Regularized Multivariate Two-way Functional Principal Component Analysis

Mobina Pourmoshir
mobina.pourmoshir@marquette.edu

Department of Mathematical and Statistical Sciences
Marquette University

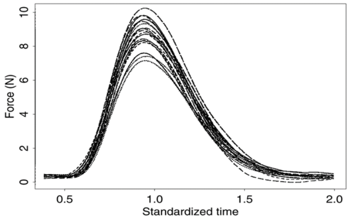
MARQUETTE UNIVERSITY
BE THE DIFFERENCE.

Outline

1. Introduction & Background
2. A SVD Approach for Regularized Multivariate FPCA
3. Two-way Regularized Multivariate FPCA
4. Conclusion & Future Work

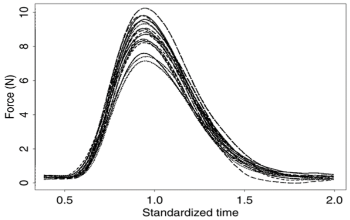
Introduction

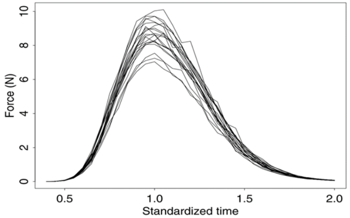
- **Functional data** are observations that change continuously over a domain (like time, space, or wavelength) and are often visualized as curves, trajectories, or functions rather than isolated points.



Introduction

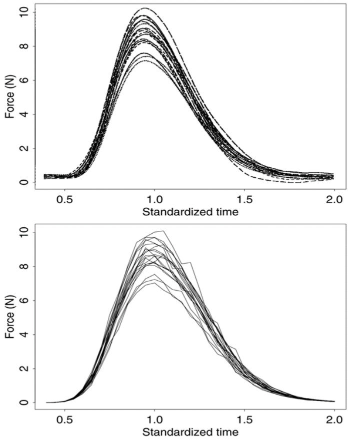
- **Functional data** are observations that change continuously over a domain (like time, space, or wavelength) and are often visualized as curves, trajectories, or functions rather than isolated points.
- In practice, these data are often recorded at discrete time points or grid locations, even though they originate from continuous processes in areas like engineering, finance, environmental science, and healthcare.





Introduction

- **Functional data** are observations that change continuously over a domain (like time, space, or wavelength) and are often visualized as curves, trajectories, or functions rather than isolated points.
- In practice, these data are often recorded at discrete time points or grid locations, even though they originate from continuous processes in areas like engineering, finance, environmental science, and healthcare.
- **Functional Data Analysis (FDA)** is a statistical framework that treats these observations as realizations of smooth underlying functions, allowing for more accurate modeling and interpretation of continuous processes.

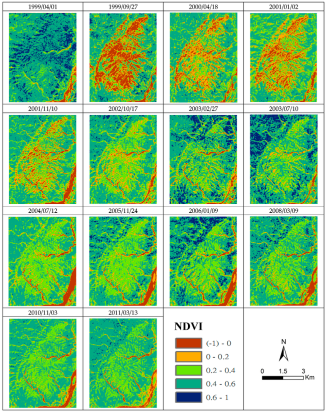


Functional Principal Component Analysis

- **Functional PCA (FPCA):** An extension of classical PCA for dimension reduction and uncovering hidden patterns in functional data; it identifies orthogonal functions that capture the main sources of variation, preserving the most important information. [Ramsay and Silverman, 2005].
- **Extensions of FPCA:**
  - Smoothed FPCA: Adds roughness penalties for smoothness [Silverman, 1996, Huang et al., 2008].
  - Sparse FPCA: Enforces sparsity for interpretability [Shen and Huang, 2008, Nie and Cao, 2020].
  - Multivariate FPCA (MFPCA): Extends FPCA to multivariate functions [Silverman, 1996, Happ and Greven, 2018].
  - Regularized MFPCA: Penalties improve estimation & interpretability [Haghbin et al., 2025].
- **Impact:** More adaptable, robust, and applicable across diverse scientific and business problems.

Two-way Functional Principal Component Analysis

- **Two-way functional data:** Observations vary along two domains (e.g., time × space, time × frequency), with applications in climate, neuroscience, finance, public health, and marketing.



Two-way Functional Principal Component Analysis

- **Two-way functional data:** Observations vary along two domains (e.g., time $\times$ space, time $\times$ frequency), with applications in climate, neuroscience, finance, public health, and marketing.
- **Extension of FPCA:** Huang [Jianhua Z. Huang and Buja, 2009] applied regularization to both left and right singular vectors in SVD.
- **Practical challenges:**
  - Data observed on discrete grids (minutes, hours, days).
  - Issues: measurement noise, irregular sampling, missing data, loss of smoothness.
- **Proposed framework:**
  - Unified FPCA for two-way multivariate functional data.
  - Smoothness penalties preserve functional structure.
  - Sparsity penalties enhance interpretability.
  - Effective for dimension reduction in complex datasets.

Foundations of FPCA through Minimizing Reconstruction Error

- **Goal:** Identify functional directions that maximize variance (low-rank approximation of functional data). For functional data $X \in \mathbb{R}^{n \times m}$ contains the discretized functional observations (rows correspond to subjects, columns to grid points), $v \in \mathbb{R}^m$ represents the estimated principal component (function), and $u \in \mathbb{R}^n$ denotes the associated principal component scores.
- **Reconstruction problem:**

$$\min_{u,v} \|X - uv^\top\|_F^2 = \mathrm{tr}\{(X - uv^\top)(X - uv^\top)^\top\},$$

- **Optimization steps:**

$$\text{Fix } v: \quad u = \frac{Xv}{v^\top v} \qquad \text{and} \qquad \text{Fix } u: \quad v = \frac{X^\top u}{u^\top u}$$

Extensions of FPCA via Regularization

- **Goal:** Balance **variance explanation**, **smoothness**, and **interpretability**.
- Reformulate FPCA as a **penalized low-rank approximation** problem:

$$\min_{u,v} \|X - uv^\top\|_F^2 + \mathcal{P}(u, v)$$

- Two directions:
  - **Smooth FPCA:** adds roughness penalty on functions.
  - **Sparse FPCA:** adds sparsity penalty on loadings.
- **Algorithms:** Based on iterative **power method** and **thresholding** updates.

Smooth Functional PCA

- Problem setup [Huang et al., 2008]:

$$\min_{u,v} \|X - uv^\top\|_F^2 + \alpha\, u^\top u\, v^\top \Omega v$$

  - $X \in \mathbb{R}^{n \times p}$: discretized functional data.
  - $u \in \mathbb{R}^n$: scores.
  - $v \in \mathbb{R}^p$: loading function.
  - Ω: roughness penalty matrix (e.g., integrated squared 2nd derivative).
  - α: tuning parameter

- A power algorithm is defined to compute the PCs while incorporating smoothness penalty.

Smooth Functional PCA

- Problem setup [Huang et al., 2008]:

$$\min_{u,v} \|X - uv^\top\|_F^2 + \alpha\, u^\top u\, v^\top \Omega v$$

  - $X \in \mathbb{R}^{n \times p}$: discretized functional data.
  - $u \in \mathbb{R}^n$: scores.
  - $v \in \mathbb{R}^p$: loading function.
  - Ω: roughness penalty matrix (e.g., integrated squared 2nd derivative).
  - α: tuning parameter

- A power algorithm is defined to compute the PCs while incorporating smoothness penalty.
- Consider the SVD of X as $X = UDV^\top$, where U and V have orthonormal columns and D is diagonal with ordered singular values. In particular, for $X = udv^\top$, v is the first principal component and $u = ud$ gives the associated scores. With these representations, the power algorithm (described below) converges quickly, typically in only a few iterations.

Power Algorithm

Algorithm: Penalized Power Iteration

1. Initialize v.
2. Repeat until convergence:
    1. $u \leftarrow Xv$
    2. $v \leftarrow (I + \alpha\Omega)^{-1} X^\top u$
    3. $v \leftarrow \dfrac{v}{\|v\|}$
3. Update $X \leftarrow X - \sigma\, uv^\top$ and proceed to next component.

- For notational convenience, we define $S(\alpha) = (I + \alpha\Omega)^{-1} \in \mathbb{R}^{m \times m}$, which simplifies expressions involving regularization. The penalty matrix Ω is set up so that larger values of the quadratic form $v^\top \Omega v$ mean rougher functions. This means that it penalizes functions that change quickly between time points.

Tunning Smoothness Parameters

- To select the optimal tuning parameters α efficiently, one can use a traditional Cross-Validation (CV) criterion and a computationally efficient closed-form Generalized Cross-Validation (GCV) criterion:

$$\mathrm{CV}(\alpha) = \frac{1}{m} \sum_{j=1}^{m} \frac{\left[\{(I - S(\alpha))(X^T u)\}_{jj}\right]^2}{(1 - \{S(\alpha)\}_{jj})^2},$$

where $\{\cdot\}_{jj}$ denotes the j-th diagonal element.

$$\mathrm{GCV}(\alpha) = \frac{1}{m} \frac{\|(I - S(\alpha))(X^T u)\|^2}{\left(1 - \frac{1}{m}\mathrm{tr}\{S(\alpha)\}\right)^2}.$$

Sparse Functional PCA

- *Standard FPCA loadings are dense, involving linear combinations of all grid points $\rightarrow$ hard to interpret.*
  Sparsity highlights only the most relevant features, thereby enhancing interpretability.

- *Dense loadings capture noise $\rightarrow$ unstable components.*
  Sparsity filters out uninformative variation, yielding more robust principal components, reducing dimensionality, and facilitating interpretation.

- *All grid points contribute equally $\rightarrow$ no feature selection.*
  Sparsity acts as an inherent feature selector, directing attention to key time points, with most entries reduced to zero while only a few contribute meaningfully to the structure.

- Sparse FPCA formulation [Shen and Huang, 2008]:

$$\min_{u,v} \|X - uv^\top\|_F^2 + p_\gamma(v) \tag{1}$$

where $p_\gamma(v)$ is a sparsity-inducing penalty.

Sparsity penalties

- **Soft-thresholding (Lasso):**

$$p_\gamma^{\text{soft}}(|\theta|) = 2\gamma|\theta|, \xrightarrow[\text{minimizer}]{} h_\gamma^{\text{soft}}(y) = \text{sign}(y)(|y| - \gamma)_+$$

- **Hard thresholding:**

$$p_\gamma^{\text{hard}}(|\theta|) = \gamma^2 I(|\theta| \neq 0), \xrightarrow[\text{minimizer}]{} h_\gamma^{\text{hard}}(y) = I(|y| > \gamma)\, y$$

- **SCAD penalty:**

$$p_\gamma^{\text{SCAD}}(|\theta|) = \begin{cases} 2\gamma|\theta|, & |\theta| \leq \gamma, \\ \dfrac{\theta^2 - 2a\gamma|\theta| + \gamma^2}{a-1}, & \gamma < |\theta| \leq a\gamma, \\ \dfrac{(a+1)\gamma^2}{2}, & |\theta| > a\gamma, \end{cases} \xrightarrow[\text{minimizer}]{} h_\gamma^{\text{SCAD}}(y) = \begin{cases} \text{sign}(y)(|y| - \gamma)_+, & |y| \leq 2\gamma, \\ \dfrac{(a-1)y - \text{sign}(y)a\gamma}{a-2}, & 2\gamma < |y| \leq a\gamma, \\ y, & |y| > a\gamma, \end{cases}$$

where $a = 3.7$ (Fan and Li [2001]).

sFPCA-rSVD Algorithm

To implement the sPCA-rSVD algorithm discussed above, we use the following iterative procedure to minimize the objective function defined in Equation (1).

Algorithm: sFPCA-rSVD

1. Initialization: Compute the best rank-one approximation of X using singular value decomposition (SVD), where $X \approx suv^\top$, and set $u \leftarrow su$.
2. Iterate until convergence:
   1. Update Left Singular Vector: $u \leftarrow Xv$
   2. Update Right Singular Vector: $v \leftarrow h_\gamma X^\top u$
   3. Normalize Right Singular Vector: $v \leftarrow \frac{v}{\|v\|}$

Cross-Validation for Sparsity Selection

- **Sparsity parameter**: Tuning parameter controlling number of non-zero loadings in v ($0 =$ dense, $p =$ full sparsity).

Algorithm: K-fold CV Tuning Parameter Selection - Degree of sparsity

1. Randomly group the rows of side-by-side data matrix X into K roughly equal-sized groups, denoted as $X^1, ..., X^K$.
2. For each sparse tuning parameter $j \in \{0, 1, ..., p\}$ (level of sparsity), do the following:
   1. For $k = 1, ..., K$, let X^{-k} be the data matrix X leaving out X^k. Apply Algorithm sFPCA-rSVD on X^{-k} and derive the FPC scores $u^{-k}(j)$. Then project X^k onto $u^{-k}(j)$ to obtain $v^k(j)$.
   2. Calculate the K-fold CV scores defined as: (N is the number of grid points in X^k)

$$CV_j = \sum_{k=1}^{K} \frac{\|X^k - u^{-k}(j)v^k(j)\|^2}{N}$$

3. Select the degree of sparsity as $j_0 = \arg\min\{CV(j)\}$.

Overview of Existing Approaches

- **Smooth FPCA:**
  - Pros: Produces smooth eigenfunctions.
  - Algorithm: Penalized power iteration.
  - Tuning: α (smoothness) via GCV.
- **Sparse FPCA:**
  - Pros: Feature selection $\rightarrow$ interpretable.
  - Algorithm: sFPCA-rSVD algorithm.
  - Tuning: γ (sparsity) via CV.
- **Combined Approaches:** Smooth + Sparse together.

$$\min_{u,v} \|X - uv^\top\|_F^2 + \alpha v^\top \Omega v + p_\gamma(v)$$

- Trade-off: **Variance explained vs Interpretability vs Smoothness**.

Regularized MFPCA

- **Context**
  - Univariate FDA → **Multivariate FDA** (e.g., simultaneously recorded EEG channels, growth patterns of multiple anatomical measures.)
  - MFPCA → **joint modes of variation** across functions
- **Challenges**
  - Discretization & irregular grids → noise, missing data
  - High dimensionality and limited sample size → unstable eigenfunctions (sensitive to small fluctuations in the data)
  - Cross-function correlation → requires enforcing smoothness both within and across functions
- **Proposed Solution: Penalized SVD**
  - **Smoothness penalties**: roughness on derivatives
  - **Sparsity penalties**: Soft, hard, or SCAD
  - **Block-diagonal roughness matrix** for cross-function structure
- **Impact**
  - Produces **smooth, sparse, interpretable** joint modes
  - More stable & applicable to high-dimensional multivariate FDA

Methodology: Multivariate Functional Data Framework

- A multivariate functional dataset is formed by **concatenating p functional data matrices**.
  - Each variable: $X_i \in \mathbb{R}^{n \times m_i}$ where n: number of observations and m_i: grid points
- **Rank-one approximation** (per variable):

$$X_i \approx u_i v_i^\top, \quad u_i \in \mathbb{R}^n, \; v_i \in \mathbb{R}^{m_i}$$

Methodology: Multivariate Functional Data Framework

- A multivariate functional dataset is formed by **concatenating p functional data matrices**.
  - Each variable: $X_i \in \mathbb{R}^{n \times m_i}$ where n: number of observations and m_i: grid points
- **Rank-one approximation** (per variable):

$$X_i \approx u_i v_i^\top, \quad u_i \in \mathbb{R}^n, \ v_i \in \mathbb{R}^{m_i}$$

- **Full data matrix**: $\mathbf{X} = [X_1 \quad X_2 \quad \cdots \quad X_p] \in \mathbb{R}^{n \times \sum_{i=1}^p m_i}$

$$\boldsymbol{X} = \begin{bmatrix} x_{11}(t_{11}) & \cdots & x_{11}(t_{1,m_1}) & \cdots & x_{1p}(t_{p1}) & \cdots & x_{1p}(t_{p,m_p}) \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1}(t_{11}) & \cdots & x_{n1}(t_{1,m_1}) & \cdots & x_{np}(t_{p1}) & \cdots & x_{np}(t_{p,m_p}) \end{bmatrix}.$$

Penalized Smooth MFPCA

- Standard FPCA loadings may be noisy; smoothness penalties (via block-diagonal Ω_i) improve structure and interpretability.
- Let $\boldsymbol{X} \in \mathbb{R}^{n \times M}$ denote multivariate functional data, where $M = \sum_{i=1}^{p} m_i$.
  Its best rank-one approximation is $\boldsymbol{X} \approx uv^\top$,
  with $u \in \mathbb{R}^n$ (score vector) and $v = [v_1, v_2, ..., v_p]^\top \in \mathbb{R}^M$ (loading vector).
  A smoothness penalty is imposed on v.

Penalized Smooth MFPCA

- Standard FPCA loadings may be noisy; smoothness penalties (via block-diagonal Ω_i) improve structure and interpretability.
- Let $\boldsymbol{X} \in \mathbb{R}^{n \times M}$ denote multivariate functional data, where $M = \sum_{i=1}^{p} m_i$.
  Its best rank-one approximation is $\boldsymbol{X} \approx uv^\top$,
  with $u \in \mathbb{R}^n$ (score vector) and $v = [v_1, v_2, ..., v_p]^\top \in \mathbb{R}^M$ (loading vector).
  A smoothness penalty is imposed on v.
- The block-diagonal penalty matrix is $\boldsymbol{\Omega} = \text{diag}(\Omega_1, \Omega_2, \ldots, \Omega_p)$, where each $\Omega_i \in \mathbb{R}^{m_i \times m_i}$ is a univariate roughness penalty matrix.

Penalized Smooth MFPCA

- Standard FPCA loadings may be noisy; smoothness penalties (via block-diagonal Ω_i) improve structure and interpretability.
- Let $\boldsymbol{X} \in \mathbb{R}^{n \times M}$ denote multivariate functional data, where $M = \sum_{i=1}^{p} m_i$.
  Its best rank-one approximation is $\boldsymbol{X} \approx uv^\top$,
  with $u \in \mathbb{R}^n$ (score vector) and $v = [v_1, v_2, \ldots, v_p]^\top \in \mathbb{R}^M$ (loading vector).
  A smoothness penalty is imposed on v.
- The block-diagonal penalty matrix is $\boldsymbol{\Omega} = \mathrm{diag}(\Omega_1, \Omega_2, \ldots, \Omega_p)$, where each $\Omega_i \in \mathbb{R}^{m_i \times m_i}$ is a univariate roughness penalty matrix.
- The penalized reconstruction error is

$$\min_{u,v} \|\boldsymbol{X} - uv^\top\|_F^2 + \boldsymbol{\alpha}^\top (v^\top \boldsymbol{\Omega} v),$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_p)^\top$ controls smoothness.

MFPCA Power Algorithm

Algorithm: Regularized Power Iteration for Smooth MFPCA

1. Initialize v.
2. Repeat until convergence:
   1. $u \leftarrow \mathbf{X}v$
   2. $v \leftarrow (I + \alpha\boldsymbol{\Omega})^{-1}\mathbf{X}^\top u$
   3. $v \leftarrow v/\|v\|$
3. Update $\mathbf{X} \leftarrow \mathbf{X} - \sigma uv^\top$ to extract the next PC.

- The smoothing operator is $\boldsymbol{S(\alpha)} = (I + \alpha\boldsymbol{\Omega})^{-1} \in \mathbb{R}^{M\times M}$.

MFPCA Power Algorithm

Algorithm: Regularized Power Iteration for Smooth MFPCA

1. Initialize v.
2. Repeat until convergence:
   1. $u \leftarrow \mathbf{X} v$
   2. $v \leftarrow (I + \alpha\boldsymbol{\Omega})^{-1}\mathbf{X}^\top u$
   3. $v \leftarrow v/\|v\|$
3. Update $\mathbf{X} \leftarrow \mathbf{X} - \sigma u v^\top$ to extract the next PC.

- The smoothing operator is $\boldsymbol{S}(\boldsymbol{\alpha}) = (I + \boldsymbol{\alpha\Omega})^{-1} \in \mathbb{R}^{M \times M}$.
- The smoothing parameter $\boldsymbol{\alpha}$ is selected via generalized cross-validation (GCV), defined as

$$\mathrm{GCV}(\boldsymbol{\alpha}) = \frac{1}{M} \frac{\|(I - \boldsymbol{S}(\boldsymbol{\alpha}))(\mathbf{X}^T u)\|^2}{\left(1 - \frac{1}{M}\mathrm{tr}\{\mathbf{S}(\boldsymbol{\alpha})\}\right)^2}. \tag{2}$$

Penalized Sparse Multivariate FPCA

- **Goal:** Extend sparse FPCA to multivariate functional data, imposing **sparsity** (select important regions) and **smoothness** (reduce noise).
- **Sparsity penalties:** Soft, hard, or SCAD thresholding Shen and Huang [2008], Zhenhua Lin and Wang [2017], Nie and Cao [2020].

Penalized Sparse Multivariate FPCA

- **Goal:** Extend sparse FPCA to multivariate functional data, imposing **sparsity** (select important regions) and **smoothness** (reduce noise).
- **Sparsity penalties:** Soft, hard, or SCAD thresholding Shen and Huang [2008], Zhenhua Lin and Wang [2017], Nie and Cao [2020].
- Sparsity parameters: $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p)$, where γ_i ranges from 0 (no sparsity) to m_i for each variable.

Algorithm: Regularized Power Iteration for Smooth MFPCA

1. Initialization: Compute rank-one SVD of $\boldsymbol{X}$, $\boldsymbol{X} \approx suv^\top$, and set $u \leftarrow su$.
2. Iterate until convergence:
   1. Update left singular vector: $u \leftarrow \boldsymbol{X} v$
   2. Update right singular vector: $v \leftarrow \boldsymbol{h_\gamma} \boldsymbol{X}^\top u$
   3. Normalize right singular vector: $v \leftarrow \dfrac{v}{\|v\|}$

Smooth and Sparse Multivariate FPCA

- The combined implementation of smoothness and sparsity on the loading vector v in multivariate functional data is achieved by the following algorithm:

Algorithm: Regularized Power Iteration for Smooth MFPCA

1. Initialize unit vectors u and v using SVD of $\boldsymbol{X}$ (best rank-one approximation of $\boldsymbol{X}$)
2. Repeat till convergence
    1. $u \leftarrow \boldsymbol{X} v$
    2. $v \leftarrow \boldsymbol{S}(\alpha)\boldsymbol{h}(\gamma_v)\boldsymbol{X}^\top u$
    3. $v \leftarrow \frac{v}{\|v\|}$
3. Update $\boldsymbol{X} = \boldsymbol{X} - \sigma u v^\top$ and proceed to find the next principal component.

- Algorithm CV Tuning for Sparsity and equation (2) are used to tune the sparsity level via K-fold CV and the smoothing parameter via GCV, respectively.

Simulation: Estimation Performance

- **Data-generating process:** Two functional variables:

$$X_{ij}^{(1)} = u_{i1} v_{11}(t_j) + u_{i2} v_{12}(t_j) + \epsilon_{ij}^{(1)}, \quad X_{ij}^{(2)} = u_{i1} v_{21}(t_j) + u_{i2} v_{22}(t_j) + \epsilon_{ij}^{(2)},$$

  - where $u_{i1} \sim N(0, \sigma_1^2)$, $u_{i2} \sim N(0, \sigma_2^2)$, $\epsilon_{ij}^{(k)} \sim N(0, \sigma^2)$, and $n = m = 101$, $t_j \in [-1, 1]$

- **True functional PCs:**
  - Variable 1: $v_{11}(t) = \frac{t+\sin(\pi t)}{s_1}, \quad v_{12}(t) = \frac{\cos(3\pi t)}{s_2}$
  - Variable 2:

$$v_{21}(t) = \begin{cases} \dfrac{\sin(3\pi t)}{s_3}, & t \in (-\frac{1}{3}, \frac{1}{3}), \\ 0, & \text{otherwise}, \end{cases} \qquad v_{22}(t) = \begin{cases} \dfrac{\sin(2\pi t)}{s_4}, & t \leq -\frac{1}{3}, \\ \dfrac{\sin(\pi t)}{s_4}, & t \geq \frac{1}{3}, \\ 0, & \text{otherwise}. \end{cases}$$

Here, s_1, s_2, s_3, s_4 are normalizing constants ensuring unit L^2 norm.

Simulation: Estimation Performance

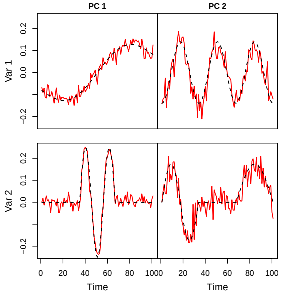
Scenarios tested:

1. Unpenalized Multivariate SVD (baseline)
2. Smoothed Multivariate SVD (smoothness penalty)
3. Sparse Multivariate SVD (sparsity penalty)
4. Sparse + Smoothed Multivariate SVD (combined regularization)

Simulation: Estimation Performance

Scenarios tested:

1. Unpenalized Multivariate SVD (baseline)
2. Smoothed Multivariate SVD (smoothness penalty)
3. Sparse Multivariate SVD (sparsity penalty)
4. Sparse + Smoothed Multivariate SVD (combined regularization)

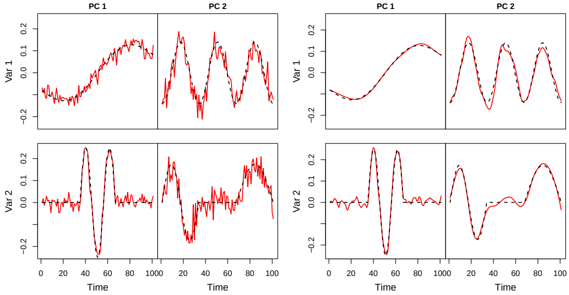


(a) MFPCA

Simulation: Estimation Performance

Scenarios tested:

1. Unpenalized Multivariate SVD (baseline)
2. Smoothed Multivariate SVD (smoothness penalty)
3. Sparse Multivariate SVD (sparsity penalty)
4. Sparse + Smoothed Multivariate SVD (combined regularization)

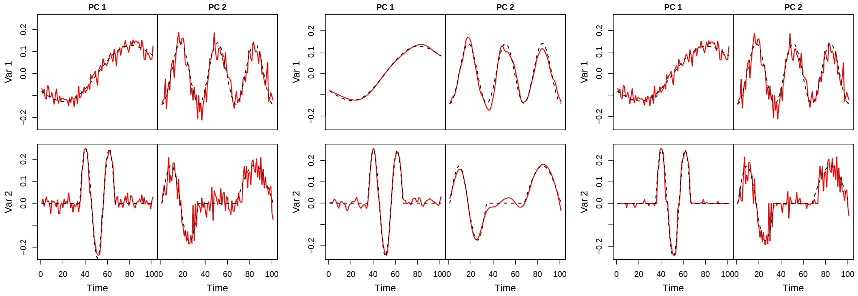


(a) MFPCA (b) Smoothed MFPCA

Simulation: Estimation Performance

Scenarios tested:

1. Unpenalized Multivariate SVD (baseline)
2. Smoothed Multivariate SVD (smoothness penalty)
3. Sparse Multivariate SVD (sparsity penalty)
4. Sparse + Smoothed Multivariate SVD (combined regularization)

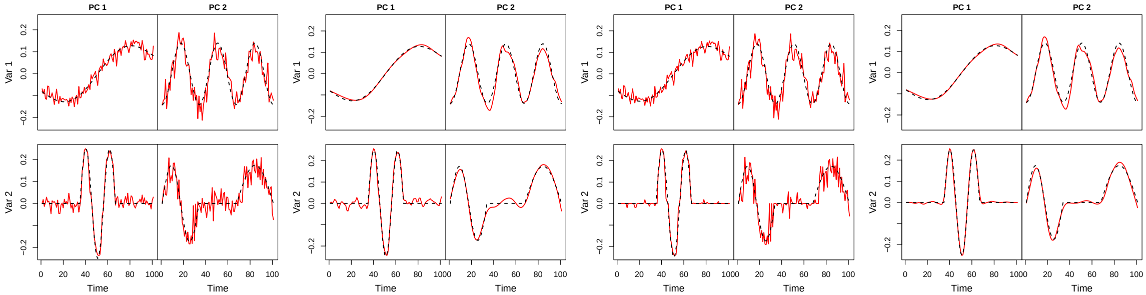


(a) MFPCA (b) Smoothed MFPCA (c) Sparse MFPCA

Simulation: Estimation Performance

Scenarios tested:

1. Unpenalized Multivariate SVD (baseline)
2. Smoothed Multivariate SVD (smoothness penalty)
3. Sparse Multivariate SVD (sparsity penalty)
4. Sparse + Smoothed Multivariate SVD (combined regularization)



(a) MFPCA (b) Smoothed MFPCA (c) Sparse MFPCA (d) Smoothed and Sparse MFPCA

Simulation: Estimation Performance

Accuracy measures:

1. Variable-wise MSE:

$$\mathrm{MSE}_{k\ell} = \frac{1}{m}\sum_{j=1}^{m}(\hat{v}_{k\ell}(t_j) - v_{k\ell}(t_j))^2$$

2. Replication-averaged MSE:

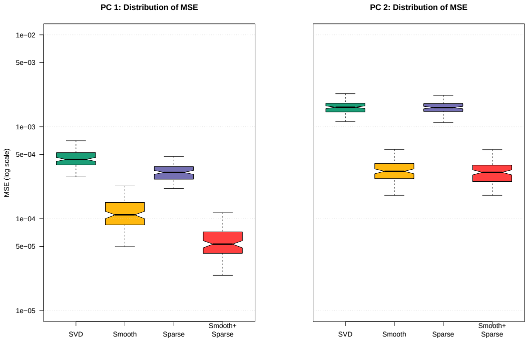
$$\overline{\mathrm{MSE}}_{k\ell} = \frac{1}{R}\sum_{r=1}^{R}\mathrm{MSE}_{k\ell}^{(r)}$$

3. Multivariate MSE:

$$\mathrm{MSE}_{\ell}^{(\mathrm{multi})} = \frac{1}{m}\sum_{j=1}^{m}\sum_{k=1}^{2}(\hat{v}_{k\ell}(t_j) - v_{k\ell}(t_j))^2$$

Simulation: Estimation Performance

- **Performance across four methods (SVD, Smooth, Sparse, Smooth+Sparse):**
  - Smoothness and/or sparsity **reduce MSE** compared to unregularized SVD.
  - **Smooth+Sparse** yields lowest error and most stable estimates.
  - Smooth estimator performs consistently well; sparsity alone less effective (esp. for PC2).
  - Joint regularization achieves best **bias–variance tradeoff**.



PC1: Quartiles and Mean log10(MSE)

Method	Q1	Median	Mean	Q3
SVD	-3.41	-3.35	-3.34	-3.28
Smooth	-4.07	-3.96	-3.92	-3.82
Sparse	-3.57	-3.50	-3.49	-3.43
Smooth+Sparse	**-4.38**	**-4.28**	**-4.22**	**-4.14**

PC2: Quartiles and Mean log10(MSE)

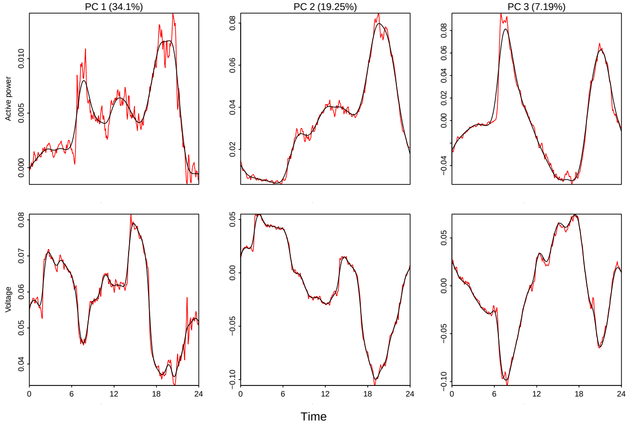
Method	Q1	Median	Mean	Q3
SVD	-2.84	-2.79	-2.79	-2.75
Smooth	-3.56	-3.48	-3.47	-3.40
Sparse	-2.83	-2.79	-2.79	-2.75
Smooth+Sparse	**-3.59**	**-3.50**	**-3.49**	**-3.42**

Application: Household Power Consumption

- **Dataset:** Bivariate functional data including active power and voltage consumption [Hebrail and Berard, 2012] for one household between December 2006 and November 2010.
- **Scaling:** To equalize the contribution of each variable in the multivariate analysis, we rescale them following [Happ and Greven, 2018].

$$\tilde{X}_j(t_i) = \hat{w}_j^{1/2} X_j(t_i),$$

$$\hat{w}_j = \left(\frac{1}{m} \sum_{i=1}^{m} \widehat{\mathrm{Var}}(X_j(t_i)) \right)^{-1}.$$



First 3 PCs: MFPCA (red) vs ReMFPCA (black)

Regularization reduces noise while preserving the dominant daily consumption patterns, enhancing interpretability without losing key structure.

Two-way Regularized MFPCA

- **Two-way functional data:** Each observation is a *matrix of curves*, with smooth variation across **two domains** (e.g., time $\times$ space in air quality, time $\times$ channels in EEG).

Two-way Regularized MFPCA

- **Two-way functional data:** Each observation is a *matrix of curves*, with smooth variation across **two domains** (e.g., time $\times$ space in air quality, time $\times$ channels in EEG).
- **Limitation of standard FPCA [Ramsay and Silverman, 2005]:**
  - Focuses on one domain (often time).
  - Penalties applied only to loadings $\rightarrow$ ignores structure in second domain.
  - Results may be rough or overly dense along the unpenalized axis.

Two-way Regularized MFPCA

- **Two-way functional data:** Each observation is a *matrix of curves*, with smooth variation across **two domains** (e.g., time $\times$ space in air quality, time $\times$ channels in EEG).
- **Limitation of standard FPCA [Ramsay and Silverman, 2005]:**
  - Focuses on one domain (often time).
  - Penalties applied only to loadings $\rightarrow$ ignores structure in second domain.
  - Results may be rough or overly dense along the unpenalized axis.
- **Two-way FPCA [Jianhua Z. Huang and Buja, 2009]:**
  - Introduced **smoothness penalties** on both scores and loadings.
  - Produces coherent, interpretable *component surfaces* instead of jagged approximations.

Two-way Regularized MFPCA

- **Two-way functional data:** Each observation is a *matrix of curves*, with smooth variation across **two domains** (e.g., time $\times$ space in air quality, time $\times$ channels in EEG).
- **Limitation of standard FPCA [Ramsay and Silverman, 2005]:**
  - Focuses on one domain (often time).
  - Penalties applied only to loadings $\rightarrow$ ignores structure in second domain.
  - Results may be rough or overly dense along the unpenalized axis.
- **Two-way FPCA [Jianhua Z. Huang and Buja, 2009]:**
  - Introduced **smoothness penalties** on both scores and loadings.
  - Produces coherent, interpretable *component surfaces* instead of jagged approximations.
- **Our contribution:**
  - Extend to **multivariate functional data** (multiple functional variables).
  - Combine **smoothness** + **sparsity** **penalties** in both directions.
  - Result: Low-rank, interpretable, noise-robust principal components for high-dimensional applications.

Two-way Smoothed MFPCA: Setup & Penalty

- Two-way multivariate functional data: $\boldsymbol{X} \in \mathbb{R}^{n\times M}, \quad M = \sum_{i=1}^{p} m_i.$
- Roughness matrices: $\boldsymbol{\Omega}_u \in \mathbb{R}^{n\times n}$, $\boldsymbol{\Omega}_v \in \mathbb{R}^{M\times M}$ (symmetric, non-negative definite).
- Smoothers: $\boldsymbol{S}_u(\alpha_u) = (I + \alpha_u \boldsymbol{\Omega}_u)^{-1}, \qquad \boldsymbol{S}_v(\boldsymbol{\alpha}_v) = (I + \boldsymbol{\alpha}_v \boldsymbol{\Omega}_v)^{-1}.$

Two-way Smoothed MFPCA: Setup & Penalty

- Two-way multivariate functional data: $\boldsymbol{X} \in \mathbb{R}^{n \times M}, \quad M = \sum_{i=1}^{p} m_i$.
- Roughness matrices: $\boldsymbol{\Omega}_u \in \mathbb{R}^{n \times n}$, $\boldsymbol{\Omega}_v \in \mathbb{R}^{M \times M}$ (symmetric, non-negative definite).
- Smoothers: $\boldsymbol{S}_u(\alpha_u) = (I + \alpha_u \boldsymbol{\Omega}_u)^{-1}, \qquad \boldsymbol{S}_v(\boldsymbol{\alpha}_v) = (I + \boldsymbol{\alpha}_v \boldsymbol{\Omega}_v)^{-1}$.
- Penalized rank-one reconstruction:

$$\min_{u,v} \|\boldsymbol{X} - uv^\top\|_F^2 + \mathcal{P}(u, v)$$

- Penalty [Jianhua Z. Huang and Buja, 2009]:
  $\mathcal{P}(u, v; \alpha_u, \boldsymbol{\alpha}_v) = u^\top(\alpha_u \boldsymbol{\Omega}_u)u\,\|v\|^2 + \|u\|^2\, v^\top(\boldsymbol{\alpha}_v \boldsymbol{\Omega}_v)v + u^\top(\alpha_u \boldsymbol{\Omega}_u)u\; v^\top(\boldsymbol{\alpha}_v \boldsymbol{\Omega}_v)v$.
- Multivariate v: $\boldsymbol{\Omega}_v = \mathrm{diag}(\Omega_1, \ldots, \Omega_p)$.

Two-way Smoothed MFPCA: Conditional GCV

- Minimizers:

$$u = \frac{\boldsymbol{S}_u(\alpha_u)\boldsymbol{X}v}{v^\top(I+\boldsymbol{\alpha}_v\boldsymbol{\Omega}_v)\,v} = \frac{\boldsymbol{S}_u(\alpha_u)}{1+\boldsymbol{\alpha}_v\boldsymbol{R}_v(v)}\,\frac{\boldsymbol{X}v}{\|v\|^2}, \qquad v = \frac{\boldsymbol{S}_v(\boldsymbol{\alpha}_v)\boldsymbol{X}^\top u}{u^\top(I+\alpha_u\boldsymbol{\Omega}_u)\,u} = \frac{\boldsymbol{S}_v(\boldsymbol{\alpha}_v)}{1+\alpha_u\boldsymbol{R}_u(u)}\,\frac{\boldsymbol{X}^\top u}{\|u\|^2}.$$

- Rayleigh quotients: $\boldsymbol{R}_u(u) = \frac{u^\top\boldsymbol{\Omega}_u u}{\|u\|^2}, \qquad \boldsymbol{R}_v(v) = \frac{v^\top\boldsymbol{\Omega}_v v}{\|v\|^2}.$

Two-way Smoothed MFPCA: Conditional GCV

- Minimizers:

$$u = \frac{\boldsymbol{S}_u(\alpha_u)\boldsymbol{X}v}{v^\top(I+\boldsymbol{\alpha}_v\boldsymbol{\Omega}_v)\,v} = \frac{\boldsymbol{S}_u(\alpha_u)}{1+\boldsymbol{\alpha}_v\boldsymbol{R}_v(v)}\,\frac{\boldsymbol{X}v}{\|v\|^2}, \qquad v = \frac{\boldsymbol{S}_v(\boldsymbol{\alpha}_v)\boldsymbol{X}^\top u}{u^\top(I+\alpha_u\boldsymbol{\Omega}_u)\,u} = \frac{\boldsymbol{S}_v(\boldsymbol{\alpha}_v)}{1+\alpha_u\boldsymbol{R}_u(u)}\,\frac{\boldsymbol{X}^\top u}{\|u\|^2}.$$

- Rayleigh quotients: $\boldsymbol{R}_u(u) = \frac{u^\top\boldsymbol{\Omega}_u u}{\|u\|^2}, \qquad \boldsymbol{R}_v(v) = \frac{v^\top\boldsymbol{\Omega}_v v}{\|v\|^2}.$
- Conditional GCV criteria:

$$\mathrm{GCV}_u(\alpha_u;\boldsymbol{\alpha}_v) = \frac{\frac{1}{n}\left\|\left(I-\frac{\boldsymbol{S}_u(\alpha_u)}{1+\boldsymbol{\alpha}_v\boldsymbol{R}_v(v)}\right)\frac{\boldsymbol{X}v}{\|v\|^2}\right\|^2}{\left(1-\frac{1}{n}\,\mathrm{tr}\left(\frac{\boldsymbol{S}_u(\alpha_u)}{1+\boldsymbol{\alpha}_v\boldsymbol{R}_v(v)}\right)\right)^2}, \qquad \mathrm{GCV}_v(\boldsymbol{\alpha}_v;\alpha_u) = \frac{\frac{1}{m}\left\|\left(I-\frac{\boldsymbol{S}_v(\boldsymbol{\alpha}_v)}{1+\alpha_u\boldsymbol{R}_u(u)}\right)\frac{\boldsymbol{X}^\top u}{\|u\|^2}\right\|^2}{\left(1-\frac{1}{m}\,\mathrm{tr}\left(\frac{\boldsymbol{S}_v(\boldsymbol{\alpha}_v)}{1+\alpha_u\boldsymbol{R}_u(u)}\right)\right)^2}.$$

- **Optimization:** Alternate updates of u and v using GCV until convergence $\rightarrow$ two-way regularized components.

Two-way Smooth + Sparse MFPCA

- **Goal:** Extract components that are **low-rank, smooth, and sparse**.
  - Smoothness $\rightarrow$ coherent variation across subjects & functions.
  - Sparsity $\rightarrow$ highlights key observations & time regions.
- **Novelty:** First framework to combine **both** in two-way functional data.

Two-way Smooth + Sparse MFPCA

- **Goal:** Extract components that are **low-rank, smooth, and sparse**.
  - Smoothness $\rightarrow$ coherent variation across subjects & functions.
  - Sparsity $\rightarrow$ highlights key observations & time regions.
- **Novelty:** First framework to combine **both** in two-way functional data.

- Data matrix $\boldsymbol{X}$: seek u, v solving:

$$\min_{u,v} \|\boldsymbol{X} - uv^\top\|_F^2 + \sum_j^J \mathcal{P}_j^{[\theta]}(u, v)$$

- J is the number of penalty components, and θ is the vector of all tuning parameters.
- The composite penalty $\sum_{j=1}^J \mathcal{P}_j^{(\theta)}(u, v)$ lets us mix regularizers, e.g., smoothness with $\theta = (\alpha_u, \boldsymbol{\alpha}_v)$ and sparsity with $\theta = (\gamma_u, \boldsymbol{\gamma}_v)$ (controlling sparsity), and can include other structures as needed.

Sequential Power Algorithm

Algorithm: Two-way Smooth + Sparse MFPCA (Sequential Power)

1. Initialization: Rank-one SVD of $\boldsymbol{X}$: $\boldsymbol{X} \approx s\, u^{(0)} v^{(0)\top}$; set $u \leftarrow s\, u^{(0)}$, $v \leftarrow v^{(0)}$.
2. Repeat until convergence:
   1. $u \leftarrow \boldsymbol{S}_u^{[\alpha_u]}\, \boldsymbol{h}_u^{[\gamma_u]}(\boldsymbol{X}\, v)$
   2. $v \leftarrow \boldsymbol{S}_v^{[\alpha_v]}\, \boldsymbol{h}_v^{[\gamma_v]}(\boldsymbol{X}^\top u)$
   3. $v \leftarrow v / \|v\|$
3. $\boldsymbol{X} \leftarrow \boldsymbol{X} - \sigma\, u\, v^\top$ to extract the next component.

- Smoothness parameters are selected with conditional GCV, while sparsity parameters are chosen via cross-validation (CV).

Selection of Regularization Parameters

- Four sets of tuning parameters:
  - α_u: smoothness of u, γ_u: sparsity of u
  - $\boldsymbol{\alpha}_v$: smoothness of v, $\boldsymbol{\gamma}_v$: sparsity of v
- **Challenge:** Ordering of tuning (smoothness vs. sparsity) affects convergence and solutions.

Selection of Regularization Parameters

- Four sets of tuning parameters:
  - α_u: smoothness of u, γ_u: sparsity of u
  - $\boldsymbol{\alpha}_v$: smoothness of v, $\boldsymbol{\gamma}_v$: sparsity of v
- **Challenge:** Ordering of tuning (smoothness vs. sparsity) affects convergence and solutions.
- **Strategy: Conditional tuning**
  1. Initialize all penalties at 0.
  2. Tune γ_u via K-fold CV.
  3. Sequentially tune $\gamma_{v,i}$ using Algorithm:Two-way Smooth + Sparse MFPCA.
  4. With sparsity fixed, tune α_u by GCV.
  5. Tune $\alpha_{v,i}$ using two-way GCV.
  6. Iterate steps 2–5 until stable.
- This alternating scheme **isolates sparsity vs. smoothness** while ensuring accuracy + interpretability.

K-Fold CV algorithm for Sparsity

K-Fold CV (Row Sparsity)

1. Split $\boldsymbol{X} \in \mathbb{R}^{n \times M}$ into K column groups $\{\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(K)}\}$.
2. For each γ_j and $k = 1, \ldots, K$:
   1. Train on $\boldsymbol{X}^{(-k)}$, estimate $u_j^{(-k)}$.
   2. Validate: $v_j^{(k)} = \boldsymbol{X}^{(k)\top} u_j^{(-k)}$.
   3. Fold error:

$$\mathrm{Err}_j^{(k)} = \frac{1}{\tilde{M}} \|\boldsymbol{X}^{(k)} - u_j^{(-k)} (v_j^{(k)})^\top\|_F^2.$$

3. CV score: $\widehat{\mathrm{CV}}_j = \frac{1}{K} \sum_k \mathrm{Err}_j^{(k)}$.
4. Select $j_0 = \arg\min_j \widehat{\mathrm{CV}}_j$.

K-Fold CV + 1-SE Rule

1. Use same folds to collect $\mathrm{Err}_j^{(k)}$.
2. Compute mean $\widehat{\mathrm{CV}}_j$ and SE $\widehat{\mathrm{SE}}_j$:

$$\widehat{\mathrm{SE}}_j = \sqrt{\frac{1}{K(K-1)} \sum_k (\mathrm{Err}_j^{(k)} - \widehat{\mathrm{CV}}_j)^2}.$$

3. Let $j^\star = \arg\min_j \widehat{\mathrm{CV}}_j$.
4. Choose sparsest j_0 with $\widehat{\mathrm{CV}}_j \le \widehat{\mathrm{CV}}_{j^\star} + \widehat{\mathrm{SE}}_{j^\star}$.

K-Fold CV algorithm for Sparsity

K-Fold CV (Column Sparsity)

1. Split $\boldsymbol{X} \in \mathbb{R}^{n \times M}$ into K row groups $\{\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(K)}\}$.
2. For each γ_j and $k = 1, \ldots, K$:
   1. Train on $\boldsymbol{X}^{(-k)}$, estimate $v_j^{(-k)}$.
   2. Validate: $u_j^{(k)} = \boldsymbol{X}^{(k)} v_j^{(-k)}$.
   3. Fold error:

$$\mathrm{Err}_j^{(k)} = \frac{1}{\tilde{n}} \|\boldsymbol{X}^{(k)} - u_j^{(k)} (v_j^{(-k)})^\top\|_F^2.$$

3. CV score: $\widehat{\mathrm{CV}}_j = \frac{1}{K} \sum_k \mathrm{Err}_j^{(k)}$.
4. Select $j_0 = \arg\min_j \widehat{\mathrm{CV}}_j$.

K-Fold CV + 1-SE Rule

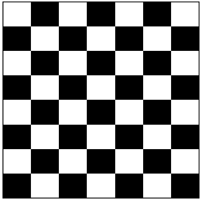
1. Use same folds to collect $\mathrm{Err}_j^{(k)}$.
2. Compute mean $\widehat{\mathrm{CV}}_j$ and SE $\widehat{\mathrm{SE}}_j$:

$$\widehat{\mathrm{SE}}_j = \sqrt{\frac{1}{K(K-1)} \sum_k (\mathrm{Err}_j^{(k)} - \widehat{\mathrm{CV}}_j)^2}.$$

3. Let $j^\star = \arg\min_j \widehat{\mathrm{CV}}_j$.
4. Choose sparsest j_0 with $\widehat{\mathrm{CV}}_j \le \widehat{\mathrm{CV}}_{j^\star} + \widehat{\mathrm{SE}}_{j^\star}$.

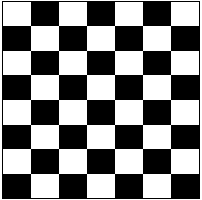
Chessboard

True Chessboard

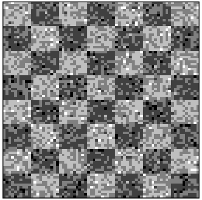


Chessboard

True Chessboard

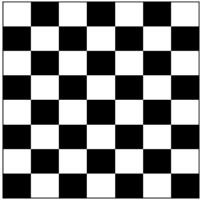


Noisy Chessboard

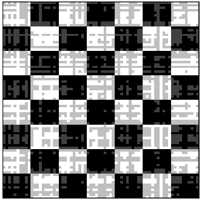


Chessboard

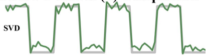
True Chessboard



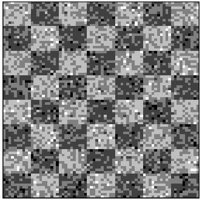
PCA with SVD



First PC Scores (u) Comparison



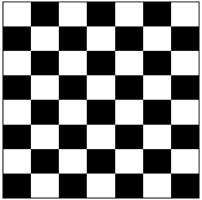
Noisy Chessboard



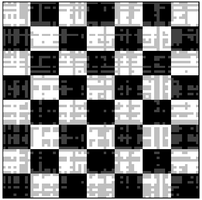
First PC (v) Comparison

Chessboard

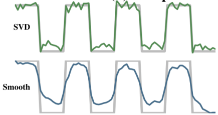
True Chessboard



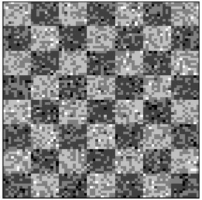
PCA with SVD



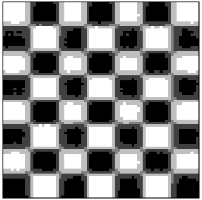
First PC Scores (u) Comparison



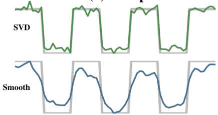
Noisy Chessboard



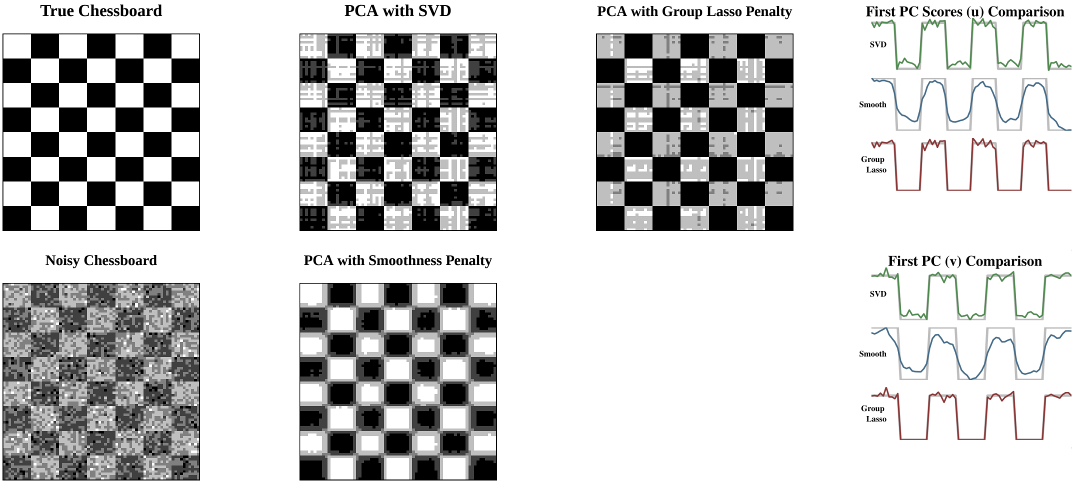
PCA with Smoothness Penalty



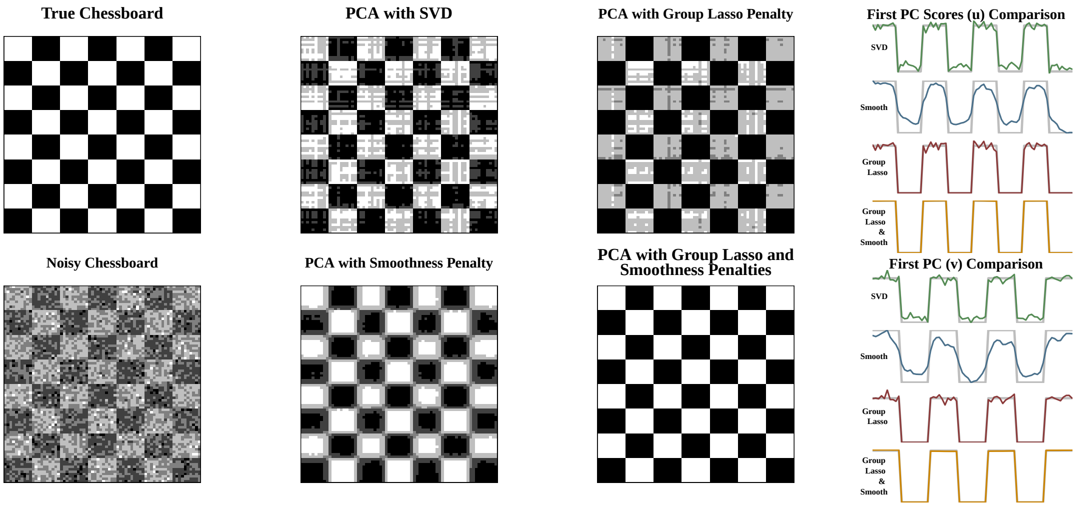
First PC (v) Comparison



Chessboard


True Chessboard
PCA with SVD
PCA with Group Lasso Penalty
First PC Scores (u) Comparison
SVD
Smooth
Group Lasso
Noisy Chessboard
PCA with Smoothness Penalty
First PC (v) Comparison
SVD
Smooth
Group Lasso

Chessboard


True Chessboard
PCA with SVD
PCA with Group Lasso Penalty
First PC Scores (u) Comparison
SVD
Smooth
Group Lasso
Group Lasso & Smooth
Noisy Chessboard
PCA with Smoothness Penalty
PCA with Group Lasso and Smoothness Penalties
First PC (v) Comparison
SVD
Smooth
Group Lasso
Group Lasso & Smooth

Simulation: Two-way Functional Data

- **Data-generating process:**
$X_{ij}^{(1)} = u_{i1} v_{11}(t_j) + u_{i2} v_{12}(t_j) + \epsilon_{ij}^{(1)}, \quad X_{ij}^{(2)} = u_{i1} v_{21}(t_j) + u_{i2} v_{22}(t_j) + \epsilon_{ij}^{(2)},$
- **Latent scores:** generated as smooth functions

$$u_1(s) = \begin{cases} \sin(\pi s), & s > 0, \\ 0, & \text{otherwise}, \end{cases} \qquad u_2(s) = \sin(2\pi s), \quad s \in [-1, 1].$$

- **Functional PCs:**
    - Variable 1: $v_{11}(t) = \frac{t + \sin(\pi t)}{s_1}, \qquad v_{12}(t) = \frac{\cos(3\pi t)}{s_2}$
    - Variable 2:

$$v_{21}(t) = \begin{cases} \dfrac{\sin(3\pi t)}{s_3}, & t \in (-\frac{1}{3}, \frac{1}{3}), \\ 0, & \text{otherwise}, \end{cases} \qquad v_{22}(t) = \begin{cases} \dfrac{\sin(2\pi t)}{s_4}, & t \le -\frac{1}{3}, \\ \dfrac{\sin(\pi t)}{s_4}, & t \ge \frac{1}{3}, \\ 0, & \text{otherwise}. \end{cases}$$

Evaluation Metrics

- **Integrated Squared Error (ISE):**
  For replicate r and component u_1:

$$\mathrm{ISE}_r^{(u_1,\text{method})} = \frac{1}{m}\sum_{j=1}^{m}\left(u_1(t_j) - \widehat{u}_1^{(\text{method})}(t_j)\right)^2.$$

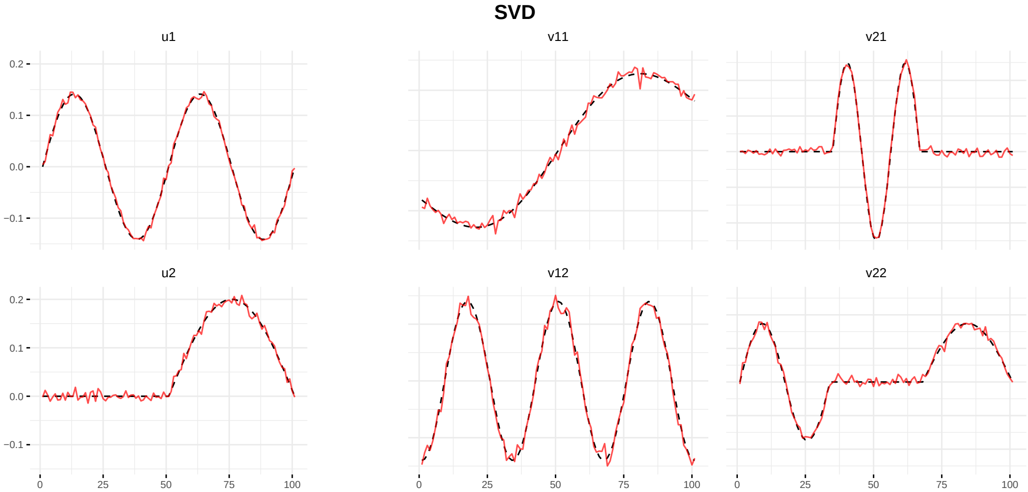
- **Relative ISE (R_ISE):** ratio vs best method

$$R_r^{(u_1,\text{method})} = \frac{\mathrm{ISE}_r^{(u_1,\text{method})}}{\mathrm{ISE}_r^{(u_1,\text{best})}}.$$

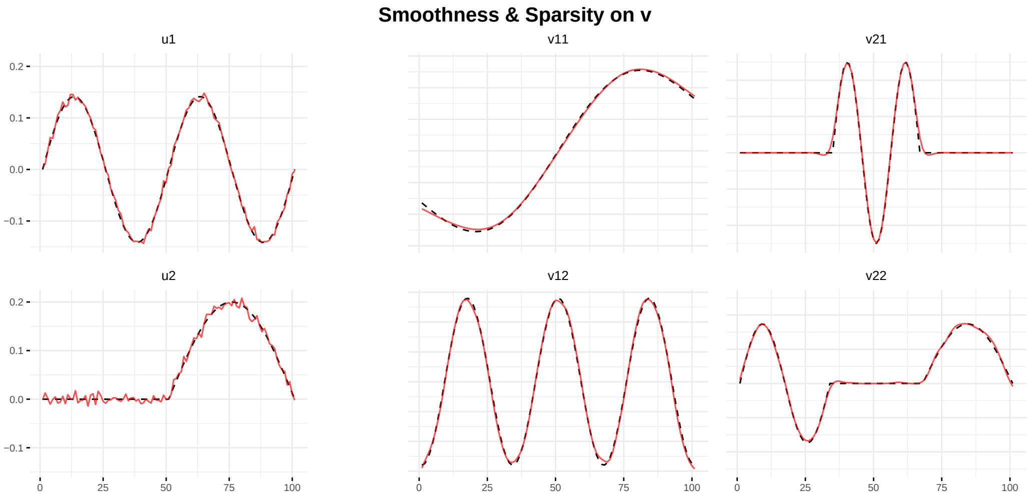
- **Monte Carlo averages:**

$$\overline{R}^{(u_1,\text{method})} = \frac{1}{N}\sum_{r=1}^{N} R_r^{(u_1,\text{method})}, \qquad \mathrm{SE}(\overline{R}) = \sqrt{\frac{1}{N(N-1)}\sum_{r=1}^{N}\left(R_r - \overline{R}\right)^2}.$$

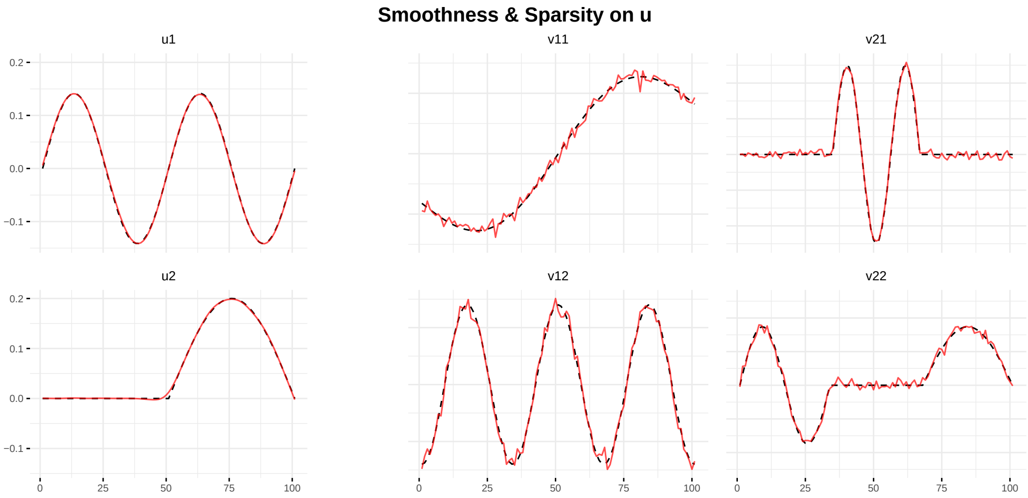
Simulation Results (SVD)



Simulation Results (Smoothness and Sparsity on v)

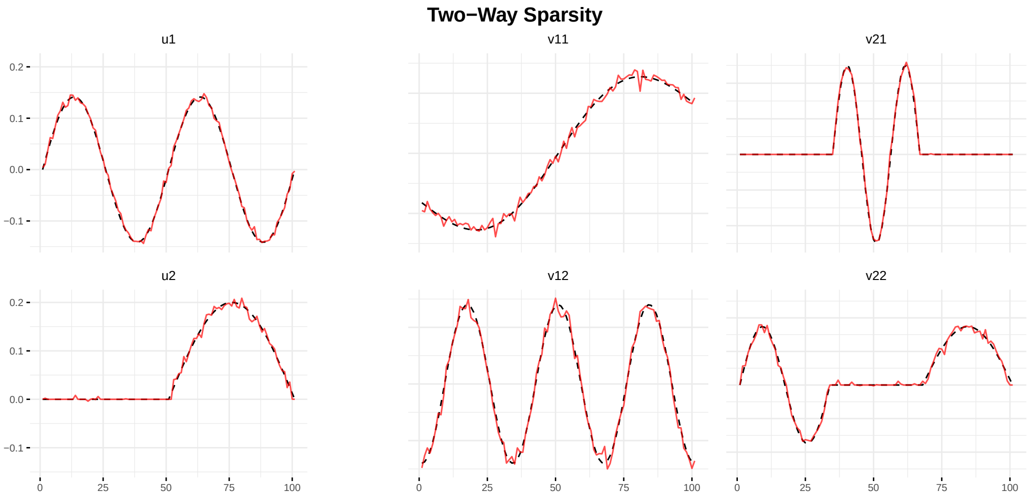


Simulation Results (Smoothness and Sparsity on u)

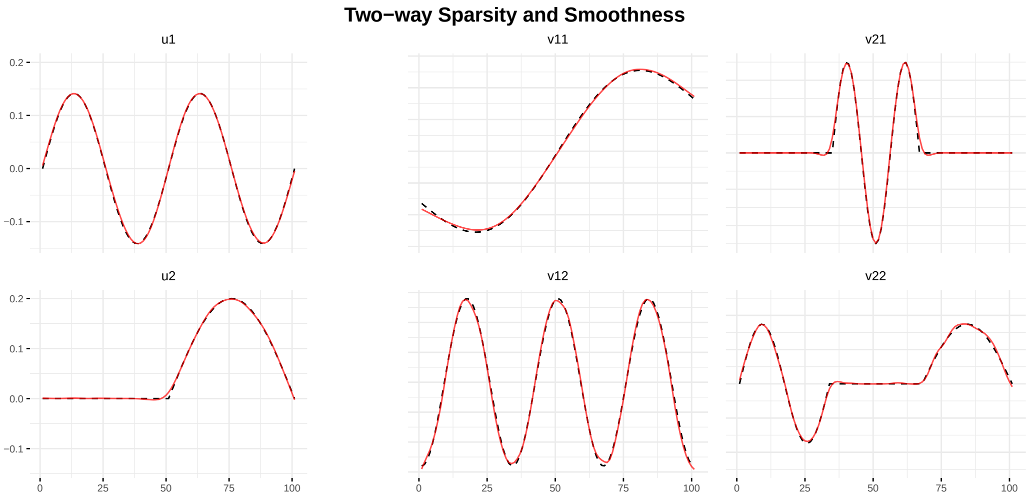


Simulation Results (Two-Way Smoothness)

Simulation Results (Two-Way Sparsity)



Simulation Results (Two-way Sparsity and Smoothness)



Simulation Results

Table 3: Mean ISE for each method and parameter

Method	u1	u2	v1	v2
SVD	0.3651	0.1587	0.00005	0.00010
Smooth+Sparse v	0.3651	0.1587	**0.00002**	**0.00002**
Smooth+Sparse u	**0.3650**	**0.1584**	0.00005	0.00010
Two-way Smoothness	**0.3650**	0.1585	**0.00002**	**0.00002**
Two-way Sparsity	0.3651	0.1586	0.00004	0.00009
Two-way Sm+Sp	**0.3650**	**0.1584**	**0.00002**	**0.00002**

Table 4: Mean Relative ISE for each method and parameter

Method	u1	u2	v1	v2
SVD	1.000	1.001	8.21	5.23
Two-way Sparsity	1.000	1.001	7.71	4.61
Smooth+Sparse v	1.000	1.001	1.05	**1.01**
Smooth+Sparse u	**1.000**	**1.000**	8.19	5.21
Two-way Smoothness	**1.000**	**1.000**	**1.00**	1.21

- Two-way Smooth+Sparse consistently yields the lowest errors across u and v.

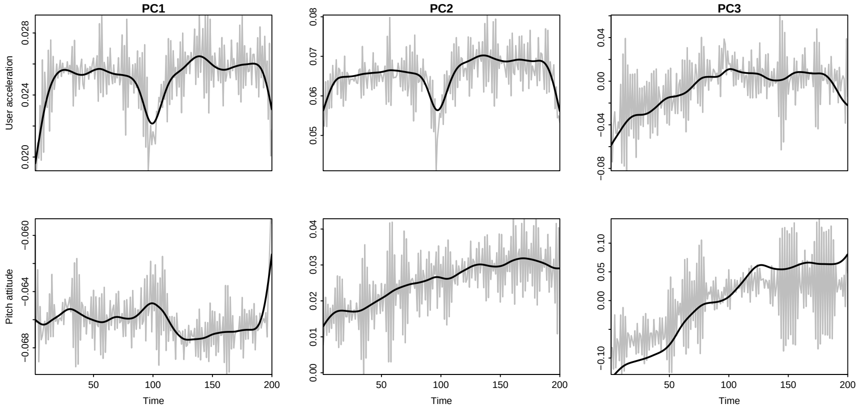
Application: Motion Sense Data

- **Dataset:** Acceleration and pitch from 24 people, 4 activities (jogging, walking, sitting, standing), about 2–3 min each.
- **Goal:** Compare SVD vs **two-way sparse + smooth ReMFPCA** on these multivariate functional signals.
- **Rescaling (Happ & Greven, 2018):** balance variables so each contributes equally.

$$\hat{w}_j = \left(\frac{1}{m} \sum_{i=1}^{m} \widehat{\mathrm{Var}}(X_j(t_i)) \right)^{-1}, \qquad \tilde{X}_j(t_i) = \hat{w}_j^{1/2} X_j(t_i).$$

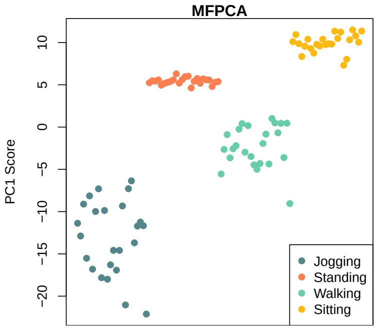
- **Penalties used:** smoothness + sparsity on loadings v; sparsity on scores u (treated as random effects).

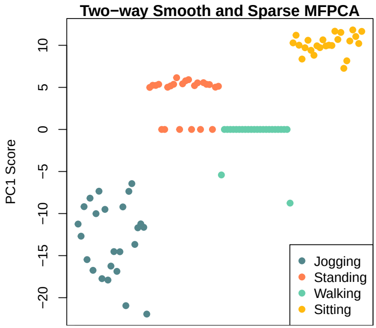
Results: Functional PCs (SVD vs ReMFPCA)



- **SVD (gray):** noisy, high-frequency wiggles.
- **ReMFPCA (black):** smoother, more interpretable PCs capturing dominant structure.

Results: PC Scores and Interpretation





- Sparsity on scores: **PC1 scores for walking about 0** (partially standing too).
- Interpretation: walking contributes little to PC1; removing it **improves interpretability** without hurting fit.
- Takeaway: **Two-way smooth + sparse ReMFPCA** yields cleaner PCs and activity-informative scores.

Conclusion & Future Work

- **Unified FPCA Framework**
  - Combines **smoothness** (denoise, interpretability) + **sparsity** (variable selection).
  - Extends from **univariate** → **multivariate** → **two-way functional data**.
- **Methodology**
  - Penalized SVD with roughness + ℓ_1 penalties.
  - Two-way regularization: smoothness & sparsity on both **scores (u)** and **loadings (v)**.
  - Efficient parameter tuning: **conditional GCV** & **K-fold CV** (with 1-SE rule).
- **Results**
  - Simulations & applications (mortality, call-center, image data).
  - Outperforms one-way or single-penalty methods.
  - Produces **low-rank, denoised, interpretable components**.

Accessible Implementation: R Package & Future Work

- Implemented in **R package `ReMPCA` (GitHub)**
  - Univariate & multivariate FPCA with penalties.
  - Two-way MFPCA for matrix-valued functions.
  - Automated tuning (CV, GCV, 1-SE rule).
  - Diagnostic tools: variance explained, visualization, heatmaps.
  - Early support for **hybrid data (scalar + functional + image)**.
- **Hybrid Data Extensions**
  - Image–Functional Hybrid PCA $\rightarrow$ simultaneous dimension reduction.
  - Scalar–Functional Integration $\rightarrow$ joint low-dim space.
  - Nonlinear Extensions $\rightarrow$ kernel FPCA, neural nets.
- **Applications:**
  - Neuroimaging
  - Personalized medicine
  - Environmental monitoring

  Takeaway: Smooth + sparse + two-way FPCA offers a **theoretical foundation, practical algorithms, and open software** to enable next-generation functional data analysis.

References

Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456): 1348–1360, 2001.

Hossein Haghbin, Yue Zhao, and Mehdi Maadooliat. Regularized multivariate functional principal component analysis for data observed on different domains. *Foundations of Data Science*, 2025. doi: 10.3934/fods.2025018. URL https://www.aimsciences.org/article/id/68b562c1bd10eb1421fa6ef0.

Clara Happ and Sonja Greven. Multivariate functional principal component analysis for data observed on different (dimensional) domains. *Journal of the American Statistical Association*, 113(522):649–659, February 2018. ISSN 1537-274X.

Georges Hebrail and Alice Berard. Individual household electric power consumption. UCI Machine Learning Repository, 2012.

Jianhua Z. Huang, Haipeng Shen, and Andreas Buja. Functional principal components analysis via penalized rank one approximation. *Electronic Journal of Statistics*, 2:678 – 695, 2008. doi: 10.1214/08-EJS218.

Haipeng Shen Jianhua Z. Huang and Andreas Buja. The analysis of two-way functional data using two-way regularized singular value decompositions. *Journal of the American Statistical Association*, 104(488):1609–1620, 2009.

Yunlong Nie and Jiguo Cao. Sparse functional principal component analysis in a new regression framework. *Computational Statistics & Data Analysis*, 152:107016, 2020. ISSN 0167-9473.

J. Ramsay and B.W. Silverman. *Functional Data Analysis*. Springer Series in Statistics. Springer, 2005. ISBN 9780387400808.

Haipeng Shen and Jianhua Z. Huang. Sparse principal component analysis via regularized low rank matrix approximation. *Journal of Multivariate Analysis*, 99(6): 1015–1034, 2008. ISSN 0047-259X.

Bernard W Silverman. Smoothed functional principal components analysis by choice of norm. *The Annals of Statistics*, 24(1):1–24, 1996.

Liangliang Wang Zhenhua Lin, Jiguo Cao and Haonan Wang. Locally sparse estimator for functional linear regression models. *Journal of Computational and Graphical Statistics*, 26(2):306–318, 2017.

Acknowledgments

This thesis would not have been possible without the support, guidance, and encouragement of many people.

Acknowledgments

Dr. Mehdi Maadooliat:
First and foremost, I am deeply thankful to my advisor, Dr. Mehdi Maadooliat. His mentorship has shaped not only this work but also my outlook as a researcher. From the start of my graduate studies, he has been a constant source of patience, thoughtful feedback, and encouragement. His dedication and belief in my potential have been invaluable in my growth as a student.

Acknowledgments

Dr. Hossein Haghbin

I would also like to acknowledge Dr. Hossein Haghbin, whose early guidance played an important role in directing my research interests. His insights and encouragement helped me approach functional data analysis with clarity and confidence.

Acknowledgments

Committee Members:
My sincere appreciation extends to my committee members, Dr. Rebecca Sanders and Dr. Hossein Haghbin, for generously offering their time and expertise. Their feedback and perspectives have strengthened this thesis and enriched my learning.

Acknowledgments

Department and peers:
I appreciate the Department of Mathematical and Statistical Sciences at Marquette University for a supportive, collaborative environment, and I thank my friends and labmates for their companionship, encouragement, and good humor.

Acknowledgments

My dear family:

Most importantly, I am profoundly indebted to my family. To my parents and brothers, whose love and sacrifices have given me every opportunity to pursue my education. Thank you for your encouragement, for reminding me to keep perspective, and for always believing in me even when I doubted myself.

And to my husband, Pouya, words cannot fully capture my gratitude. His patience, unwavering support, and countless sacrifices have carried me through every stage of this journey. He has been my anchor during challenges and my greatest source of joy in moments of success. This achievement belongs as much to my family as it does to me.

Thank you!

MARQUETTE UNIVERSITY

BE THE DIFFERENCE.

Appendix

Variance Explained: Classical vs Regularized FPCA

- **Classical FPCA:** Loadings v_j orthonormal; scores $u_j = Xv_j$ uncorrelated. Variance explained by first J PCs:

$$\sum_{j=1}^{J} \|u_j\|^2 = \mathrm{trace}(V_J^\top X^\top X\, V_J), \qquad V_J = [v_1, \ldots, v_J].$$

Appendix

Variance Explained: Classical vs Regularized FPCA

- **Classical FPCA:** Loadings v_j orthonormal; scores $u_j = Xv_j$ uncorrelated. Variance explained by first J PCs:

$$\sum_{j=1}^{J} \|u_j\|^2 = \mathrm{trace}(V_J^\top X^\top X\, V_J), \qquad V_J = [v_1, \ldots, v_J].$$

- **Issue under regularization:** smoothness/sparsity break orthogonality → scores become correlated → naive sum $\sum \|u_j\|^2$ **double-counts** variance (cf. Huang et al., 2008).

Appendix

Subspace-Projection Definition of Explained Variance

- Normalize loadings and stack:

$$V_J = [v_1, \dots, v_J], \qquad v_j \leftarrow \frac{v_j}{\|v_j\|}.$$

Appendix

Subspace-Projection Definition of Explained Variance

- Normalize loadings and stack:

$$V_J = [v_1, \ldots, v_J], \qquad v_j \leftarrow \frac{v_j}{\|v_j\|}.$$

- Orthogonal projector onto span $(v_1, \ldots, v_J)$:

$$H_J = V_J\,(V_J^\top V_J)^{-1} V_J^\top,$$

where H_J is a a symmetric idempotent matrix. $(H_J^2 = H_J,\ H_J^\top = H_J)$

Appendix

Subspace-Projection Definition of Explained Variance

- Normalize loadings and stack:

$$V_J = [v_1, \ldots, v_J], \qquad v_j \leftarrow \frac{v_j}{\|v_j\|}.$$

- Orthogonal projector onto span $(v_1, \ldots, v_J)$:

$$H_J = V_J\,(V_J^\top V_J)^{-1} V_J^\top,$$

where H_J is a a symmetric idempotent matrix. $(H_J^2 = H_J,\; H_J^\top = H_J)$

- Projected data and explained variance:

$$X_J = XH_J, \qquad \mathrm{V}_{\text{tot}} = \mathrm{tr}(X^\top X), \qquad \mathcal{V}_J = \|X_J\|_F^2 = \mathrm{tr}\big(H_J X^\top X H_J\big).$$

Appendix

PVE, Incremental PVE, and Properties

- Incremental variance:

$$\Delta\mathcal{V}_j = \mathcal{V}_j - \mathcal{V}_{j-1}, \qquad \mathcal{V}_0 = 0.$$

Appendix

PVE, Incremental PVE, and Properties

- Incremental variance:

$$\Delta\mathcal{V}_j = \mathcal{V}_j - \mathcal{V}_{j-1}, \qquad \mathcal{V}_0 = 0.$$

- **Proportion of variance explained (PVE):**

$$\mathrm{PVE}(J) = \frac{\mathcal{V}_J}{\mathsf{V}_{\text{tot}}}, \qquad \mathrm{pve}_j = \frac{\Delta\mathcal{V}_j}{\mathsf{V}_{\text{tot}}} = \mathrm{PVE}(j) - \mathrm{PVE}(j-1).$$

Appendix

PVE, Incremental PVE, and Properties

- Incremental variance:

$$\Delta\mathcal{V}_j = \mathcal{V}_j - \mathcal{V}_{j-1}, \qquad \mathcal{V}_0 = 0.$$

- **Proportion of variance explained (PVE):**

$$\mathrm{PVE}(J) = \frac{\mathcal{V}_J}{\mathsf{V}_{\mathsf{tot}}}, \qquad \mathrm{pve}_j = \frac{\Delta\mathcal{V}_j}{\mathsf{V}_{\mathsf{tot}}} = \mathrm{PVE}(j) - \mathrm{PVE}(j-1).$$

- **Key properties:**
  - No double-counting (works with correlated scores).
  - Reduces to classical PCA when $V_J^\top V_J = I_J$.
  - Monotone in J ($\mathcal{V}_J$ increases).
  - $\Delta\mathcal{V}_j$ = unique variance added by component j.