




Dynamic Multivariate Functional Data Modeling via Sparse Subspace Learning

Chen Zhang, Hao Yan, Seungho Lee & Jianjun Shi


To cite this article: Chen Zhang, Hao Yan, Seungho Lee & Jianjun Shi (2020): Dynamic Multivariate Functional Data Modeling via Sparse Subspace Learning, Technometrics, DOI: [10.1080/00401706.2020.1800516](https://doi.org/10.1080/00401706.2020.1800516)

To link to this article: <https://doi.org/10.1080/00401706.2020.1800516>

 View supplementary material [↗](#)


 Published online: 08 Sep 2020.

 Submit your article to this journal [↗](#)

 Article views: 365

 View related articles [↗](#)

 View Crossmark data [↗](#)

 Citing articles: 2 View citing articles [↗](#)



Dynamic Multivariate Functional Data Modeling via Sparse Subspace Learning

Chen Zhang^a, Hao Yan^b, Seungho Lee^c, and Jianjun Shi^d

^aDepartment of Industrial Engineering, Tsinghua University, Beijing, China; ^bSchool of Computing, Informatics, & Decision Systems Engineering, Arizona State University, Tempe, AZ; ^cSamsung Electronics, Suwon, South Korea; ^dH. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA

ABSTRACT

Multivariate functional data from a complex system are naturally high-dimensional and have a complex cross-correlation structure. The complexity of data structure can be observed as that (1) some functions are strongly correlated with similar features, while some others may have almost no cross-correlations with quite diverse features; and (2) the cross-correlation structure may also change over time due to the system evolution. With this regard, this article presents a dynamic subspace learning method for multivariate functional data modeling. In particular, we consider that different functions come from different subspaces, and only functions of the same subspace have cross-correlations with each other. The subspaces can be automatically formulated and learned by reformatting the problem as a sparse regression. By allowing but regularizing the regression change over time, we can describe the cross-correlation dynamics. The model can be efficiently estimated by the fast iterative shrinkage-thresholding algorithm, and the features of each subspace can be extracted using the smooth multi-channel functional principal component analysis. Some theoretical properties of the model are presented. Numerical studies, together with case studies, demonstrate the efficiency and applicability of the proposed methodology.

ARTICLE HISTORY

Received February 2019
Accepted June 2020

KEYWORDS

Dynamic correlation;
Functional data analysis;
Fused lasso; Sparse subspace
learning

1. Introduction

1.1. Background

Multivariate functional data, which arise from a collection of simultaneous recordings of several time courses for many subjects or units, are increasingly common and important in various applications. One concrete motivating example we introduce here is human gesture tracking (Gabel et al. 2016). In particular, the movements of a human subject are tracked by capturing real-time positions of 18 body joints using a Kinect pose estimation pipeline. Its data acquisition rate is 30 Hz with 2 cm accuracy in joint positions. Every joint is recorded as a point in a three-dimensional Cartesian coordinate system. Treating every coordinate of every joint as one function, we totally have 18×3 functions. The subject is instructed to conduct two gestures “bow up” and “throw” sequentially within total 248 frames captured. Figure 1 shows eight selected frames during these two gestures. The values of these 54 functions observed at these 248 time points are also shown in Figure A.4 in the supplementary materials. Clearly, some functions share very similar features (such as Functions 18, 21, and 24 in Figure 2(a)), indicating that they are strongly correlated. This is because these joints move in similar ways, such as the six joints on the two arms. In contrast, some other functions have quite diverse features with each other (such as Functions 33, 51, and 45 in Figure 2(b)), indicating that they are not correlated with each other. This is because these joints move in different ways, such as one joint on the arm and another joint on the

leg. With this regard, we may infer that these functions can be naturally clustered into different groups. Equivalently, we can say that these functions lie in different subspaces. Another thing to be noted is that their cross-correlation structure changes over time. For example, Functions 33 and 21 only share similar patterns in the first 110 time points. Then their cross-correlation disappears. On the contrary, Functions 51 and 21 have quite diverse patterns in the first 110 time points, but then they begin to move similarly in the subsequent 138 time points. This is because the first 110 time points are from the “bow up” gesture, while the last 138 points are from the “throw” gesture. Because for different gestures, the joints are required to cooperate and move in different ways, their cross-correlation structure would change.

In addition to the example shown above, multivariate functional data exist in many other applications. For example, in traffic monitoring, many traffic variables such as vehicle speed, flow rate, occupancy, etc., are continuously recorded for anomaly detection. In semiconductor manufacturing systems, hundreds of sensors are installed in a chamber to real-time monitor different process variables (e.g., temperature, pressure, electronic flows, etc.). In electroencephalography tests, multiple electrodes are placed at different places to record the brain activity over a certain time period for epilepsy signal detection. Therefore, there is a pressing need to model and analyze those multivariate functional data with consideration of their complex cross-correlation structure and dynamics.

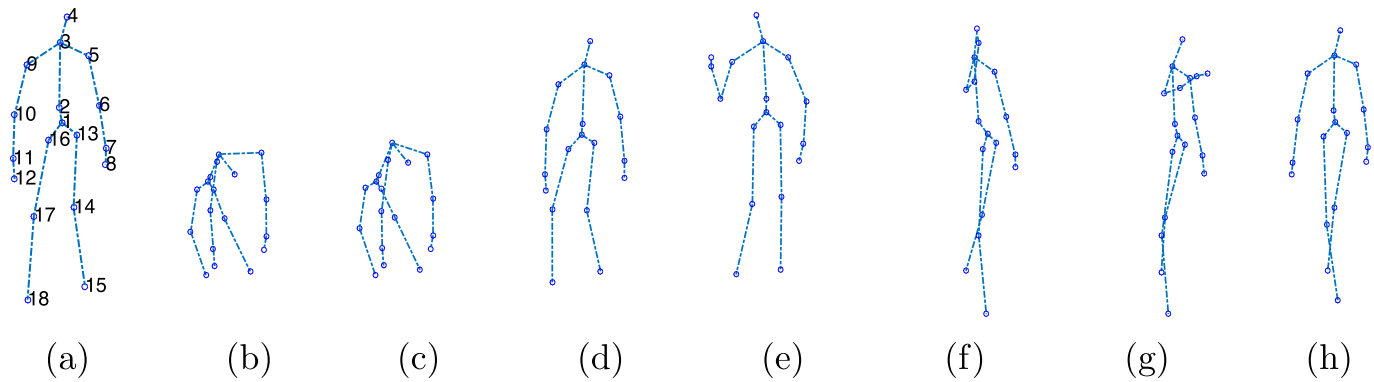


Figure 1. Snapshots of the joints at time point (a) t_{25} , (b) t_{50} , (c) t_{75} , (d) t_{100} , (e) t_{125} , (f) t_{150} , (g) t_{175} , (h) t_{200} . Specifically, the first four time points (a)–(d) belong to the first gesture “bow up,” and the second four time points (e)–(h) belong to the second gesture “throw.”

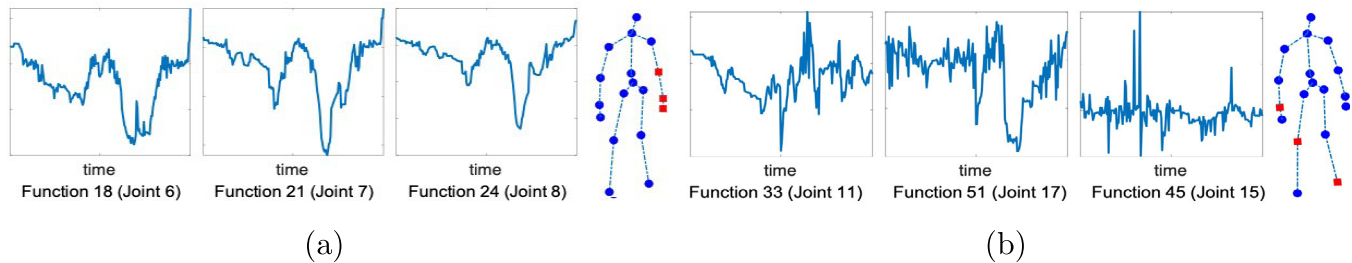


Figure 2. (a) Functions 18, 21, and 24, with the corresponding joints (denoted in red); (b) Functions 33, 51, and 45 with the corresponding joints (denoted in red).

1.2. Literature Review

We first review the current methodologies on the correlation analysis of multivariate functional data. In the literature, there are pioneer works focusing on bivariate functional data modeling, by developing measures to quantify their cross-correlations, such as the canonical correlation analysis (Leurgans, Moyeed, and Silverman 1993), the dynamical correlation analysis (Dubin and Müller 2005), and the functional singular value decomposition (Yang, Müller, and Stadtmüller 2011). As to multivariate functional data analysis (i.e., for more than two functions), for linear cross-correlations, Pan and Yao (2008) proposed a dynamic factor model to extract common factors from multiple functional data. Di et al. (2009), Chiou, Chen, and Yang (2014), and Paynabar, Zou, and Qiu (2016) introduced several multi-channel functional principal component analyses (MFPCA) methods to describe the within-function and between-function correlations. Chiou and Müller (2016) proposed a pairwise interaction model based on the cross-covariance surfaces between functions. For nonlinear cross-correlations, Chiou and Müller (2014) used a functional manifold model to regularize the functional features and characterize the cross-correlations. Besides dimension reduction techniques, other works include adapting the traditional parametric random-effects models (Fieuws and Verbeke 2006), and the nonparametric kernel smoothing approach for individual function modeling (Xiang, Qiu, and Pu 2013). However, one common limitation of all the aforementioned methods is that they assume multivariate functions are strongly correlated since each function is assumed to be a linear combination of all the extracted features. This assumption leads these methods to fail to recover correct functional features and fail to model

the multivariate functions accurately when they have diverse features with sparse cross-correlations, that is, come from different clusters.

To deal with the sparse cross-correlations, illustrated by sparse coding, Zhang et al. (2018b) proposed a sparse MFPCA which constrained the scores of functional PCA to be sparse, to restrict the unrelated basis functions to have zero weights. Consequently, different functions have nonzero weights on various bases, and can be weakly cross-correlated. Because it assumes the bases should be orthogonal with each other, this method may lose efficacy when the function space becomes complex, and the total number of basis functions increases. As a more intuitive way to deal with sparse cross-correlations, Zhang et al. (2018a) proposed to first cluster different functions by formulating the distance matrix directly using the correlations of different functions, and then used MFPCA of Paynabar, Zou, and Qiu (2016) for dictionary learning. However, since the correlation matrix is calculated by treating different time points as independent samples, the clustering result may be misleading, and consequently, the extracted principal components cannot guarantee accuracy. Furthermore, all the aforementioned methods cannot tackle functions with dynamic cross-correlations.

As a more powerful tool to describe conditional dependence structures between different random variables, recently graphical models are proposed for multivariate functional data analysis. In particular, Qiao, Guo, and James (2019) extended the graphical lasso (Yuan and Lin 2007) to multivariate functional data by constructing a penalized log-Gaussian likelihood on the functional PCA scores. Zhu, Strawn, and Dunson (2016) proposed a Bayesian framework by first implementing the functional PCA and then applying the Markov distributions and hyper Markov laws on the extracted PCA scores for graph

decomposition. Li and Solea (2018) proposed a nonparametric graphical model based on the additive conditional dependence.

Considering that functional data in most real applications are sampled over a grid of time points, one possible approach to handle dynamic cross-correlations is to construct separate graphs for every time point and regularize these graphs to be temporally consistent. Specifically, Kolar and Xing (2012) proposed to use a group lasso with l_2 penalty to regularize the change of graphs. Zhou, Lafferty, and Wasserman (2010) and Kolar and Xing (2011) proposed to construct the dynamic graph using a nonparametric kernel smoothing approach on the precision matrix. Qiao, Qian, and James (2020) further proposed a class of doubly functional graphical models by capturing the smooth evolution of the correlation matrix as another function. However, the methods usually give too much flexibility on the graph dynamics, by either causing the entire graph to restructure at specific time points or leading to more estimated change points than actual existence. Their model estimations for every time point with kernel smoothing also require extremely heavy computation, and hence hinder their application in cases with large sample size or high dimensions. Furthermore, because they typically assume that every variable (dimension) follows a Gaussian distribution (or Gaussian process), they have limited description power for general functional data. Last, dynamic graphical models mainly target at modeling and visualizing the dependence structure of different variables instead of modeling data for representation. Consequently, their performances for node representation are limited.

Motivated by the broad applications of multivariate or even high-dimensional functional data with sparse and dynamic cross-correlations, and the infancy of reasonable models to describe them, this article further explores this field. Considering that functional data often exist strong within-function correlations (or autocorrelations), all the functions actually lie in a low-dimensional subspace instead of being uniformly distributed across the ambient space. As such, recovering low-dimensional subspace in the functional data helps not only reduce the computational cost and memory requirements of algorithms, but also reduce the effect of the noise in the high-dimensional space. As to multivariate or high-dimensional functional data, they may lie in different low-dimensional subspaces.

As such, in this article, we propose a dynamic functional subspace-based framework for functional data modeling. Related to the *subspace structure*, our core idea is that different functions belong to different subspaces. Functions from the same subspace have strong cross-correlations with each other, while functions from different subspaces have weak or no cross-correlations. In this way, we can describe the complex cross-correlation structure of multivariate or high-dimensional functional data. Related to the *dynamic structure*, we assume that the subspace structure can change over time, and hence can capture the dynamics of the cross-correlations in real applications. In particular, we model the dynamic subspace structure by the so-called self-representation matrix. Based on it, we detect both change points of subspaces and the subspace structures simultaneously. We would like to further learn the structure of each subspace in each time segment, that is, the basis functions, by a smooth functional PCA. Finally, the function data in each

subspace can be represented by the corresponding bases. To our best knowledge, our method is the first that can model multivariate functions with both dynamic and sparse cross-correlation structures. In contrast, other existing methods of functional data modeling cannot be extended to this scenario trivially. Furthermore, our method is also the first to extend sparse subspace learning to dynamic cases for jointly spatiotemporal segmentation.

The remainder of the article is organized as follows. Section 2 introduces our proposed dynamic functional subspace model in detail. Section 3 discusses the model inference procedure. Section 4 talks about subspace segmentation and basis learning in detail. Section 5 uses some numerical studies to demonstrate the advantages of the proposed model by comparing it with some other state-of-the-art methods. Section 6 applies the developed methods into two real-data examples from human gesture tracking experiments and manufacturing systems. Finally, Section 7 concludes this article with remarks. Some additional figures, tables, numerical studies, and technical details are provided in the supplementary materials.

2. Dynamic Functional Subspace Learning

We first review the work on the static sparse subspace learning and provide a straightforward extension to model sparse cross-correlations of multivariate functions in Section 2.1. We then extend this framework to the dynamic correlation structure with the proposed dynamic functional subspace learning (DFSL) via the fused lasso penalty in Section 2.2. Finally, we study the theoretical properties of the proposed DFSL in Section 2.3.

2.1. Modeling Multivariate Functions via Static Functional Subspace Learning (SFSL)

Consider N p -dimensional (e.g., p channels) functional samples with the i th sample defined as $\mathbf{Y}_i(t) = [Y_{i1}(t), \dots, Y_{ip}(t)]$, $i = 1, \dots, N$, where t is on a compact interval $\mathcal{T} = [0, T]$, such that $\int_{\mathcal{T}} E[Y_{ij}(t)^2] dt < \infty$ for $j = 1, \dots, p$. In particular, we assume

$$Y_{ij}(t) = X_{ij}(t) + \epsilon_{ij}(t), \quad (1)$$

where $X_{ij}(t)$ is the signal function and $\epsilon_{ij}(t)$ ($j = 1, \dots, p$) are independent noise functions with mean $E[\epsilon_{ij}(t)] = 0$ and homoscedastic variance $\text{var}[\epsilon_{ij}(t)] = E[\epsilon_{ij}(t)^2] = \sigma_0^2$. We assume that the noise-to-signal ratio can be bounded as $\sigma^2 = \int_{\mathcal{T}} E[\epsilon_{ij}(t)^2] dt / \int_{\mathcal{T}} X_{ij}(t)^2 dt$. The noise can also have nonzero autocorrelation $\Gamma_j(t, s) = E[\epsilon_{ij}(t), \epsilon_{ij}(s)] / \sigma_0^2$. Furthermore, we assume that these p signal functions $\mathbf{X}_i(t) = [X_{i1}(t), \dots, X_{ip}(t)]$ can be partitioned into L different subspaces, \mathcal{S}_l ($l = 1, \dots, L$). The functions in the same subspace have strong cross-correlations, while the functions in different subspaces have no cross-correlations. Here, we will modify the basic assumptions of the traditional sparse subspace learning (Wang and Xu 2016) to model multiple multivariate functional samples as follows:

Assumption 1 ((A1) Subspace assumption). Each subspace \mathcal{S}_l is defined as the set of all functions that can be represented by the linear combination of d_l basis functions $\Phi_l =$

$\{\phi_{l1}(t), \dots, \phi_{ld_l}(t)\}$, that is,

$$\mathcal{S}_l \triangleq \left\{ X(t) | X(t) = \sum_{q=1}^{d_l} \alpha_q \phi_{lq}(t), \alpha_q \in \mathcal{R} \right\}. \quad (2)$$

In this article, we consider orthogonal basis functions, that is, $\int_{\mathcal{T}} \phi_{lq}(t) \phi_{lm}(t) dt = 0$, for all $q, m = 1, \dots, d_l, q \neq m$.

In particular, we assume that the subspace affiliation of $X_{ij}(t)$ is fixed for all samples, that is, the subspace membership of $X_{ij}(t)$ is the same for all $i = 1, \dots, N$. However, their basis coefficients $\alpha_{ij} = [\alpha_{ij1}, \dots, \alpha_{ijd_l}]$ can be different for different samples, that is, sampled at random from the unit sphere in $\mathcal{R}^{d_l \times 1}$ for different samples. In addition, denote that $\mathcal{X}_l = \{X_{ij}(t) | X_{ij}(t) \in \mathcal{S}_l, j = 1, \dots, p\}$ with the cardinality p_l . We have $\sum_{l=1}^L p_l = p$.

Assumption 2 ((A2) Self-expressive assumption). If there are sufficient functions from each subspace, which means (1) $p_l > d_l$ for $l = 1, \dots, L$, and (2) for these p_l functions, no d_l functions are spanned on the same $d_l - 1$ basis functions, then $X_{ij}(t)$ is *self-expressive*, which means for every $X_{ij}(t) \in \mathcal{X}_l$, we have

$$X_{ij}(t) = \sum_{X_{ir}(t) \in \mathcal{X}_l, r \neq j} b_{jr} X_{ir}(t). \quad (3)$$

This means that if $X_{ij}(t)$ is a function in \mathcal{X}_l , it can be represented as a linear combination of the other $p_l - 1$ functions in the same subspace. It should be noted that the above equation is a general formula, and it does not mean that all b_{jr} from the same subspace as $X_{ij}(t)$ should always have nonzero values.

With this assumption, $X_{ij}(t)$ can be recovered as a sparse solution of the multilinear regression equation $X_{ij}(t) = \mathbf{X}_i(t) \mathbf{b}_j$, with the regression coefficients $\mathbf{b}_j \in \mathcal{R}^{p \times 1}$ that have $b_{jr} \neq 0$ only for $\{r | X_{ir}(t) \in \mathcal{X}_l, r \neq j\}$, and $b_{jr} = 0$ otherwise. Notably, for a system with equations such as (3), \mathbf{b}_j may have infinite number of solutions. Similar to the sparse subspace learning, we can obtain the optimal solution by minimizing the objective function with the l_q -norm of the solution, that is,

$$\begin{aligned} & \min \|\mathbf{b}_j\|_q, \\ & \text{subject to } X_{ij}(t) = \mathbf{X}_i(t) \mathbf{b}_j, b_{jj} = 0. \end{aligned} \quad (4)$$

Different choices of q have different effects on the obtained solution. Typically, by decreasing the value of q from positive infinity toward zero, the sparsity of the solution increases. The extreme case of $q = 0$ corresponds to the general NP-hard problem of finding the sparsest representation of the given functions. Because we are interested in efficiently finding the nontrivial sparse representation of $X_{ij}(t)$ in the dictionary $\mathbf{X}_i(t)$, we consider minimizing the tightest convex relaxation of the l_0 -norm, that is, the l_1 -norm, which can be solved efficiently using convex programming tools. To handle noisy $Y_{ij}(t)$, a natural extension is to relax the equality constraint in (4) and solve the following unconstrained minimization problem instead (Elhamifar and Vidal 2013; Wang and Xu 2016) as

$$\begin{aligned} & \min_{\mathbf{b}_j} \lambda \|\mathbf{b}_j\|_1 + \frac{1}{2} \sum_{i=1}^N \|Z_{ij}\|_{\Gamma_j}^2, \\ & \text{subject to } Z_{ij}(t) = Y_{ij}(t) - \mathbf{Y}_i(t) \mathbf{b}_j, \quad b_{jj} = 0, \end{aligned} \quad (5)$$

for $j = 1, \dots, p$. Following Berrendero, Bueno-Larraz, and Cuevas (2020), here $\|\cdot\|_{\Gamma_j}^2$ is defined as

$$\|Z_{ij}\|_{\Gamma_j}^2 = \sum_{l=1}^{\infty} \frac{\int_{\mathcal{T}} Z_{ij}(t) e_l(t) dt}{\lambda_l}, \quad (6)$$

where $\{e_l, \lambda_l\}_{l=1, \dots, \infty}$ are the eigenfunctions and the corresponding eigenvalues of $\Gamma_j(t, s)$. When $\mathbf{Y}_i(t)$ are densely sampled at an equally spaced grid, according to Theorem 10 of Wang and Xu (2016), if λ is well tuned, with a very high probability, the solution of (5) will only have nonzero values for functions from the same subspace as $Y_{ij}(t)$. This indicates that solving (5) can recover the true subspaces. More details about this property can be found in the supplementary materials. Hereafter we denote the learning system with (5) as the *static functional subspace learning* (SFSL).

It should be noted that only when Z_{ij} is in the RKHS space of Γ_j , which is defined as $\mathcal{H}(\Gamma_j) = \{\int_{\mathcal{T}} |x(t)|^2 dt < \infty : x(t) = \sum_{l=1}^{\infty} a_l \sqrt{\lambda_l} e_l(t), \text{ for } \sum_{l=1}^{\infty} a_l^2 \leq \infty\}$, (6) has bounded value. Otherwise, when $Z_{ij} \notin \mathcal{H}(\Gamma_j)$, (6) goes to infinity, and the problems would be ill-defined and we need to follow the refined Mahalanobis functional distance in Berrendero, Bueno-Larraz, and Cuevas (2020) instead of (6) as

$$\|Z_{ij}\|_{a+\Gamma_j}^2 = \sum_{l=1}^{\infty} \frac{\lambda_l}{(\lambda_l + a)^2} \int_{\mathcal{T}} Z_{ij}(t) e_l(t) dt, \quad (7)$$

where a is a small positive value. Actually, (7) is the l_2 norm of $Z_{ij}(t)$ in the RKHS space expanded by $\Gamma_j + aI$, which is always invertible (Gohberg and Goldberg 2013, Theorem 8.1, p. 183).

With the obtained $\mathbf{b}_j, j = 1, \dots, p$ by the static model in (5), define the stacked estimated matrix $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_p] \in \mathcal{R}^{p \times p}$. It can be regarded as a representation of the cross-correlation structure of the p -dimensional functions. Actually, when $Y_{ij}(t), t \in [0, T]$, degenerates to a scalar, if $Y_{ij}(t)$ follows a Gaussian distribution, $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_p]$ is an approximate estimation of the sparse inverse covariance matrix (concentration matrix) in graphical lasso (Yuan and Lin 2007, Lemma 4). It is only an approximation solution since \mathbf{B} does not incorporate the symmetry and positive definiteness constraint, so an additional step is needed to transfer \mathbf{B} to the exact graphical lasso solution. Yet this approximation is proved to be very computationally attractive for estimating graphical lasso in sparse high-dimensional graphs (Meinshausen and Bühlmann 2006).

2.2. Dynamic Correlation Modeling via Fused Lasso

In this subsection, we consider that the cross-correlations between different functions can change over time t , which is very common in reality. For many systems (as the example in Section 1), their cross-correlation structure generally remains constant for a certain time period, and then changes to another constant state when the system undergoes some typically external disturbance. In another word, the cross-correlations only have stepwise changes at certain time points. Assume there are totally $S - 1$ change points inside \mathcal{T} with S time segments $\mathcal{T}^s (s = 1, \dots, S)$. For every time segment, we have

$$Y_{ij}^s(t) = X_{ij}^s(t) + \epsilon_{ij}(t), t \in \mathcal{T}^s, \quad (8)$$

where $X_{ij}^s(t)$ is a function in the subspace $\mathcal{S}_l^s (l = 1, \dots, L^s)$. For different segments $s = 1, \dots, S$, the subspace that the j th function belongs to can change (but is consistent for all the samples). The subspace number L^s and structures $\mathcal{S}_l^s (l = 1, \dots, L^s)$ can also be different, with the corresponding $\mathcal{X}_l^s = \{X_{ij}^s | X_{ij}^s \in \mathcal{S}_l^s, j = 1, \dots, p\}$ and cardinality p_l^s . The distribution of $\epsilon_{ij}(t)$ is consistent for all the time segments, without changes. In reality, the change points and changed subspace structures are usually unknown. To describe this dynamic system of (8) using a similar way as SFSL, $b_{jr}(j, r = 1, \dots, p)$ should be dynamic over time, as $\mathbf{b}_j(t) = [b_{j1}(t), \dots, b_{jp}(t)]'$ for $t \in \mathcal{T}$. Then $\mathbf{B}(t) = [\mathbf{b}_1(t), \dots, \mathbf{b}_p(t)]$, $t \in \mathcal{T}$ would eventually capture the dynamic cross-correlations. However, this naive relaxation gives too much flexibility on the change of \mathbf{b}_j , which could lead to the severe overfitting. With this in mind, to better regularize the dynamics, we consider penalizing the change of $\mathbf{b}_j(t)$, that is, $d\mathbf{b}_j(t)/dt$, to encourage its local constancy by borrowing the idea from the fused lasso (Tibshirani et al. 2005). In particular, we get $\mathbf{b}_j(t)$ as the solution of the following problem, that is,

$$\min_{\mathbf{b}_j(t)} \lambda_1 \int_{\mathcal{T}} \|d\mathbf{b}_j(t)/dt\|_1 dt + \lambda_2 \int_{\mathcal{T}} \|\mathbf{b}_j(t)\|_1 dt + \frac{1}{2} \sum_{i=1}^N \|Z_{ij}\|_{\Gamma_j}^2, \quad (9)$$

subject to $Z_{ij}(t) = Y_{ij}(t) - \mathbf{Y}_i(t)\mathbf{b}_j(t)$, $b_{jj}(t) = 0$, $t \in \mathcal{T}$,

for $j = 1, \dots, p$, separately. In (9), the second term encourages sparsity in the regression coefficients with the tuning parameter λ_2 . The first term encourages sparsity in their differences, that is, the flatness of the coefficient functions $\mathbf{b}_j(t)$, with the tuning parameter λ_1 .

As introduced earlier, in practice, $\mathbf{Y}_i(t)$ are densely recorded at a grid of discrete points. In this article, we assume that the grid points are dense and equally spaced at n points, that is, $t_1 < \dots < t_n$ and $t_j - t_{j-1} = t_{j+1} - t_j$ for all j , and the grid points are the same for all the samples. The time segment points are denoted as $t_{\tau_s} (s = 1, \dots, S - 1)$. Then we have $\mathbf{Y}_i, \mathbf{X}_i \in \mathcal{R}^{n \times p}$. We further denote the k th row of $\mathbf{Y}_i \in \mathcal{R}^{n \times p}$ as $\mathbf{Y}_i(t_k) \in \mathcal{R}^{1 \times p}$, with $\mathbf{Y}_i(t_k) = [Y_{i1}(t_k), \dots, Y_{ip}(t_k)]'$, and the j th column of \mathbf{Y}_i as $\mathbf{Y}_{ij} \in \mathcal{R}^{n \times 1}$ with $\mathbf{Y}_{ij} = [Y_{ij}(t_1), \dots, Y_{ij}(t_n)]$. Similarly, we set the k th row of $\mathbf{X}_i \in \mathcal{R}^{n \times p}$ as $\mathbf{X}_i(t_k)$ and the j th column of \mathbf{X}_i as $\mathbf{X}_{ij} \in \mathcal{R}^{n \times 1}$. In this article, we consider the normalized model with $\|\mathbf{X}_{ij}\|_2 = 1$. With the noise-to-signal assumption, we have $\epsilon_{ij} \in \mathcal{R}^{n \times 1}$ with $E[\epsilon_{ij}(t)] = 0$, $\text{var}[\epsilon_{ij}(t)] = \sigma^2/n$ and $E[\epsilon_{ij}\epsilon_{ij}']/\text{var}[\epsilon_{ij}(t)] = \Gamma_j$. Then (9) can be reformulated in terms of $\mathbf{b}_j(t_k)$ for $k = 1, \dots, n$ as

$$\begin{aligned} \min_{\mathbf{b}_j(t_k), k=1, \dots, n} \lambda_1 \sum_{k=2}^n \|\mathbf{b}_j(t_k) - \mathbf{b}_j(t_{k-1})\|_1 \\ + \lambda_2 \sum_{k=1}^n \|\mathbf{b}_j(t_k)\|_1 + \frac{1}{2} \sum_{i=1}^N \mathbf{Z}'_{ij} \Gamma_j^{-1} \mathbf{Z}_{ij}, \\ \text{subject to } Z_{ij}(t_k) = Y_{ij}(t_k) - \mathbf{Y}_i(t_k)\mathbf{b}_j(t_k), \\ b_{jj}(t_k) = 0, \text{ for } k = 1, \dots, n. \end{aligned} \quad (10)$$

Hereafter, we name the learning system with (10) as DFSL. Based on (10), we can estimate the change points for function

$Y_{ij}(t)$ as those $\{\hat{\tau}_s, s = 1, \dots, \hat{S} - 1 | b_{jr}(t_{\hat{\tau}_s}) - b_{jr}(t_{\hat{\tau}_{s-1}}) \neq 0, \text{ for any } r = 1, \dots, p\}$ where $\hat{S} - 1$ is the number of estimated change points (\hat{S} is the number of estimated time segments).

2.3. Theoretical Properties

Now we detail the assumptions of the proposed DFSL, under which its theoretical properties can be better established. In particular, it is assumed that there are $S - 1$ cross-correlation change points $1 < \tau_1 < \dots < \tau_{S-1} < n$ in the n -length functional data with $\tau_0 = 1$ and $\tau_S = n + 1$. For every time segment $\mathcal{T}^s = [\tau_{s-1}, \tau_s)$, the true constant cross-correlations can be represented by $[\boldsymbol{\beta}_{1,s}, \dots, \boldsymbol{\beta}_{p,s}]$ where $\boldsymbol{\beta}_{j,s} = \mathbf{b}_j(t_k), k \in \mathcal{T}^s$. Except for Assumptions (A1) and (A2), we list the additional assumptions here.

2.3.1. Other Assumptions

- (A3) The minimum coefficient change $\xi_{\min} = \min_{j=1, \dots, p} \min_{1 \leq s \leq S-1} \|\boldsymbol{\beta}_{j,s+1} - \boldsymbol{\beta}_{j,s}\|_2$ has a lower bound.
- (A4) The maximum coefficient change $\xi_{\max} = \max_{j=1, \dots, p} \max_{1 \leq s \leq S-1} \|\boldsymbol{\beta}_{j,s+1} - \boldsymbol{\beta}_{j,s}\|_2$ has an upper bound.
- This and (A3) are to constrain that the model is in the identifiable region, since too small a dynamic change cannot be identified (leading to a slower convergence rate of estimated change points), while too large a dynamic change may lead to unbounded estimation.
- (A5) The sequence $\{\delta_N^n\}_{N \geq 1, n \geq 1}$ is a nonincreasing and positive sequence tending to zero as either N or n tends to infinity, that is, $\delta_N^n \rightarrow 0$ as $n \rightarrow \infty$ and $\delta_N^n \rightarrow 0$ as $N \rightarrow \infty$. It further satisfies $Nn\delta_N^n(\xi_{\min})^2/\log(N) \rightarrow \infty$, as either $n \rightarrow \infty$ or $N \rightarrow \infty$. This is used to bound the convergence rate of the estimated change points.
- (A6) The minimum interval length $\Delta_{\min} = \min_{0 \leq s \leq S-1} |\tau_{s+1} - \tau_s|$ satisfies $\Delta_{\min} \geq n\delta_N^n$. This is to guarantee that each time segment should be long enough for correct recovery. Actually, this assumption is a restriction for n . When N is limited with a not very small δ_N^n , given $S - 1$ fixed change points on \mathcal{T} , only if n is not very small, we can guarantee this assumption is satisfied.
- (A7) The noise ϵ_{ij} follows the normal distribution with mean $\mathbf{0}$ and covariance matrix $\sigma^2 \Gamma_j/n$. This is to bound the noise-to-signal ratio. Too large the noise would lead to the false identification.
- (A8) The affinity between two subspaces \mathcal{S}_l and \mathcal{S}_r is defined as

$$\text{aff}(\mathcal{S}_l, \mathcal{S}_r) = \sqrt{\frac{\cos^2(\theta^{(1)}) + \dots + \cos^2(\theta^{(d_l \wedge d_r)})}{d_l \wedge d_r}}, \quad (11)$$

where $\theta^{(1)}, \dots, \theta^{(d_l \wedge d_r)}$ are principal angles between the two subspaces of dimension d_l and d_r , and are recursively defined by

$$\cos(\theta^{(k)}) = \max_{\phi_{li}(t) \in \mathcal{S}_l} \max_{\phi_{rj}(t) \in \mathcal{S}_r} \frac{\int_{\mathcal{T}} \phi_{li}(t)\phi_{rj}(t)dt}{\sqrt{\int_{\mathcal{T}} \phi_{li}^2(t)dt \int_{\mathcal{T}} \phi_{rj}^2(t)dt}}. \quad (12)$$

When $\phi_{li}(t) (i = 1, \dots, d_l)$ and $\phi_{rj}(t) (j = 1, \dots, d_r)$ are sampled at a discrete grid and denoted as Φ_l and Φ_r , the

affinity can be reformulated as $\|\Phi_l' \Phi_r\|_F$ where $\|\cdot\|_F$ is the Frobenius norm of the matrix. We assume

$$\max_{\substack{s=1,\dots,S \\ l,r=1,\dots,L^s}} \frac{\text{aff}(\mathcal{S}_l^s, \mathcal{S}_r^s)}{\sqrt{\max(d_l^s, d_r^s)}} \leq \kappa_0 / \log p, \quad (13)$$

where κ_0 is a constant. This assumption is to regularize the distance between different subspaces. If two subspaces are too close to each other and the noise further conceals their difference, it would be very hard to distinguish them from each other.

Theorem 1. Under Assumptions (A1)–(A7), the change points $\{\hat{\tau}_s, s = 1, \dots, \hat{S} - 1\}$ estimated by $\mathbf{b}_j(t_k)$ in (10) satisfy that, if $\hat{S} = S$, with probability tending to one:

$$P(\max_{1 \leq s \leq \hat{S}-1} |\hat{\tau}_s - \tau_s| \leq n\delta_N^n) \rightarrow 1, \text{ as } N \rightarrow \infty, \text{ or } n \rightarrow \infty, \quad (14)$$

when λ_1 and λ_2 satisfy $(Nn\delta_N^n \xi_{\min})^{-1} \lambda_1 \rightarrow 0$ and $(N\xi_{\min})^{-1} \lambda_2 \rightarrow 0$ as either $N \rightarrow \infty$ or $n \rightarrow \infty$.

The proof of [Theorem 1](#) is given in the supplementary materials. [Theorem 1](#) states that when the number of time segments is known or correctly estimated, the estimated time segments recover the true ones consistently as the sample size N increases. Furthermore, given a certain N , when we have more densely sampled functional data, that is, $n \rightarrow \infty$, we have $P(\max_{1 \leq s \leq \hat{S}-1} \frac{1}{n} |\hat{\tau}_s - \tau_s| \leq \delta_N^n) \rightarrow 1$. Then since $\delta_N^n \rightarrow 0$ as $n \rightarrow \infty$, this indicates that more accurate ‘‘relative’’ time segmentations, that is, smaller $\frac{1}{n} |\hat{\tau}_s - \tau_s|$, would be guaranteed for more densely sampled data.

Theorem 2. Based on [Theorem 1](#), assume Assumption (A8) also holds. Suppose $Y_{ij}^s \in \mathcal{X}_{i_s}^s$ for time segment \mathcal{T}^s . We have as either $N \rightarrow \infty$ or $n \rightarrow \infty$, the probability that the solution of (10) can have and only have nonzero values for functions from the same subspace as $Y_{ij}^s(t_k)$ in every time segment \mathcal{T}^s is at least $1 - c_1/p_l^{s^2} - c_2 \exp(-\Delta_{\min}\omega^2/8)$. Here ω is a positive constant satisfies

$$\sqrt{1+\omega} \leq \frac{1 + \sigma^2 - \sqrt{24} \log p_l^s (\max_{r,l=1,\dots,L^s} \frac{\text{aff}(\mathcal{S}_r^s, \mathcal{S}_l^s)}{\sqrt{\max(d_r^s, d_l^s)}}) - 2c_0\sigma}{\sqrt{24} \frac{\log p_l^s}{\sqrt{p_l^s}} \sigma + 2c_0\sigma^2}, \quad (15)$$

with $0 < c_0 < 1$. λ_1 and λ_2 should satisfy $\lambda_1/\lambda_2 \rightarrow 0$ as either $N \rightarrow \infty$ or $n \rightarrow \infty$, $\lambda_2/\sqrt{N \log N} \rightarrow \infty$ as $N \rightarrow \infty$, and $\lambda_2 n \delta_N^n / \sqrt{n} \rightarrow \infty$ as $n \rightarrow \infty$. c_1 and c_2 are two fixed positive constants.

The proof of [Theorem 2](#) is provided in the supplementary materials. There are multiple ways to set δ_N^n, λ_1 , and λ_2 . For example, if $\delta_N^n = \frac{(\log n)^{1.5} (\log N)^{1.5}}{nN}$, one possible choice of $\{\lambda_1, \lambda_2\}$ that meets the requirements of [Theorems 1](#) and [2](#) would be $\lambda_1 = \sqrt{\frac{(\log n)(\log N)}{(nN)}}$, and $\lambda_2 = N^{0.8} \sqrt{\frac{\log n}{n^{0.5}}}$.

Then built upon [Theorem 1](#), [Theorem 2](#) states that for each estimated time segment, the probability that the estimated \mathbf{b}_j^s only has nonzero values for functions from the same subspace as the output function \mathbf{Y}_{ij} is guaranteed to be bigger than a

threshold. Here ω is an intermediate variable. It depends on the affinity of different subspaces, that is, $\text{aff}(\mathcal{S}_r^s, \mathcal{S}_l^s)$, the cardinality p_l^s of the subspace l , and the noise-to-signal ratio σ^2 . Note that only if $1 + \sigma^2 - \sqrt{24} \log p_l^s (\max_{r,l=1,\dots,L^s} \frac{\text{aff}(\mathcal{S}_r^s, \mathcal{S}_l^s)}{\sqrt{\max(d_r^s, d_l^s)}}) - 2c_0\sigma - \sqrt{24} \frac{\log p_l^s}{\sqrt{p_l^s}} \sigma - 2c_0\sigma^2 > 0$, the right side of (15) can be guaranteed to be bigger than 1 and ω has feasible solution. Therefore, we need to bound $\text{aff}(\mathcal{S}_r^s, \mathcal{S}_l^s)$ and σ . Larger values of them lead to smaller (or even infeasible) ω , and hence a smaller (or even unguaranteed) probability of correct \mathbf{b}_j^s identification. As to the influence of p_l^s , a larger p_l^s leads to decrease of $c_1/p_l^{s^2}$, but its influence on ω is complicated and depends on $\text{aff}(\mathcal{S}_r^s, \mathcal{S}_l^s)$ and σ . Generally, as p_l^s increases, the decrease of the first part is faster than the change of the second part, so a larger p_l^s would still guarantee a larger probability of correct \mathbf{b}_j^s identification. Furthermore, according to Assumption (A6), as $n \rightarrow \infty$, we have $\Delta_{\min} \rightarrow \infty$, and the part $c_2 \exp(-\Delta_{\min}\omega^2/8) = 0$. This indicates the spatial subspace identification would also be better guaranteed. Last, note that [Theorem 2](#) shows the guarantee of correct \mathbf{b}_j^s identification for each function $\mathbf{Y}_j(t)$. From it, we can get the system-level guarantee by applying the union bound for all $\mathbf{Y}_j(t)$.

3. Model Inference

In this section, we will first introduce an efficient estimation method for the proposed DFSL model in [Section 3.1](#). Then we will talk about its tuning parameter selection in [Section 3.2](#).

3.1. Optimization via Fast Iterative Shrinkage-Thresholding Algorithm

Equation (10) is a convex problem including two parts: the smooth part for \mathbf{b}_j , that is,

$$f(\mathbf{b}_j) = \frac{1}{2} \sum_{i=1}^N \mathbf{z}_{ij}' \mathbf{\Gamma}_j^{-1} \mathbf{z}_{ij},$$

and the non-smooth part for \mathbf{b}_j , that is,

$$g(\mathbf{b}_j) = \lambda_2 \sum_{k=1}^n \|\mathbf{b}_j(t_k)\|_1 + \lambda_1 \sum_{k=2}^n \|\mathbf{b}_j(t_k) - \mathbf{b}_j(t_{k-1})\|_1.$$

Several categories of first-order methods have been developed to optimize this kind of composite function. Among them, the most popular one is in the class of iterative shrinkage-thresholding algorithm (ISTA). The pillar of ISTA-based methods is to construct the following model to approximate the composite objective function based on a searching point \mathbf{s}_j as

$$Q_{L_j}(\mathbf{b}_j, \mathbf{s}_j) := f(\mathbf{s}_j) + (\mathbf{b}_j - \mathbf{s}_j, \nabla f(\mathbf{s}_j)) + \frac{L_j}{2} \|\mathbf{b}_j - \mathbf{s}_j\|^2 + g(\mathbf{b}_j), \quad (16)$$

where L_j is a constant bigger than the Lipschitz constant of ∇f . With (16), we can develop the following gradient descent-like method for solving (10) by iteratively minimizing (16), that is,

$$\mathbf{b}_j^{k+1} = \arg \min_{\mathbf{b}_j} Q_{L_j}(\mathbf{b}_j, \mathbf{s}_j^k), \quad (17)$$

where k is the iteration step, and \mathbf{s}_j^k is the searching point for the current step.

Proposition 1. For the objective function of (10), (16) and (17) can be derived and decomposed for every $\mathbf{b}_{jr} = [b_{jr}(t_1), \dots, b_{jr}(t_n)]$ for $r = 1, \dots, p, r \neq j$ separately, as

$$\mathbf{b}_{jr}^{k+1} = \min_{\mathbf{b}_{jr}} \frac{1}{2} \|\mathbf{b}_{jr} - \mathbf{z}_{jr}\|_2^2 + s_{j2} \|\mathbf{b}_{jr}\|_1 + s_{j1} \|\mathbf{D}\mathbf{b}_{jr}\|_1, \quad (18)$$

where $\mathbf{z}_{jr} = \mathbf{s}_{jr}^k - \frac{1}{L_j} \sum_{i=1}^N \text{diag}(\tilde{\mathbf{Y}}_{ij}) (\tilde{\mathbf{Y}}_{ij} - \sum_{r \neq j} \text{diag}(\tilde{\mathbf{Y}}_{ir}) \mathbf{s}_{jr}^k)$ with $\tilde{\mathbf{Y}}_{ij} = \mathbf{\Gamma}_j^{-\frac{1}{2}} \mathbf{Y}_{ij}$ for $j = 1, \dots, p, i = 1, \dots, N$; $\mathbf{D} \in \mathcal{R}^{(n-1) \times n}$ is the first-order difference matrix that has $D_{kk} = -1, D_{k(k+1)} = 1$ for $k = 1, \dots, n-1$, and other components equal to 0.

Equation (18) is the exact form of the fused lasso signal appropriator (FLSA) function with $L_j = \|\mathbf{\Upsilon}_{-j}\|_2^2$ where $\mathbf{\Upsilon}_{-j} = [\mathbf{\Upsilon}'_{1(-j)}, \dots, \mathbf{\Upsilon}'_{N(-j)}]'$ and $\mathbf{\Upsilon}_{i(-j)} = [\text{diag}(\tilde{\mathbf{Y}}_{i1}), \dots, \text{diag}(\tilde{\mathbf{Y}}_{i(j-1)}), \text{diag}(\tilde{\mathbf{Y}}_{i(j+1)}), \dots, \text{diag}(\tilde{\mathbf{Y}}_{ip})]$, for $i = 1, \dots, N$. s_{j1} and s_{j2} are scaled λ_1 and λ_2 by L_j . Consequently, Proposition 1 indicates that the objective function of (10) could be decomposed and solved efficiently using the FLSA estimators. Here we adopt the method in Liu, Yuan, and Ye (2010).

In particular, the searching point \mathbf{s}_{jr}^k in every iteration is defined as

$$\mathbf{s}_{jr}^k = \mathbf{b}_{jr}^k + \rho_k (\mathbf{b}_{jr}^k - \mathbf{b}_{jr}^{k-1}),$$

where $\rho_k = (\sqrt{1 + 4\rho_{k-1}^2})/2$ is an iteratively chosen coefficient. This is the fast ISTA (FISTA) algorithm (Nemirovski 2005) which employs the approximation model not only based on the previous iteration, but rather on the previous two iterations. FISTA generally has the convergence rate of $O(1/k^2)$. The detailed algorithm of solving (10) based on FISTA is shown in Algorithm 1.

Data: $\mathbf{Y}_i, i = 1, \dots, N, \rho_0, \mathbf{\Gamma}_j, L_j, \mathbf{b}_{jr}^0, r = 1, \dots, p, r \neq j$

Result: Estimated $\mathbf{b}_{jr} = \mathbf{b}_{jr}^{k+1}, r = 1, \dots, p, r \neq j$

initialization

Initialize $\mathbf{b}_{jr}^1 = \mathbf{b}_{jr}^0$ for $r = 1, \dots, p, r \neq j$

Estimation

while $\sum_{r \neq j} \|\mathbf{b}_{jr}^k - \mathbf{b}_{jr}^{k-1}\|_2^2 > \epsilon_1$ **do**

 Set $\rho_k = (\sqrt{1 + 4\rho_{k-1}^2})/2$

 Set $\mathbf{s}_{jr}^k = \mathbf{b}_{jr}^{k-1} + \rho_k (\mathbf{b}_{jr}^k - \mathbf{b}_{jr}^{k-1})$ for $r = 1, \dots, p, r \neq j$

for $r = 1, \dots, j-1, j+1, \dots, p$ **do**

$\mathbf{z}_{jr} =$
 $\mathbf{s}_{jr}^k - \frac{1}{L_j} \sum_{i=1}^N \text{diag}(\tilde{\mathbf{Y}}_{ij}) [\tilde{\mathbf{Y}}_{ij} - \sum_{r \neq j} \text{diag}(\tilde{\mathbf{Y}}_{ir}) \mathbf{s}_{jr}^k]$

 Solve

$\mathbf{b}_{jr}^{k+1} = \min_{\mathbf{b}_{jr}} \frac{1}{2} \|\mathbf{b}_{jr} - \mathbf{z}_{jr}\|_2^2 + s_{j2} \|\mathbf{b}_{jr}\|_1 + s_{j1} \|\mathbf{D}\mathbf{b}_{jr}\|_1$
 using FLSA

end

 Set $k = k + 1$

end

Algorithm 1: Solving $\mathbf{b}_{jr} (r = 1, \dots, p, r \neq j)$ based on FISTA

It should be noted that $\{\mathbf{\Gamma}_j(t, s), \sigma\}$ may be unknown in practice. σ may be even relaxed to be different for $j = 1, \dots, p$. In this case, we need to estimate $\{\mathbf{\Gamma}_j, \sigma_j\}$ as well. Then (10) becomes no longer convex. As such, we may use the block coordinate descent (BCD) algorithm to estimate $\mathbf{b}_{jr} (r = 1, \dots, p, r \neq j)$ and $\mathbf{\Sigma}_j = \sigma_j^2 \mathbf{\Gamma}_j/n$ separately and iteratively, for $j = 1, \dots, p$. The detailed algorithm is shown in Algorithm A.1 in the supplementary materials.

3.2. Tuning Parameter Selection

In general, the selection of optimal tuning parameters for a given model can be a difficult task, which is further complicated as the number of tuning parameters increases. Here, we propose to follow the tuning procedure for the fused lasso in Nowak et al. (2011) to reduce the computation. Specifically, to simplify the search for the optimal tuning parameters, given the sample size N , we reparameterize λ_1 and λ_2 in terms of λ_0 and $\rho \in (0, 1)$ (here without confusion, we omit N in the later notation for conciseness), such that $\lambda_1 = \rho\lambda_0$ and $\lambda_2 = (1 - \rho)\lambda_0$. We can think of λ_0 as an overall tuning parameter with ρ determining how much emphasis is placed on sparsity versus smoothness. By fixing the possible values that ρ can take, we effectively reduce the search over λ_1 and λ_2 , to a search over one parameter λ_0 . In particular, we initially fix the possible values of ρ (e.g., $\{0.1, 0.3, 0.5, 0.7, 0.9\}$), the values would depend on n and N by using the results of Theorems 1 and 2). For each value of ρ , we find the value of λ_0 that results in each estimated variable to be 0, and denote this value by $\lambda_{0,\rho}^{\max}$. Then we chose a fixed number of candidate values for λ_0 from the interval $(0, \lambda_{0,\rho}^{\max})$. The optimal values of ρ and λ_0 are selected by searching over this two-dimensional grid for the value that minimizes the following criterion:

$$(Nn) \sum_{j=1}^p \log \left(\frac{\sum_{i=1}^N \mathbf{Z}'_{ij} \mathbf{\Gamma}_j^{-1} \mathbf{Z}_{ij}}{Nn} \right) + \log(Nn) \sum_{j=1}^p \sum_{r \neq j} k_{\rho, \lambda_0}(\mathbf{b}_{jr}). \quad (19)$$

Here, $k_{\rho, \lambda_0}(\mathbf{b}_{jr})$ is the number of nonzero elements in \mathbf{b}_{jr} given the current ρ and λ_0 . The term of $\sum_{j=1}^p k_{\rho, \lambda_0}(j)$ represents the complexity of the model, with larger values indicating greater complexity. This criterion is similar to the Bayesian information criterion. Its rational is that by minimizing (19), we attempt to find an appropriate model without overfitting the data. The first term will tend to be smaller for complex models, whereas the second term will tend to be smaller for simple models. For computational reasons, we prefer this approach for selecting the optimal tuning parameters.

4. Subspace Segmentation and Representation

Based on the estimated DFSL model, we first discuss how to identify the time segments and spatial segments in Section 4.1, and then discuss how to infer each subspace structure, that is, its corresponding basis functions and do function representation in Section 4.2.

4.1. Spatiotemporal Segmentation and Subspace Identification

In particular, we define

$$c_{jk} = \sum_{r \neq j} |\hat{b}_{jr}(t_k) - \hat{b}_{jr}(t_{k-1})|, \quad k = 2, \dots, n, j = 1, \dots, p, \quad (20)$$

where $\hat{b}_{jr}(t_k)$ is the solution of (10) solved by Algorithm 1 or Algorithm A.1 in the supplementary materials. According to Theorem 1, we know that under limited sample size N and unknown number of change points S , the estimated $\mathbf{b}_j(t)$ still has probability to change at nonchange points, which is false change point identification. As such, when c_{jk} of (20) is directly used to identify the potential change points for function $\mathbf{X}_j(t)$, the identified change points are probabilistic to be false. Therefore, we use c_{j0} to control the false alarm rate and define the change points as $T_j = \{k | c_{jk} > c_{j0}\}$. This is similar to setting control limit for control charts in statistical process control. In practice, for a real system, the setting of c_{j0} can be determined by the pre-specific false alarm rate and domain knowledge. It is not meant to be subjective, but is actually case-specific combined with system background. Similarly, for each time point t_k , we calculate $C_k = \sum_{j=1}^p \mathcal{I}(t_k \in T_j)$, and use $C_k(1 \leq k \leq n)$ for system-level change point decision-making.

After we have identified totally $S - 1$ change points $\hat{\tau}_s (s = 1, \dots, S - 1)$. For every time segment, we calculate the average DFSL coefficient $\bar{\mathbf{b}}_j^s \in \mathcal{R}^{p \times 1}$ where $\bar{b}_{jr}^s = \sum_{k=\hat{\tau}_{s-1}}^{\hat{\tau}_s-1} |\hat{b}_{jr}(t_k)|$ for $r \neq j$ and $\bar{b}_{jj}^s = 0$. Define $\bar{\mathbf{B}}^s = [\bar{\mathbf{b}}_1^s, \dots, \bar{\mathbf{b}}_p^s]$. Then similar to the procedure of Elhamifar and Vidal (2013) and Bahadori et al. (2015), we can define the symmetric affinity matrix $\mathbf{A}^s = \bar{\mathbf{B}}^s + \bar{\mathbf{B}}^{s'}$, and apply the spectral clustering (Ng, Jordan, and Weiss 2001) on \mathbf{A}^s for subspace clustering. We will not detail too much about the spectral clustering. One thing to be noted is that we allow the number of subspaces or the subspace basis functions to change for different time segments. When the number of subspaces is unknown in advance, it can be estimated as the number of eigenvalues of \mathbf{L}^s whose value are close to zero based on the spectral graph theory. Here the \mathbf{L}^s is the Laplacian matrix of the graph based on similarity matrix \mathbf{A}^s . Of course, other unsupervised clustering methods, such as hierarchical clustering and K-means can be used for subspace identification as well.

4.2. Basis Function Extraction and Data Representation

After segmenting functional data from spatiotemporal perspective, suppose for the time segment s , there are L^s subspaces identified, and the l th subspace \mathcal{S}_l^s has the cardinality p_l^s and the function set $\mathbf{Y}_{i(l)}^s = \{\mathbf{Y}_{i1}^s, \dots, \mathbf{Y}_{ip_l^s}^s\}$, where \mathbf{Y}_{ij}^s are the signals of \mathbf{Y}_{ij} in the s th time segment, that is, $Y_{ij}^s(t_k) (k = \tau_{s-1}, \dots, \tau_s - 1, j = 1, \dots, p_l^s)$. Then we can estimate the basis functions of \mathcal{S}_l^s using the MFPCA (Paynabar, Zou, and Qiu 2016). However, since here we assume that the basis functions are smooth, we optimize the loss function of MFPCA together with a smoothness regularization. In particular, our objective function is

$$\min_{\alpha_{iq}^s, \phi_{iq}^s, q=1, \dots, d_1^s} \sum_{i=1}^N \|\mathbf{Y}_{i(l)}^s - \sum_{q=1}^{d_1^s} \phi_{iq}^s \alpha_{iq}^{s'}\|_2^2 + \lambda_3 \sum_{q=1}^{d_1^s} \sum_{k=2}^{n_s} (\phi_{iq}^s(t_k) - \phi_{iq}^s(t_{k-1}))^2, \quad (21)$$

$$\text{subject to } \|\phi_{iq}^s\|_2 = 1, \phi_{iq}^{s'} \phi_{ir}^s = 0, \text{ for all } q, r = 1, \dots, d_1^s, q \neq r,$$

where $\phi_{iq}^s \in \mathcal{R}^{n_s \times 1}$ with $n_s = \tau_s - \tau_{s-1}$ is the q th basis function, and $\alpha_{iq}^s \in \mathcal{R}^{p_l^s \times 1}$ are the projections of $\mathbf{Y}_{i(l)}^s$ on ϕ_{iq}^s . We chose d_1^s such that the explained cumulative percentage of the sample variance by the first d_1^s MFPCA loadings is 95%. With this d_1^s , the penalty parameter λ_3 can be tuned by the cross-validation algorithm and (21) can be efficiently solved, following the similar procedure as Huang, Shen, and Buja (2008). After extracting the features, we can represent $\mathbf{Y}_i = [\mathbf{Y}_{i1}, \dots, \mathbf{Y}_{ip}]$ as a segment-wise linear decomposition, that is, $\mathbf{Y}_{i(l)}^s = \sum_{q=1}^{d_1^s} \phi_{iq}^s \alpha_{iq}^{s'}$ for $l = 1, \dots, L^s, s = 1, \dots, S$.

5. Numerical Studies

To evaluate the effectiveness of DFSL, we perform some numerical experiments using synthetic data generated from the assumed subspace model described in Section 2.2. We will first illustrate the description power of DFSL for multivariate functional data and the efficiency of the proposed estimation algorithm. Then we will compare DFSL with some state-of-the-art methods. Some sensitivity analyses of DFSL are also shown in the supplementary materials.

5.1. Synthetic Data Experiments

We assume every function sample has in total n equally spaced sampling time points. Among them there are $S - 1$ correlation change points $\tau_s (s = 1, \dots, S - 1)$ with a total of S time segments, that is, $n = \sum_{s=1}^S n_s$. For every time segment, we assume that there exist L^s subspaces $\mathcal{S}_l^s (l = 1, \dots, L^s)$. Then for every sample i , we can denote its functional data in the s th time segment as $\mathbf{Y}_i^s = [\mathbf{Y}_{i(1)}^s, \dots, \mathbf{Y}_{i(L^s)}^s]$ with the l th subspace data $\mathbf{Y}_{i(l)}^s \in \mathcal{R}^{n_s \times p_l^s}$. We generate $\mathbf{Y}_{i(l)}^s = \Phi_l^s \mathbf{A}_{i(l)}^s + \mathbf{E}_{i(l)}^s (l = 1, \dots, L^s)$, for $s = 1, \dots, S$. $\mathbf{E}_{i(l)}^s = [\epsilon_{i1}^s, \dots, \epsilon_{ip_l^s}^s]$ is the noise function, which is also time-segmented with the autocorrelation matrix Γ_j^s . $\Phi_j^s \in \mathcal{R}^{n_s \times d_1^s}$ are the basis functions for \mathcal{S}_j^s , and $\mathbf{A}_{i(l)}^s \in \mathcal{R}^{d_1^s \times p_l^s}$ is the weight matrix with the (q, j) component as the weight of the q th basis function ϕ_{iq}^s on the j th function. We further consider that $\mathbf{A}_{i(l)}^s$ can be decomposed into two parts, that is, $\mathbf{A}_{i(l)}^s = \mathbf{R}_{i(l)}^s \mathbf{V}_l^s$, where $\mathbf{V}_l^s \in \mathcal{R}^{m_l^s \times p_l^s}$ is the variation matrix, with every row orthogonal to each other indicating one variation pattern; $\mathbf{R}_{i(l)}^s \in \mathcal{R}^{d_1^s \times m_l^s}$ is the variation coefficient matrix, with every row $\mathbf{r}_{iq}^s (q = 1, \dots, d_1^s)$. Here in our simulation below, p_l^s, d_1^s, m_l^s, L^s and the subspace that $\mathbf{Y}_{ij} (j = 1, \dots, p)$ belongs to are temporarily assumed to be unchanged for different time segments. Hence, we omit the superscript s hereafter (however, in practice, these parameters may vary as the

case studies show in Section 6). Then we consider the following two models.

- Model (I): We consider $p = 8, n = 40, S = 2, \tau_1 = 21$, and $L = 2$. The first four functions come from the B-spline space with $d_1 = 3$, that is, $\phi_{1q}^s (q = 1, 2, 3)$ are the 1st, 4th, and 7th B-spline basis functions of order 3 with a grid of n_s equally spaced knots in $[0, 1]$ for $s = 1, 2$, respectively. The second four functions come from the Fourier space with $d_2 = 3$, that is, $\phi_{2q}^s(t_k) = \cos((q+1)\pi t_k - (q+1)\pi/2) (q = 1, 2, 3)$ with a grid of n_s equally spaced time points t_1, \dots, t_{n_s} in $[0, 1]$ for $s = 1, 2$, respectively. We assume $m_l = 2$ with \mathbf{r}_{iq}^s following a two-dimensional multivariate normal distribution with mean vector $\mathbf{0}$ and the same covariance matrix Σ whose $(\Sigma)_{uv} = 0.5^{|u-v|} (u, v = 1, 2)$, for $q = 1, 2, 3, l = 1, 2$, and $s = 1, 2$. For Γ_j^s , we set it the same for all $j = 1, \dots, p$ with $(\Gamma^s)_{uv} = 0.2^{|u-v|} (u, v = 1, \dots, n_s)$ for $s = 1, 2$.
- Model (II): We consider $p = 12, n = 128, S = 3, \tau_1 = 33, \tau_2 = 65$, and $L = 3$. The first four functions come from the B-spline space, and the second four functions come from the Fourier space, both of which has the same setting as Model (I). The last four functions come from the wavelet space, with $d_3 = 3$, that is, $\phi_{3q}^s (q = 1, 2, 3)$ are the q th

Vaidyanathan basis functions. \mathbf{r}_{iq}^s and Γ_j^s are generated in the same way as Model (I).

For each model, we generate $N = 500$ samples $\mathbf{Y}_i (i = 1, \dots, 500)$ from the corresponding model with the noise standard deviation $\sigma_0 = 0.05$, and use them to estimate \mathbf{b}_j and $\Gamma_j (j = 1, \dots, p)$ for DFSL. As shown in Figure 3, for every estimated $\hat{\mathbf{b}}_j$, only the functions in the same subspace as \mathbf{Y}_{ij} have nonzero regression coefficients (such as the first four functions from the B-spline subspace and the second four functions from the Fourier subspace). They keep constant over time and only have a step-wise change at t_{21} . The estimated $\hat{\Gamma}_j$ for Model (I) are jointly shown in Figure 4. Clearly, $\hat{\Gamma}_j$ approximates to the true $\Gamma_j = \text{diag}(\Gamma_j^1, \Gamma_j^2)$ with a standardized norm error $\|\hat{\Gamma}_j - \Gamma_j\|_2 / \|\Gamma_j\|_2 = 0.02$, demonstrating the efficiency of the proposed estimation method. Similar results also appear for Model (II). We further demonstrate the self-expression results of DFSL by evaluating its curve of $\mathbf{Y}_i(t)\mathbf{b}_j(t) (j = 1, \dots, p)$ for two selected functions in Figure 5. For comparison, we also plot the self-expression results of SFSL. Clearly, DFSL can self-express the multivariate functions very well, while unsurprisingly SFSL fails since it does not capture the dynamic cross-correlations. Based on the estimated model, we further use the smooth

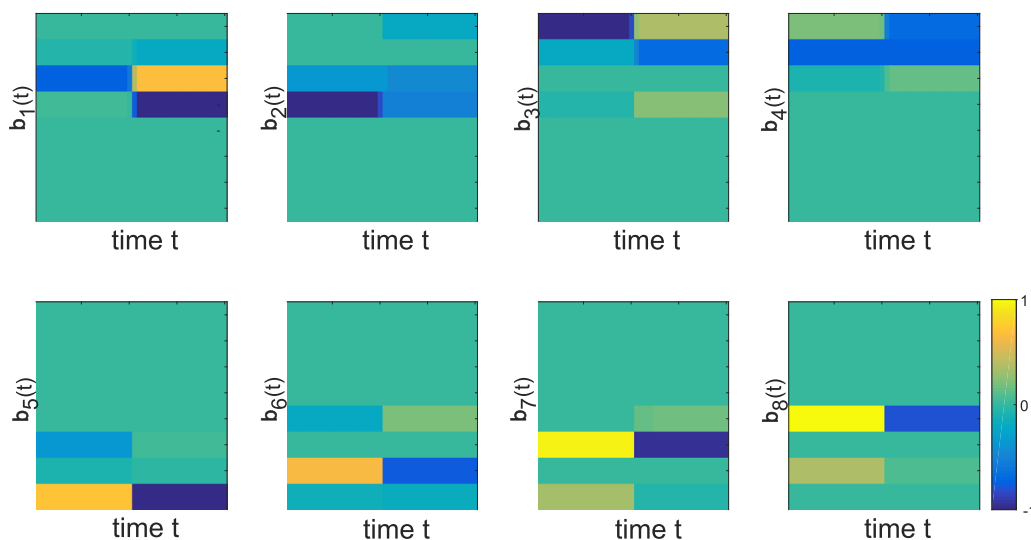


Figure 3. Estimated $\hat{\mathbf{b}}_j (j = 1, \dots, p)$ for Model (I).

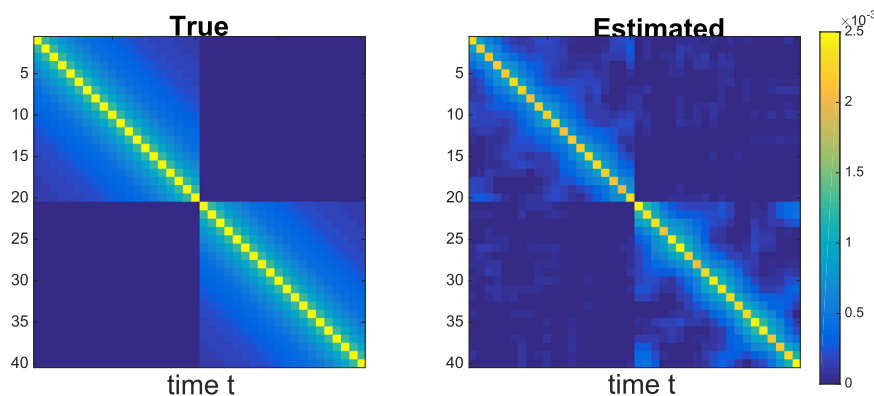


Figure 4. Estimated $\hat{\Gamma}_j$ of Model (I).

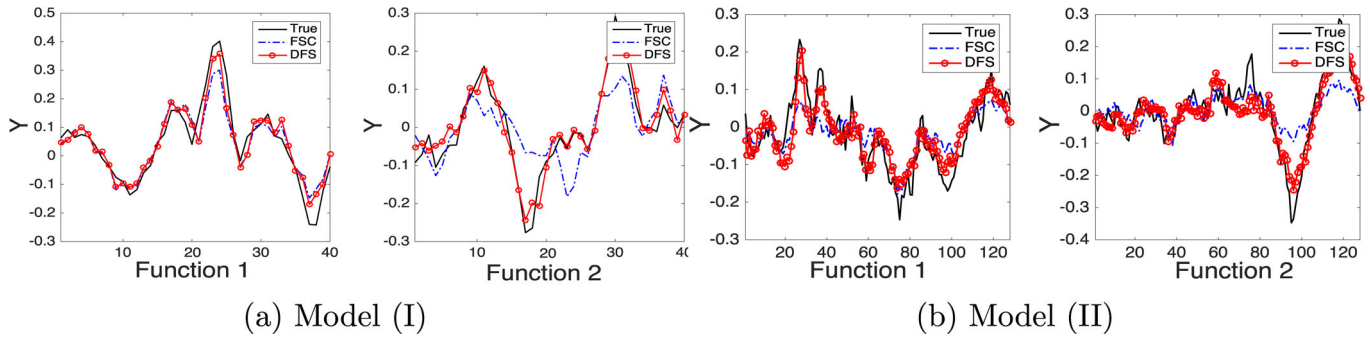


Figure 5. Self-expression results for two selected functions by DFSL and SFSL for Model (I) and Model (II).

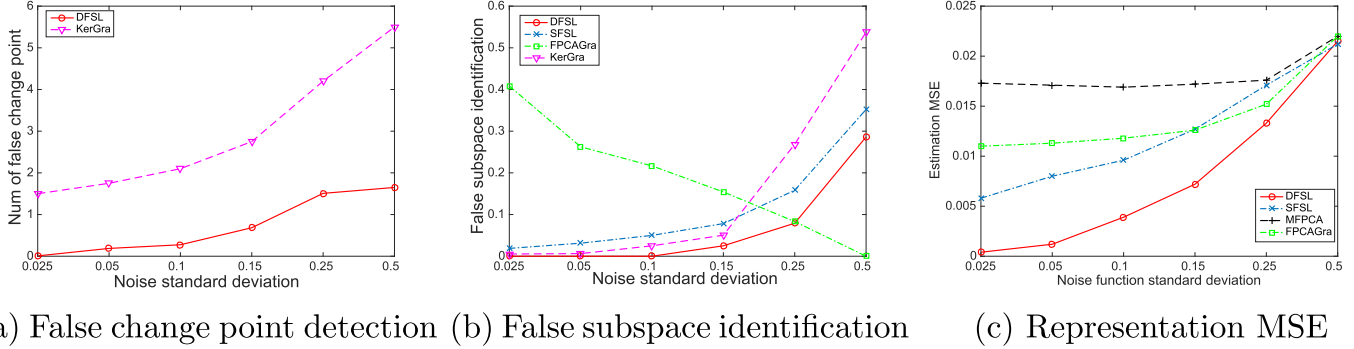


Figure 6. Modeling results for Model (I).

MFPCA for subspace inference, and compare the extracted basis functions $\hat{\Phi}_j^s$ with the true ones Φ_j^s . Then the extracted basis functions for the three subspaces in the time segment $t \in [t_{65}, t_{128}]$ of Model (II) are shown in Figure A.3 in the supplementary materials. They match the true basis functions very well.

5.2. Comparison With State-of-the-Art Methods

To better evaluate the performance of DFSL, besides SFSL, we have also made comparison of DFSL with some other state-of-the-art models introduced in Section 1, including the MFPCA of Paynabar, Zou, and Qiu (2016), the functional PCA-based sparse graph model of Qiao, Guo, and James (2019) (denoted as FPCAgra), and the kernel based graphical lasso of Kolar and Xing (2011) (denoted as KerGra).

First, we test the detection capability of cross-correlation change of DFSL and KerGra, which is the only baseline able to describe dynamic dependence structures. In particular, for DFSL, we set the threshold as $c_{j0} = 3 \times \text{std}(c_{jk})$ and identify the change points T_j of \mathbf{Y}_j . Then we calculate $C_k (k = 1, \dots, n)$ and identify the system-level change points as those t_k with $C_k \geq 1$.¹ For KerGra, we use the similar identification procedure by simply changing $b_{jr}(t_k)$ in (20) as the smoothed partial cross-correlation of \mathbf{Y}_j and \mathbf{Y}_r at time t_k . In this way, c_{j0} is expected to filter out some noisy dynamics caused by the kernel smoothing. For the identified change points of DFSL and KerGra, only the

change identifications occurred close to the true change points, that is, in the set $\{\tau_{s-1}, \tau_s, \tau_{s+1}\} (s = 1, \dots, S - 1)$, are accurate change point detections, while the other change point identifications are regarded as false detections. We run the experiment for 100 replications for different noise magnitudes for Model (I) and Model (II). The average false change point detection for Model (I) is reported in Figure 6(a). It shows that DFSL almost has no false change point detection for small noise magnitudes. As the noise magnitude increases, the false detection increases, but is still satisfactorily small. Furthermore, DFSL has much less false change point detection than KerGra, especially for cases with large noise magnitudes, indicating the robustness of DFSL. This can be better observed in Figure 7(a) for Model (II), where DFSL has no false change point detection for all noise magnitudes, while KerGra still performs very unsatisfactorily. As to miss change point detection, since DFSL has constant zero miss detection for all the simulation settings, we do not report the result here. However, KerGra does have nonzero miss detection, further demonstrating the superiority of DFSL.

We then compare DFSL with SFSL, FPCAgra, and KerGra in terms of the expected number of false subspace identification. (Though SFSL and FPCAgra cannot track dynamics of dependence structure since in our experiment the subspace membership of each function does not change over time, we can still test their expected number of false subspace identification using the subspace identification results based on the whole time duration.) In particular, for SFSL, for the estimated $\mathbf{b}_j (j = 1, \dots, p)$ in every replication, we test if any function not belonging to the subspace of \mathbf{Y}_{ij} has nonzero coefficients. If so, we conclude that \mathbf{Y}_{ij} is falsely identified. For DFSL, we use the average $\bar{\mathbf{b}}_j = \sum_{k=1}^n \mathbf{b}_j(t_k) / n$ for the test, following the same procedure as SFSL. As to the two graphical models KerGra

¹In the numerical studies, since the data are generated synthetically, we want to be most conservative to identify all the false alarms for model performance evaluation. As such, we set $\{k | C_k > 0\}$ as the system-level change points. However, in real-world data analysis, we recommend $\{k | C_k > 3 \times \text{std}(C_k)\}$.

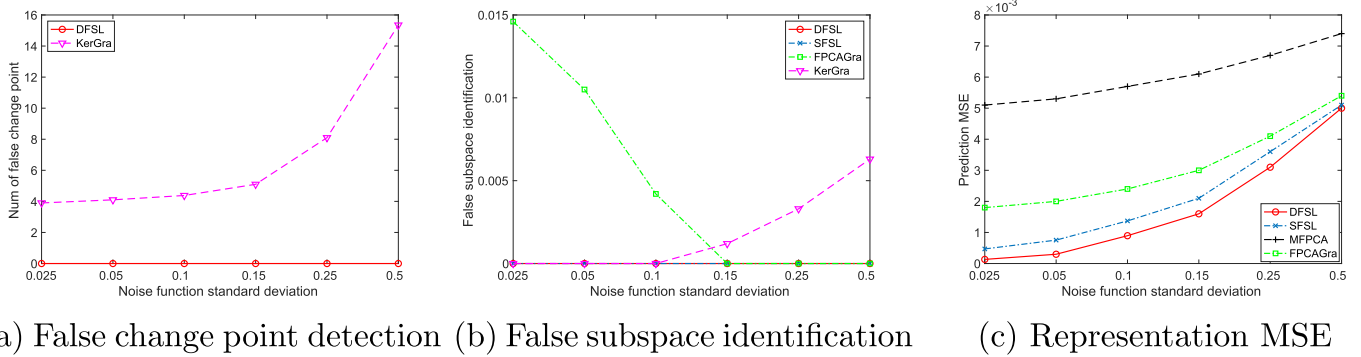


Figure 7. Modeling results for Model (II).

and FPCAGra, we test whether two functions from different subspaces have an edge. If so, considering that the graph is undirected, we treat both of these two functions as falsely identified. Then we calculate the average number of false subspace identification for different models using 100 simulation replications. As Figure 6(b) of Model (I) shows, DFSL has the smallest average number of false subspace identification for different noise magnitudes. Though KerGra follows DFSL closely for smaller noise magnitudes, as the noise increases, their difference becomes larger. It is noted that for Model (II), the advantage of DFSL over other methods is more significant since DFSL has no false identification all the way for different noise magnitudes. In addition, it is very interesting to see that in contrast to the other three models, the false identification of FPCAGra becomes larger as the noise magnitude decreases. Intuitively, as noise decreases, the correlation of different nodes should be better estimated. However, on the one hand, since FPCAGra treats the correlations of different functions static, as long as two functions have correlations in one segment, FPCAGra tends to treat them as correlated and connects an edge between each other, finally leading to over-connected graphs. On the other hand, FPCAGra prefers a sparser graph when the cross-correlation structure of different functions becomes weaker, which is precisely the case with increased independent noise.

After we segment the functional data from both spatial and temporal perspectives and identify each subspace's structure, we use SMFPCA of (21) for basis function estimation. We also implement SMFPCA on the identified subspaces for SFSL. Then we test the modeling power of DFSL and SFSL by calculating their representation mean square error (MSE) for additional 50 samples for each experiment replication. We further compare their performance with the other two functional PCA based algorithms, that is, MFPCA and FPCAGra. Their implementations follow their algorithms in the literatures, respectively. (It is notable that we do not involve KerGra in our comparison, since KerGra is not targeted at data representation.) The results based on 100 replications for different noise magnitudes are shown in Figure 6(c) for Model (I) and Figure 7(c) for Model (II). As expected, the representation MSEs of all the four models increase with the noise magnitude. When the noise standard deviation grows close to the magnitude of the signal, that is, $\sigma_0 = 0.5$, all the models lose their inference accuracy. Among these models, DFSL consistently has the smallest MSE, indicating its superiority. As to the other models, they lose the modeling power due to their no account of dynamic

cross-correlations. In particular, SFSL performs second best, since it can still identify correct subspaces. Its extracted basis functions combining all the time segments together would still be tolerable to capture the features of different time segments.

Furthermore, comparing the results of Model (I) and Model (II), it is notable that the performance of Model (II) is generally better than that of Model (I). This is because for Model (II), the number of sampling points of each time segment is bigger than that of Model (I). Consequently for each segment, we have more data to estimate the self-expression matrix \mathbf{B}^s , and can achieve more accurate spatiotemporal segmentations with almost zero false change point detection and false subspace identification. Consequently, the estimated basis functions for each subspace would be more precise, leading to smaller representation MSEs. This phenomenon is also consistent with Theorems 1 and 2. In addition, all the values shown in Figures 6 and 7 are average of results (i.e., representation MSE, false subspace identification, and false change point detection) from running the experiments for 100 replications. For better illustration and comparison, we also calculate their standard deviations in Tables A1–A4 in the supplementary materials. It shows that the estimated results have quite small standard deviations, demonstrating that the performance difference of these methods is statistically significant.

6. Case Studies

6.1. Motion Tracking

Now we reconsider the human gesture tracking experiments introduced in Section 1. In the experiment, this subject repeats the two gestures for nine times. Each of them is one sample $\mathbf{Y}_i(t_k) = [Y_{i1}(t_k), \dots, Y_{i54}(t_k)]$ ($k = 1, \dots, n_i$), where n_i is the number of frames (i.e., the time length) of sample i . Because for different samples n_i can be different, we first remove this non-synchronization among multiple samples using the dynamic time warping method (Keogh 2002) and make their time length equal to each other. Then we use eight of the nine samples for model training (including spatiotemporal subspace segmentation and basis function extraction), and use the last one for testing in terms of representation MSE).

Based on the training set, we can get the estimated $\hat{\mathbf{b}}_j$ ($j = 1, \dots, 54$) via DFSL. Figure A.5 in the supplementary materials shows the self-expressed curve of one training sample. Then we can calculate C_k ($1 \leq k \leq n$) and estimate the potential

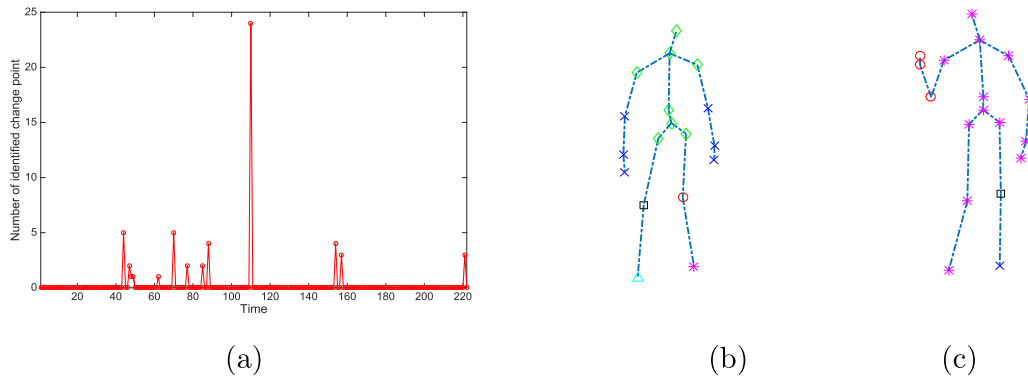


Figure 8. (a) Number of identified change points; (b) the joint clustering results for the motion 1 (first segment) at t_{25} ; (c) the joint clustering results for the motion 2 (second segment) at t_{125} . In particular, the joints of the same cluster are drawn in the same color and marker.

change points of the cross-correlations of the 54 functions, as shown in Figure 8(a). By defining the system potential change points as those $\{k | C_k > 3 \times \text{std}(C_k)\}$, $k = 111$, which is exactly the change point of these two gestures, is identified. Then we do subspace clustering for each of these two time segments separately. One thing to be noted is that if we cluster the 54 coordinates directly for this application, it is very possible to cluster the three coordinates of one joint into different clusters (subspaces), which is unreasonable to some degree. As such, here we do further operation on $\mathbf{B}^s = [\bar{\mathbf{b}}_1^s, \dots, \bar{\mathbf{b}}_p^s] \in \mathcal{R}^{54 \times 54}$ where $\bar{b}_{jr}^s = \sum_{l=\hat{t}_{s-1}}^{\hat{t}_s-1} |\hat{b}_{jr}(t_l)|$ for $s = 1, 2$. In particular, we integrate B_{jk}^s for coordinates of the same joint together and get $\bar{\mathbf{B}}^s \in \mathcal{R}^{18 \times 18}$ with $\bar{B}_{J_1 J_2}^s = \sum_{j \in J_1} \sum_{k \in J_2} \bar{b}_{jk}$ for $J_1, J_2 \in 1, \dots, 18$. Then we use $\bar{\mathbf{B}}^s$, which demonstrates the affinity matrix of different joints for spectral clustering, as discussed in Section 4. It is to be noted this integration method may not be optimal to combine results of coordinates of different joints together. Because we just use the case study to demonstrate the efficacy and efficiency of the proposed method on real datasets, further explorations on better integration method of coordinates for gesture analysis are not our emphasis in this article, and might be left as future work. Finally, we achieve six clusters for the first time segment and five clusters for the second one. The clustering results for the two segments are visualized in Figures 8(b) and (c), where the joints from different clusters are denoted by different marker styles and colors. In particular, for the first segment, the six joints on the two arms are clustered together, since every three joints on each arm are connected together and the two arms move in the same way in this “bow up” gesture. Similarly, the six joints on the trunk are clustered together. As to the four joints on the kicks and feet, each of them is identified as one single subspace. This is because, as shown in the video, the kicks have some “swing” movements, which yet do not appear in the movements of arms or trunks. As to the feet, they are not required to move as other joints, but only have some random fluctuation signals, that is, random noises (as shown in Figure A.6 in the supplementary materials). Therefore, the functions of the two joints on the feet (joint 15 and 18) cannot be predicted well by other functions, and their $\mathbf{b}_j(t)$ are almost zero for all the components. As such, the two joints are identified as two individual “falsely defined” functional subspaces. Strictly speaking, this situation does not satisfy Assumption 2, and should be treated as a special case. It demonstrates that even if there

are some noisy functions in the system, the proposed algorithm can still identify them separately and will not cluster them into other well-defined subspaces. As to the second segment, since this gesture requires the trunk, the right leg and the left arm to twist in the same way, these 12 joints are clustered together. The right arm is responsible for the “throw” action, so its joints are clustered together. As to the left leg, its two joints are identified as individual subspaces due to the same reason as the first segment.

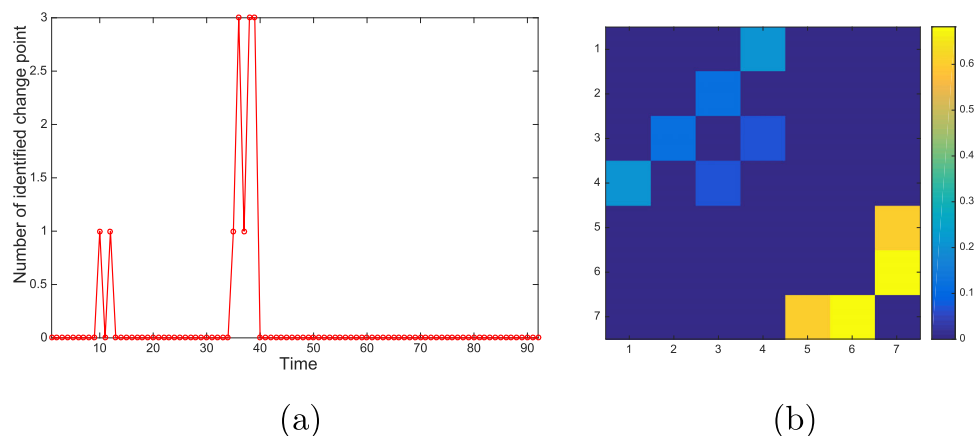
For better evaluating the performance of DFSL, we further compare it with the baselines in Section 5. Specifically, we do the experiment for in total nine times. For the i th replication, we select the i sample as the testing sample and the others as training samples. The average number of false change point detection and representation MSEs of the nine replications together with the standard deviations are shown in Table 1. (We do not compare the number of false subspace identification due to its unknown ground truth.) Similar to the performance in Section 5, DFSL has the best performance with the smallest representation MSE and the fewest number of false change point detection.

6.2. Manufacturing Process Monitoring

Now we consider another example from an advanced manufacturing system. In the system, seven sensors are used to monitor different process variables during the fabrication of every product sample. For simplification purpose, we denote these sensors as S1–S7. Figure A.8 in the supplementary materials illustrates their functional data for one product sample. It is clearly observed that some functions are quite similar with each other (such as S5–S7), demonstrating their strong correlations. Some other functions have quite diverse features (such as S1 and S5), indicating their weak cross-correlations. Furthermore, during the fabrication, there is an on-off operation at $t = 35$, which artificially changes the process variables, and consequently influences their correlation structure. In the dataset, we totally have $N = 46$ samples. Similar to the previous example, the profile length of different samples is different due to the fabrication inherent fluctuations. Therefore, we first remove the non-synchronization effect for different samples and set their time length as $n = 92$. We use 40 out of the 46 samples as the training set for subspace identification and basis function learning, and use the last six samples as the testing set for representation performance evaluation.

Table 1. Average false change point detection and average representation MSE of different models for case studies (with standard deviations in the parentheses).

Case study	False change point		Representation MSE ($\times 10^{-4}$)				
	DFSL	KerGra	DFSL	SFSL	MFPCA	FPCAra	KerGra
6.1	0.11(0.11)	6.08(0.35)	15.1(10.0)	37.3(4.21)	124(19.4)	30.5(6.34)	42.1(17.0)
6.2	0(0)	2(0)	0.26(0.07)	0.28(0.07)	0.66(0.17)	0.52(0.13)	21.5(7.86)

**Figure 9.** (a) The number of identified change points for every time point; (b) the affinity matrix for the manufacturing system data.

The estimated self-expression coefficients \mathbf{b}_j ($j = 1, \dots, 7$) are reported in Figure A.7, with the self-expressed curve for one sample shown in Figure A.8 in the supplementary materials. We can see that the coefficients are quite sparse, and most of them have a jump at around $k = 38$. Similar to Section 6.1, we calculate C_k for every time point. As shown in Figure 9(a), most of the identified change points are concentrated around $k = 38$. This delay is caused by the system operation delay itself, that is, most process variables do not respond to the on-off operation until $k = 38$. Then we use the spectral clustering algorithm for sensor clustering. The affinity matrix is shown in Figure 9(b), and the clustering result is quite consistent with engineering evaluations. The first four sensors belong to one subspace, and the other three belong to another subspace, demonstrating the efficiency of the proposed model once again.

Similar to Section 6.1, we compare DFSL with the other baselines. Here we run the experiments for a total of 7 times. For the i th replication, we select the $6i - 5$ to $6i$ samples as the testing set and the others as the training set for evaluation. The results shown in Table 1 are consistent with Sections 5 and 6.1. The representation MSEs of DFSL and SFSL are very similar. There are mainly two reasons for this small difference: (i) In the second segment $t \in [35, 92]$, the first four functions have 0 values all the time, so even if we use \mathbf{B}^1 (or any other self-regression matrix) for self-representation, the representation results will still be 0. (ii) As to the other three functions, their \mathbf{B}^2 and \mathbf{B}^1 are close to each other. Consequently, SFSL can achieve a satisfactory performance as DFSL.

7. Concluding Remarks

This article proposes a dynamic functional subspace learning method for multivariate functional data modeling. In particular, our model considers that different functions come from

different subspaces. Only functions from the same subspace have nonzero cross-correlations with each other, while functions from different subspaces have no cross-correlations at all. Furthermore, we allow the subspace structure to change over time but regularize its change flexibility. Consequently, we can describe the cross-correlation dynamics and also avoid over-parameterization. We also discuss the model inference in detail in terms of parameter estimation based on the fast iterative shrinkage-thresholding algorithm, parameter tuning, and subspace recovering based on the smooth multi-channel functional PCA. Finally, some numerical studies together with two real case studies demonstrate the efficiency and applicability of the proposed methodology.

Along this research direction, there are several potential valuable extensions. First, as shown in those two case studies, different functional samples may have different time lengths, or even irregular sampling time points. This is the common misalignment (deformation) problem. In these cases, the current proposed model cannot be applied directly. Furthermore, this misalignment may introduce additional noise and functional dissimilarity into the data. To eliminate these problems, some transformations with data alignment techniques need to be incorporated into the model. Second, in this article, we temporarily assume $d_l < p_l$ for every subspace. When this assumption is violated for a certain subspace, it becomes unidentifiable. Then how to tackle this scenario deserves more research. Finally, how to use the proposed model to construct a statistical monitoring scheme to detect outlier functional samples is another future work direction.

Supplementary Materials

The supplementary files contain additional simulations results, figures, tables, algorithms, proofs of theorems, as well as the MATLAB code of DFSL.

Acknowledgments

The authors thank the editor, associate editor, and two anonymous reviewers for many constructive comments and suggestions that have improved the quality of this work significantly.

Funding

This work was partially supported by NSFC 71901131, NSFC 71932006, NSF CCF 1740776, NSF DMS 1830363, NSF CMMI 1922739, and Tsinghua University Intelligent Logistics and Supply Chain Research Center grant THUCSL20182911756-001.

References

- Bahadori, M. T., Kale, D., Fan, Y., and Liu, Y. (2015), “Functional Subspace Clustering With Application to Time Series,” in *International Conference on Machine Learning*, pp. 228–237. [8]
- Berrendero, J. R., Bueno-Larraz, B., and Cuevas, A. (2020), “On Mahalanobis Distance in Functional Settings,” *Journal of Machine Learning Research*, 21, 1–33. [4]
- Chiou, J.-M., Chen, Y.-T., and Yang, Y.-F. (2014), “Multivariate Functional Principal Component Analysis: A Normalization Approach,” *Statistica Sinica*, 24, 1571–1596. [2]
- Chiou, J.-M., and Müller, H.-G. (2014), “Linear Manifold Modelling of Multivariate Functional Data,” *Journal of the Royal Statistical Society, Series B*, 76, 605–626. [2]
- (2016), “A Pairwise Interaction Model for Multivariate Functional and Longitudinal Data,” *Biometrika*, 103, 377. [2]
- Di, C.-Z., Crainiceanu, C. M., Caffo, B. S., and Punjabi, N. M. (2009), “Multilevel Functional Principal Component Analysis,” *The Annals of Applied Statistics*, 3, 458. [2]
- Dubin, J. A., and Müller, H.-G. (2005), “Dynamical Correlation for Multivariate Longitudinal Data,” *Journal of the American Statistical Association*, 100, 872–881. [2]
- Elhamifar, E., and Vidal, R. (2013), “Sparse Subspace Clustering: Algorithm, Theory, and Applications,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35, 2765–2781. [4,8]
- Fieuws, S., and Verbeke, G. (2006), “Pairwise Fitting of Mixed Models for the Joint Modeling of Multivariate Longitudinal Profiles,” *Biometrics*, 62, 424–431. [2]
- Gabel, M., Gilad-Bachrach, R., Renshaw, E., and Schuster, A. (2016), “Full Body Gait Analysis With Kinect,” in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, IEEE, pp. 1964–1967. [1]
- Gohberg, I., and Goldberg, S. (2013), *Basic Operator Theory*, Basel: Birkhäuser. [4]
- Huang, J. Z., Shen, H., and Buja, A. (2008), “Functional Principal Components Analysis via Penalized Rank One Approximation,” *Electronic Journal of Statistics*, 2, 678–695. [8]
- Keogh, E. (2002), “Exact Indexing of Dynamic Time Warping,” in *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB Endowment, pp. 406–417. [11]
- Kolar, M., and Xing, E. P. (2011), “On Time Varying Undirected Graphs,” in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 407–415. [3,10]
- (2012), “Estimating Networks With Jumps,” *Electronic Journal of Statistics*, 6, 2069. [3]
- Leurgans, S. E., Moyeed, R. A., and Silverman, B. W. (1993), “Canonical Correlation Analysis When the Data Are Curves,” *Journal of the Royal Statistical Society, Series B*, 55, 725–740. [2]
- Li, B., and Solea, E. (2018), “A Nonparametric Graphical Model for Functional Data With Application to Brain Networks Based on fMRI,” *Journal of the American Statistical Association*, 113, 1637–1655. [3]
- Liu, J., Yuan, L., and Ye, J. (2010), “An Efficient Algorithm for a Class of Fused Lasso Problems,” in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, pp. 323–332. [7]
- Meinshausen, N., and Bühlmann, P. (2006), “High-Dimensional Graphs and Variable Selection With the Lasso,” *The Annals of Statistics*, 34, 1436–1462. [4]
- Nemirovski, A. (2005), “Efficient Methods in Convex Programming,” Lecture Notes, Georgia Institution of Technology, <https://www2.isye.gatech.edu/~nemirovs/LecEMCO.pdf> [7]
- Ng, A. Y., Jordan, M. I., and Weiss, Y. (2001), “On Spectral Clustering: Analysis and an Algorithm,” in *NIPS* (Vol. 14), pp. 849–856. [8]
- Nowak, G., Hastie, T., Pollack, J. R., and Tibshirani, R. (2011), “A Fused Lasso Latent Feature Model for Analyzing Multi-Sample aCGH Data,” *Biostatistics*, 12, 776–791. [7]
- Pan, J., and Yao, Q. (2008), “Modelling Multiple Time Series via Common Factors,” *Biometrika*, 95, 365–379. [2]
- Paynabar, K., Zou, C., and Qiu, P. (2016), “A Change-Point Approach for Phase-I Analysis in Multivariate Profile Monitoring and Diagnosis,” *Technometrics*, 58, 191–204. [2,8,10]
- Qiao, X., Guo, S., and James, G. M. (2019), “Functional Graphical Models,” *Journal of the American Statistical Association*, 114, 211–222. [2,10]
- Qiao, X., Qian, C., and James, G. M. (2020), “Doubly Functional Graphical Models in High Dimensions,” *Biometrika*, 107, 415–431. [3]
- Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., and Knight, K. (2005), “Sparsity and Smoothness via the Fused Lasso,” *Journal of the Royal Statistical Society, Series B*, 67, 91–108. [5]
- Wang, Y.-X., and Xu, H. (2016), “Noisy Sparse Subspace Clustering,” *The Journal of Machine Learning Research*, 17, 320–360. [3,4]
- Xiang, D., Qiu, P., and Pu, X. (2013), “Nonparametric Regression Analysis of Multivariate Longitudinal Data,” *Statistica Sinica*, 23, 769–789. [2]
- Yang, W., Müller, H.-G., and Stadtmüller, U. (2011), “Functional Singular Component Analysis,” *Journal of the Royal Statistical Society, Series B*, 73, 303–324. [2]
- Yuan, M., and Lin, Y. (2007), “Model Selection and Estimation in the Gaussian Graphical Model,” *Biometrika*, 94, 19–35. [2,4]
- Zhang, C., Yan, H., Lee, S., and Shi, J. (2018a), “Multiple Profiles Sensor-Based Monitoring and Anomaly Detection,” *Journal of Quality Technology*, 50, 344–362. [2]
- (2018b), “Weakly Correlated Profile Monitoring Based on Sparse Multi-Channel Functional Principal Component Analysis,” *IIEE Transactions*, 50, 878–891. [2]
- Zhou, S., Lafferty, J., and Wasserman, L. (2010), “Time Varying Undirected Graphs,” *Machine Learning*, 80, 295–319. [3]
- Zhu, H., Strawn, N., and Dunson, D. B. (2016), “Bayesian Graphical Models for Multivariate Functional Data,” *Journal of Machine Learning Research*, 17, 1–27. [2]