P. Eilers (2005). Multidimensional Penalized Signal Regression. *Technometrics 47*, 13–22.

Ramsay, J. and C. J. Dalzell (1991). Some tools for functional data analysis (with Discussion). *J. R. Statist. Soc. B 53*, 539–572.

Ramsay, J. and B. Silverman (2005). *Functional data analysis.* New York: Springer.

Ramsay, J. and B. W. Silverman (2002). *Applied Functional data analysis: Methods and Case Studies.* New York: Springer.

Rice, J. and B. Silverman (1991). Estimating the mean and covariance structure nonparametrically when the data are curves. *J. R. Statist. Soc. B 53*, 233–243.

Ruppert, D. (2002). Selecting the number of knots for penalized splines. *J. Comput. Graph. Statist. 1*, 735–757.

Ruppert, D., M. Wand, and R. Carroll (2003). *Semiparametric Regression.* Cambridge: Cambridge University Press.

Seber, G. (2007). *A Matrix Handbook for Statisticians.* New Jersey: Wiley-Interscience.

Shou, H., V. Zipunnikov, C. Crainiceanu, and S. Greven (2013). Structured functional principal component analysis. Available at `http://arxiv.org/pdf/1304.6783.pdf`.

Staniswalis, J. and J. Lee (1998). Nonparametric regression analysis of longitudinal data. *J. Amer. Statist. Assoc. 93*, 1403–1418.

Swihart, B., B. Caffo, C. Crainiceanu, and N. Punjabi (2012). Models of sleep hypnograms scalable to epidemiological studies. *Tentatively accepted in Statistics in Medicine*.

Wang, X., J. Shen, and D. Ruppert (2011). Some asymptotic results on generalized penalized spline smoothing. *Electronic J. Statist. 4*, 1–17.

Wood, S. (2003). Thin plate regression splines. *J. R. Statist. Soc. B 65*, 95–114.

Wood, S. (2013). R package *mgcv*: Mixed GAM computation vehicle with GCV/AIC/REML, smoothese estimation (version 1.7-24). URL:`http://cran.r-project.org/web/packages/mgcv/index.html`.

Xiao, L., Y. Li, T. Apanasovich, and D. Ruppert (2012). Local asymptotics of *P*-splines. Available at `http://arxiv.org/abs/1201.0708v3`.

Xiao, L., Y. Li, and D. Ruppert (2013). Fast bivariate *P*-splines: the sandwich smoother. *J. R. Statist. Soc. B 75*, 577–599.

Yao, F., H. Müller, A. Clifford, S. Dueker, J. Follett, Y. Lin, B. Buchholz, and J. Vogel (2003). Shrinkage estimation for functional principal component scores with application to the population kinetics of plasma folate. *Biometrics 20*, 852–873.

Yao, F., H. Müller, and J. Wang (2005). Functional data analysis for sparse longitudinal data. *J. Amer. Statist. Assoc. 100*, 577–590.

Zhang, J. and J. Chen (2007). Statistical inferences for functional data. *Ann. Statist. 35*, 1052–1079.

Zipunnikov, V., B. S. Caffo, C. M. Crainiceanu, D. Yousem, C. Davatzikos, and B. Schwartz (2011). Multilevel functional principal component analysis for high-dimensional data. *J. Comput. Graph. Statist. 20*, 852–873.

Zipunnikov, V., S. Greven, B. S. Caffo, and C. Crainiceanu (2012). Longitudinal high-dimensional data analysis. Available at `http://biostats.bepress.com/jhubiostat/paper234/`.

Table 1: Simulation study. 100×MISEs of the SVDS and FACE methods for estimating the eigenfunctions. The number of knots for the FACE method is 100. The incomplete data has about 13% observations missing.

| | Eigenfunction | SVDS | FACE | FACE (incomplete data) |
|---|---|---|---|---|
| Case 1 | 1 | 6.23 | 6.18 | 6.50 |
| | 2 | 11.66 | 10.51 | 12.00 |
| | 3 | 13.28 | 12.41 | 13.14 |
| | 4 | 7.63 | 8.14 | 7.86 |
| Case 2 | 1 | 9.31 | 9.26 | 9.42 |
| | 2 | 13.40 | 13.11 | 13.61 |
| | 3 | 12.16 | 9.86 | 12.77 |
| | 4 | 9.11 | 7.16 | 9.72 |

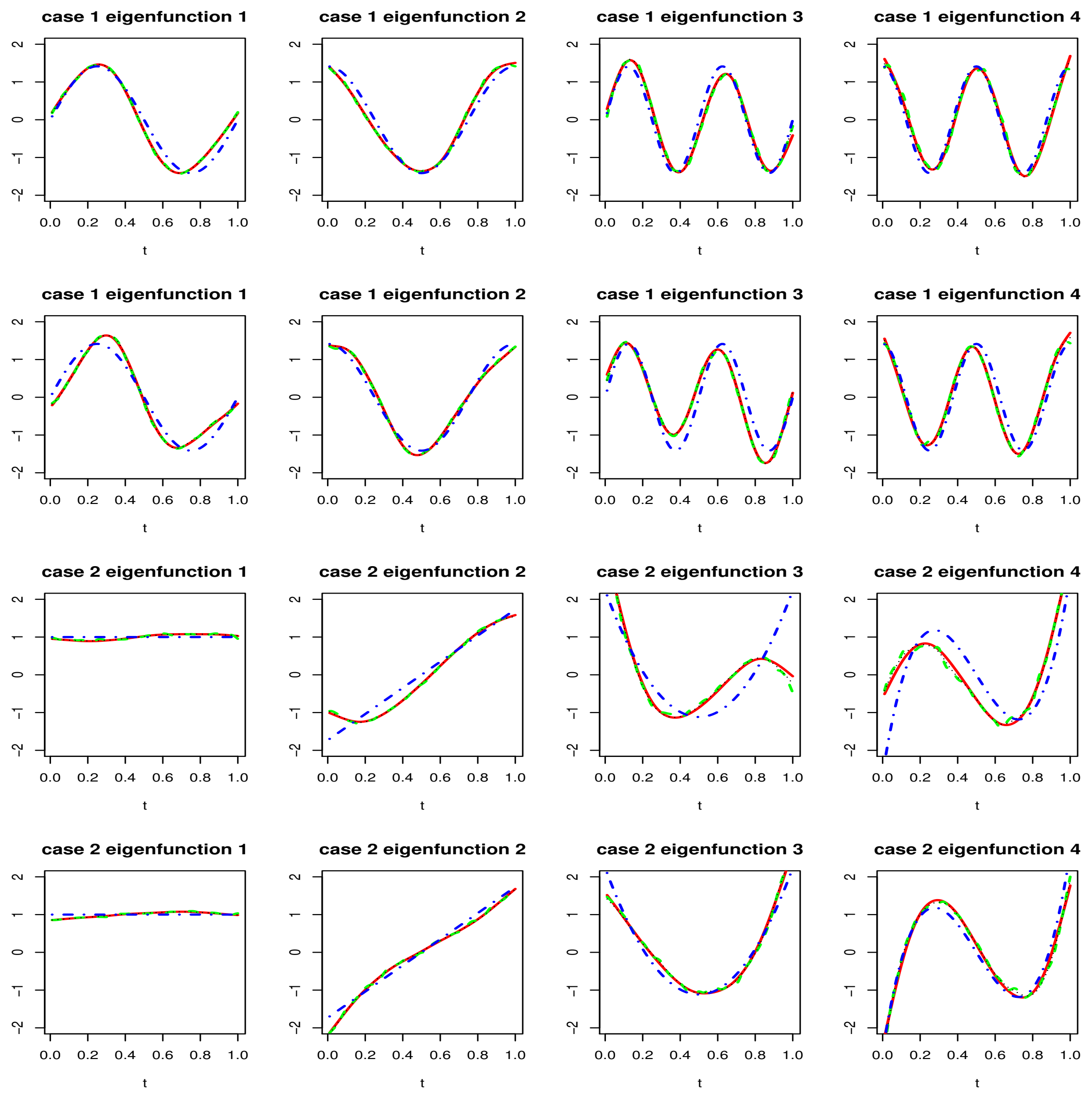

Figure 1: Simulation study. True and estimated eigenfunctions for each case with two simulated data sets. Each row corresponds to one simulated data set. Each box shows the true eigenfunction (blue dot-dashed lines), the estimated eigenfunction using FACE (red solid lines), the estimated eigenfunction using FACE with incomplete data (green dashed lines), and the estimated eigenfunction using SVDS (black dotted lines). Note that the three estimated eigenfunctions are similar and hence one plot may be superimposed over another.

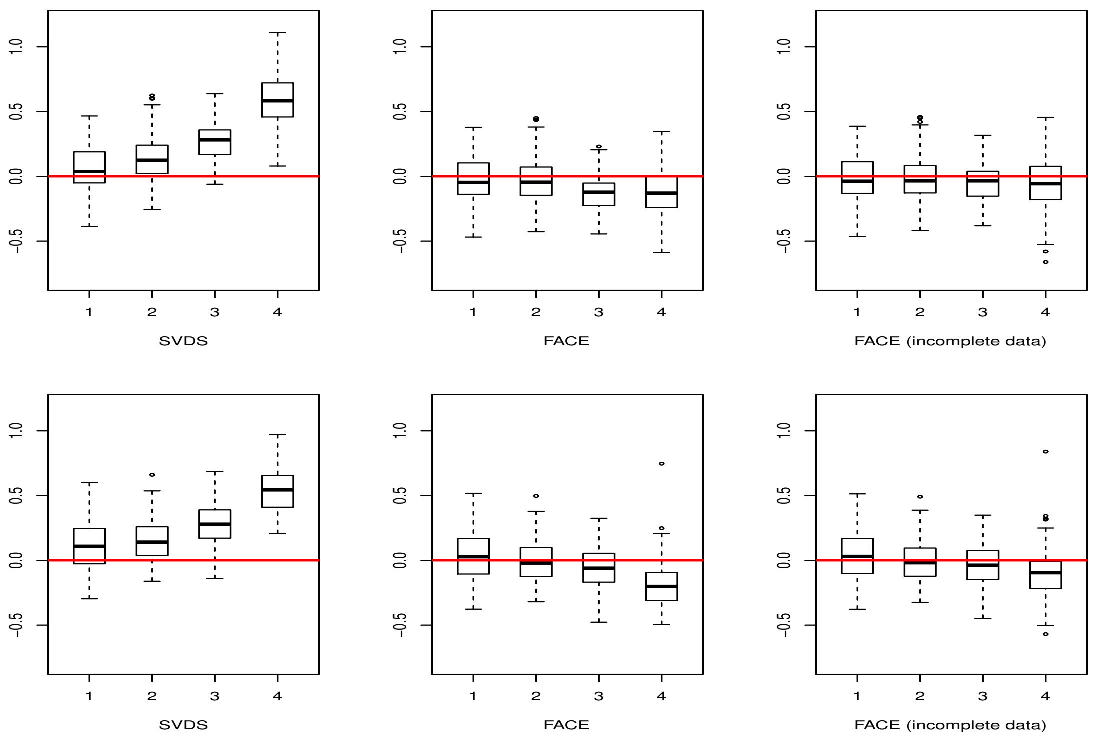

Figure 2: Simulation study. Boxplots of the centered and standardized estimated eigenvalues, $(\widehat{\lambda}_k - \lambda_k)/\lambda_k$. The top panel is for case 1 and the bottom panel is for case 2. The zero is shown by the solid red line.

Table 2: Simulation study. 100×AMSEs of the SVDS and FACE methods for estimating the eigenvalues. The number of knots for FACE is 100. The incomplete data has about 13% observations missing.

| | Eigenvalue | SVDS | FACE | FACE (incomplete data) |
|---|---|---|---|---|
| Case 1 | 1 | 4.00 | 3.39 | 7.34 |
| | 2 | 1.27 | 0.82 | 1.61 |
| | 3 | 0.62 | 0.22 | 0.41 |
| | 4 | 0.62 | 0.07 | 0.08 |
| Case 2 | 1 | 5.08 | 4.05 | 5.89 |
| | 2 | 1.29 | 0.69 | 1.41 |
| | 3 | 0.66 | 0.20 | 0.34 |
| | 4 | 0.54 | 0.11 | 0.10 |

Table 3: Simulation study. Computation time (in seconds) of the SVDS and FACE methods averaged over 100 data sets on 2.4GHz Mac computers with 4 gigabytes of random access memory. The computation time of the sandwich smoother is also provided except for $J = 10,000$ and is averaged over 10 datasets only.

| $J$ | $I$ | FACE 100 knots | FACE 500 knots | SVDS | Sandwich 100 knots | Sandwich 500 knots |
|---|---|---|---|---|---|---|
| $3,000$ | 50 | 0.11 | 2.40 | 0.24 | 86.89 | 124.78 |
| | 500 | 0.52 | 4.08 | 7.36 | 93.94 | 131.82 |
| $5,000$ | 50 | 0.16 | 2.50 | 0.40 | 433.33 | 467.83 |
| | 500 | 0.81 | 4.39 | 11.13 | 509.67 | 570.79 |
| $10,000$ | 50 | 0.26 | 2.70 | 0.82 | - | - |
| | 500 | 1.54 | 5.17 | 22.84 | - | - |

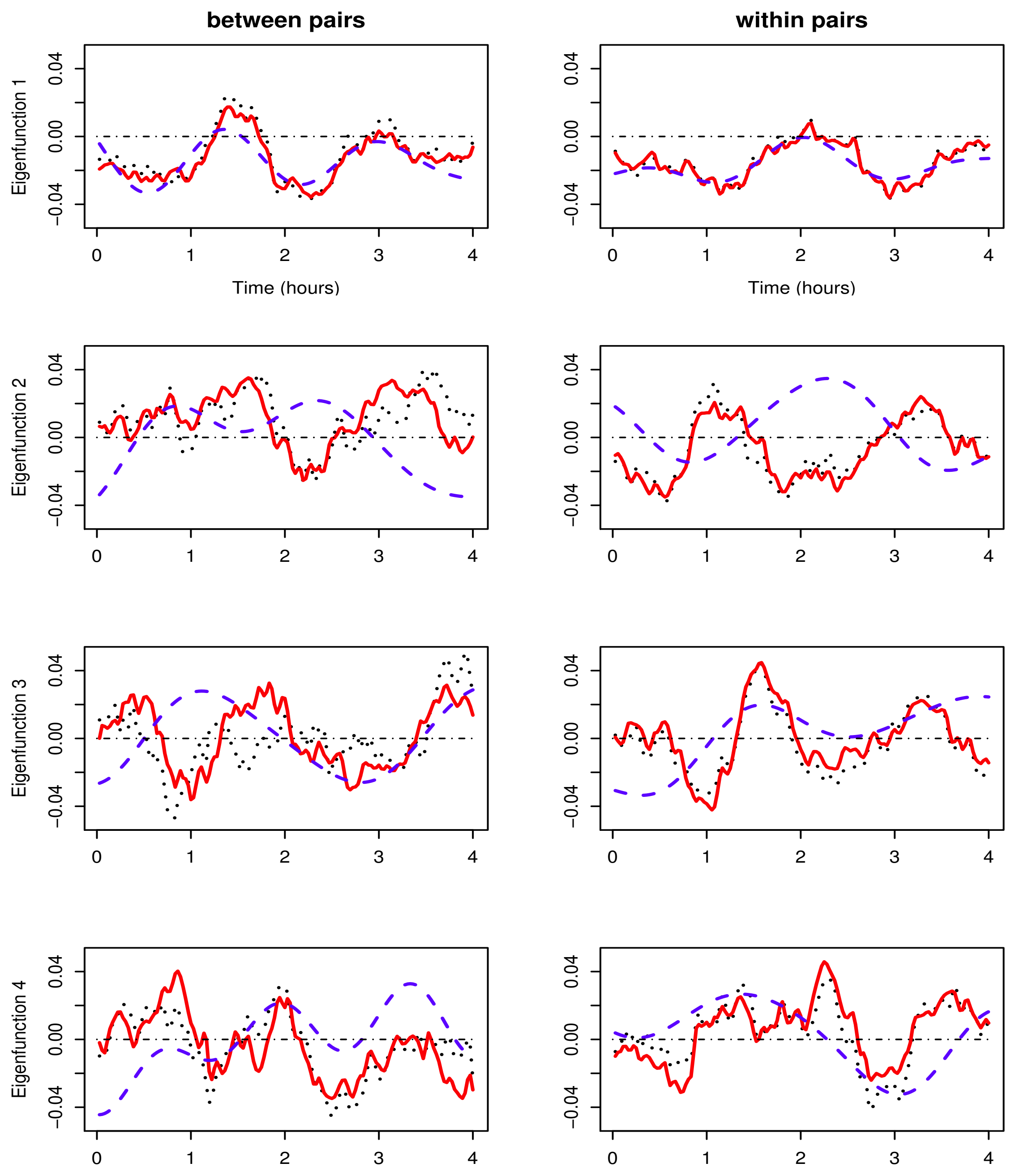

Figure 3: SHHS study. The eigenfunctions associated with the top four eigenvalues of $K_X$ and $K_U$. The left column is for $K_X$ and the right one is for $K_U$. The red sold lines are for FACE, the blue dashed lines are for thin plate splines, and the black dotted lines are for SVDS.

Table 4: SHHS study. Estimated eigenvalues of $K_X$ and $K_U$. All eigenvalues are multiplied by $J$ to refer to the variation in the data explained by the eigenfunctions. The row 'all' refers to the sum of all positive eigenvalues.

| | Eigenfunction | SVDS | FACE | Thin Plate Splines |
|---|---|---|---|---|
| | 1 | 4.31 | 3.64 | 1.91 |
| | 2 | 2.64 | 2.51 | 0.50 |
| $K_X$ | 3 | 1.88 | 1.46 | 0.31 |
| | 4 | 1.67 | 1.07 | 0.08 |
| | all | 48.14 | 15.81 | 2.81 |
| | 1 | 8.84 | 6.15 | 6.75 |
| | 2 | 5.69 | 3.31 | 2.55 |
| $K_U$ | 3 | 5.03 | 3.03 | 2.04 |
| | 4 | 4.49 | 2.51 | 1.49 |
| | all | 107.95 | 28.67 | 12.95 |