Bachelor's Thesis

Regularized Covariance Estimation for Functional Data

Department of Statistics
Ludwig-Maximilians-Universität München

Anna Nazarova

Munich, 22 March 2023

Submitted in partial fulfillment of the requirements for the degree of B. Sc.
Supervised by Dr. Fabian Scheipl

Covariance estimation is of great importance for functional data analysis, not least because it is an essential part of functional principle component analysis. This paper gives an introduction on the basic methodological framework of functional data analysis and presents three methods to estimate the covariance function through penalized tensor product spline smoothing: square smoothing, triangle smoothing and FACE. While square smoothing applies the general procedure of tensor product smoothing of bivariate functions, triangle smoothing and FACE exploit the covariance's symmetry and estimate only half of the spline coefficients, thus providing computationally efficient alternatives. The latter even applies a joint estimation of covariance and measurement-error-variance. In order to compare the methods, a two-part simulation study is conducted for various data types. We find that triangle smoothing has in general a similar accuracy compared to square smoothing and sometimes provides even more precise estimates. Furthermore, in most examined settings, FACE shows a clear dominance over both square and triangle smoothing, in terms of precision.

Contents

List of Figures

List of Tables

1. Introduction

From GDP growth rates and stock prices to magnetic resonance images, functional data has become an increasingly important data type for various scientific fields, ranging from economics to medicine and biology. Thus, functional data analysis (FDA), which focuses on continuously measurable and therefore functional data, such as curves, shapes, images, and higher dimensional surfaces, has become an indispensable field of statistics. In practice, functional data is usually observed over a countable finite set of points in the functions' domain, commonly known as grid points. This paper targets functional data in the form of curves, thus we consider the observations to be realizations of a one-dimensional stochastic process.

The infinite dimensional nature of functional data makes it a rich source of information, but results in difficulties while interpreting the data structure (Wang et al., 2016). Therefore, techniques for dimension reduction are essential in FDA. Functional principle component analysis, which is the most prominent tool in FDA, characterizes the covariance structure in the data and reduces the data's dimensionality through an eigendecomposition of the covariance operator. The covariance operator is defined by the covariance function, which summarizes the dependence across pairwise observations of one curve (Ramsay and Silverman, 2005). Hence, in order to obtain functional principle components, one has to estimate the underlying covariance function first. This thesis focuses on exactly this issue of estimating the bivariate covariance function. Covariance estimation in FDA is carried out through smoothing empirical covariance estimates using e.g. local linear surface smoothers (Yao et al., 2005), or tensor product splines (Wood, 2006). In this work, we will focus on the latter. A prevalent issue in smoothing is overfitting, which results in an excessively "rough" shape of the estimated covariance surface. Therefore, regularization in the form of penalization is required in the estimation procedure.

The outline of the thesis is as follows: Chapter 2 gives a fundamental introduction on the methodological background of FDA by illustrating the modeling of functional data as well as differentiating the data into three types: regular, irregular and fragmentary data. While estimating the covariance function of regular data is fairly easy and broadly discussed in literature, irregular and fragmentary data impose some theoretical and computational challenges. Moreover, Chapter 2 explains penalized spline smoothing for univariate and bivariate functions.
After providing the necessary background, we describe three methods for the covariance estimation based on tensor product splines, in Chapter 3. In order to boost computational time, Reiss and Xu (2019) exploit the symmetry of the covariance function by a bisection of its domain along the diagonal and estimate it on one half. This half is then mirrored on the diagonal. Reiss and Xu (2019) refer to this procedure as "triangle" smoothing, as opposed to "square" smoothing which aims to estimate the covariance on the entire quadratic set of grid points. As an alternative to these two methods, Xiao et al. (2018) propose the FACE-algorithm that, similar to triangle smoothing, considers the covariance's symmetry while simultaneously estimating the covariance function and the measurement-error-variances.
In Chapter 4 we provide a two-part simulation study. During the first simulation (Simulation A) we compare square and triangle smoothing in terms of their precision and assess their differences. The second simulation (Simulation B) compares the accuracy of FACE with triangle and square smoothing. We conclude this paper with a discussion of the simulation results.

2. Methodological Background

This chapter gives a general introduction on the methodological framework of (FDA). The first subsection 2.1 gives a brief overview over the modeling of functional data and its three different data types. Subsection 2.2 introduces a spline-based representation of individual univariate functions using penalized spline smoothing. Extending the concepts of 2.2 to multivariate functions, subsection 2.3 provides the nessecary background for the regularized smoothing of the bivariate covariance function.

2.1. Modeling of Functional Data

FDA aims to examine data which arises in the form of curves, surfaces, images and shapes. In this work, we focus on the former two types, not covering the analysis of images and shapes such as brain and neurological data.
The curves are defined on a compact interval $\mathcal{T} \in \mathbb{R}$. Thus, functional data are usually observed discretely on a grid of length N_i, over the domain $\mathcal{T}$, giving us N_i pairs (t_{ij}, Y_{ij}), where Y_{ij} is the recorded value for the function $X_i(\cdot)$ at $t_{ij} \in \mathcal{T}$. In practice, Y_{ij} is often polluted by measurement error, leading Y_{ij} to possibly deviate from the real $X_i(t_{ij})$.
Because the curves' domain is infinite, functional data is infinite dimensional, making it a rich source of information (Wang et al., 2016, Ramsay and Silverman, 2005).

In general, we model functional data as a random sample of realizations of a one-dimensional stochastic process. Hence, we examine the real-valued functions $X_1(t), ..., X_n(t)$ defined on $\mathcal{T}$, which are observed at some points in $\mathcal{T}$. We refer to the set of points $\{t_{ij} \in \mathcal{T}\}$ where a function value is recorded as grid.
Note that we often assume the underlying stochastic process to be Gaussian and in a Hilbert space such as $L^2(\mathcal{T})$. Furthermore, we impose smoothness on the individual function, i.e. the second derivative of $X_i(t)$ exists and is continuous for all $i = 1, .., n$ (Wang et al., 2016, p. 2 f).
A prominent model for functional data is:

$$Y_{ij} = X_i(t_{ij}) + \epsilon_{ij}, \quad i = 1, ..., n, \ j = 1, ..., N_i,$$

where Y_{ij} is the value of the jth observation of the function $X_i(\cdot)$ made at a random point $t_{ij} \in \mathcal{T}$. The left summand can be specified into $X_i(t_{ij}) = \mu(t_{ij}) + f_i(t_{ij})$, where $\mu(t_{ij})$ is the mean function for the underlying stochastic process and $f_i(t_{ij})$ is a smooth curve-specific variation from $\mu(t_{ij})$ (Cederbaum et al., 2018, Yao et al., 2005).
This gives us

$$Y_{ij} = \mu(t_{ij}) + f_i(t_{ij}) + \epsilon_{ij}, \quad t_{ij} \in \mathcal{T} \tag{2.1}$$

where ϵ_{ij} is the additional measurement error with $E(\epsilon_{ij}) = 0$ and $\text{var}(\epsilon_{ij}) = \sigma^2_{ij}$. The measurement errors are commonly referred to as white noise. We assume the ϵ_{ij} to be independent across i and j. Another popular but not strictly necessary assumption is homoscedasticity, yielding $\sigma^2_{ij} = \sigma^2$ to be constant (Wang et al., 2016, p. 5).

We classify functional data into three types: regular data, irregular data and fragmentary data. Their main difference lies in the way the observations (t_{ij}, Y_{ij}) were sampled.
Usually, we consider the observations to be recorded on a time grid, that is $t_1, ..., t_{N_i}$ are ordered time points.

In practice, regular data are observed by using a recording instrument such as an electroencephalograph (Wang et al., 2016). Then, every function is observed on the same time grid, that is $t_{ij} = t_j$ and $N_i = N$ for every $i = 1, ..., n$. Although in some cases the time points are equidistant, this is not strictly necessary as long as the time grid, which is also known as sampling schedule, is the same for every curve.
In contrast to regular data, the sampling schedules of irregular data are not the same across all curves and an individual's time points are irregularly spaced. Consequently, the number of observations N_i and the time points t_j can differ from curve to curve.
This data type usually arises from longitudinal studies, where an individual is repeatedly measured over time and the number of observations as well as the several time points might vary from individual to individual (Wang et al., 2016, p. 3).
Wang et al. (2016) also consider the distinction of "dense" and "sparse" functional data. They define regular data being densely sampled as opposed to irregular data which is sparsely sampled. For our analysis this extra specification is redundant, wherefore we only distinguish the data types by their sampling schedules. Furthermore, we will assume that all three data types have the same expected values for the number of grid points N_i.
The third data type is known as fragmentary functional data. Similar to irregular data, fragmentary data commonly arises from longitudinal studies where the individual's measurement starts at a random time point (Lin and Wang, 2022). In this setting we only observe a fragment of each curve $X_i(t)$, i.e. all time points are defined on a subinterval $\mathcal{T}_i \subset \mathcal{T}$, which gives us a setting of incomplete functional data (Delaigle et al., 2021). Note, that the subintervals vary from curve to curve. Consequently, fragmentary functional data imposes some challenges in estimating the covariance structure of the data as we will discuss in Chapter 3.

2.2. Spline-based Representation of Functional Data

Since in FDA-applications our main objectives are continous functions, it is only reasonable to give some thoughts on how to record them. Functions are infinite-dimensional in their nature, which prevents us from observing the full curve or surface that we are trying to examine. Hence, our first concern is how to form functional data using the discretely sampled observations y_{ij}. We address this challenge by approximating the univariate functions $x_i(t)$ as we smooth the observations y_{ij}. We consider only the estimation of an individual curve $x(t)$, thus we will renounce the index i.
The generic approach to this estimation problem is to use a spline basis system. A spline is constructed as polynomials of a specific order m defined over L subintervals into which the domain of target function $x(t)$ is divided. These polynomials are joined together at values $\tau_1, ..., \tau_{L-1}$ that are commonly known as *knots*. The order of the spline specifies how many parameters are required to define it in one subinterval, that is one more than its polynomial degree, yielding $d = m + L - 1$ parameters for the whole spline function (Ramsay and Silverman, 2005, p. 47).
A spline *basis*, which is defined by the spline basis functions, spans the space of splines where the true $x(t)$ or its close approximation lies inside (Wood, 2006, p. 122). With this idea, we can represent smooth functions as a linear combination of fully known basis functions $B_k(t)$, $k = 1, ..., d$

$$x(t) = \sum_{k=1}^{d} \gamma_k B_k(t) \tag{2.2}$$

yielding the observations to have the form of the linear model where γ_k are unknown parameters.

Note, that we can re-express (2.2) in matrix terms as

$$\sum_{k=1}^{d} \gamma_k B_k(t) = \boldsymbol{\gamma}'\boldsymbol{b}(t) = \boldsymbol{b}(t)'\boldsymbol{\gamma},$$

where $\boldsymbol{\gamma}$ is the d-vector of parameters and $\boldsymbol{b}$ is the d-vector of basis functions (Ramsay and Silverman, 2005, p. 86).
This allows us to approximate $x(t)$ by estimating the coefficients γ_k by minimizing the least squares criterion $\sum_{j=1}^{N}(y_j - x(t_j))^2$ (Fahmeir et al., 2013, Wood, 2006).
While there exists a variety of spline bases, the most prominent basis system is that of the *B-splines* (see e.g., Fahmeir et al., 2013, Wood, 2006, for a definition). In the following sections we will explore the use of B-Splines to estimate the data's covariance structure. But first we need to contemplate some limitations of the approximation (2.2).

Penalized Spline Smoothing

An essential task in working with splines is choosing an appropriate number of knots, which amounts to choosing the number d of basis functions. Although the fit improves with the number of basis functions, choosing a too high value for d results in overfitting, i.e. the estimated function is locally variable and has an excessively "rough" or "wiggly" form. To counteract this, we formulate a penalized least squares (PLS) criterion, by adding a penalty term to the standard least squares criterion. This additional term $\lambda J_r(x)$ prevents the overfitting to the data by penalizing those estimated functions x that are too rough (Fahmeir et al., 2013, Ch. 8.1.2).
To characterize a function's roughness, we make use of its derivative. Assuming that Eq. (2.2) holds, we define a penalty for the parameters γ_k as the integral of the quadratic rth-order derivative. Our minimization criterion then becomes

$$PLS(\lambda) = \sum_{j=1}^{n}(y_j - x(t_j))^2 + \lambda \int \left[\frac{\partial^r}{\partial t^r} x(t) \right]^2 dt, \tag{2.3}$$

where $\lambda \geq 0$ is the smoothing parameter, controlling the extend of the penalty (Ramsay and Silverman, 2005, p. 84).
For example, a second-derivative penalty in the form of

$$J_2(x) = \int x''(t)^2 dt$$

seems appealing, because it measures of the function's curvature (Fahmeir et al., 2013, p. 433). The choice of the derivative's order dictates the shape of the penalty null space, which is spanned by the functions $x(t)$ for which the penalty term is equal to zero. Hence, the functions in the penalty null space are treated as "completely smooth" and consequently, as $\lambda \to \infty$, the functions estimated by (2.3) are shrinked towards the penalty null space. E.g. the second-derivative penalty term is equal to zero, when $x(t)$ is linear, so the penalty null space is spanned by functions of the form $x(t) = a + b \cdot t$. Analogously, the penalty null space of a first-derivative penalty is spanned by constant functions (Wood, 2017, Wood et al., 2013). Using the matrix notation of (2.2), we can re-express $J_r(x) = \int \left[\frac{\partial^r}{\partial t^r} x(t) \right]^2 dt$ into

$$J_r(x) = \int \left[\frac{\partial^r}{\partial t^r} \boldsymbol{\gamma}'\boldsymbol{b}(t) \right]^2 dt = \gamma' \boldsymbol{R} \gamma,$$

where $\boldsymbol{R} = \int \left[\frac{\partial^r}{\partial t^r}\boldsymbol{b}(t)\frac{\partial^r}{\partial t^r}\boldsymbol{b}(t)'\right] dt$ is referred to as rth-order penalty matrix (Ramsay and Silverman, 2005, p. 87). To apply penalized spline smoothing, we plug (2.2) into (2.3) and then minimize (2.3) with respect to the parameters γ_k.
Choosing a B-Spline basis for (2.2), requires an approximation for the rth-order derivatives. Therefore, Eilers and Marx (1996) propose P-Splines, which approximate the penalty term in (2.3) through a difference penalty of the parameters. E.g., the proposed difference penalty that approximates $J_2(x)$ is $\sum_{j=3}^{d}(\Delta^2\gamma_j)^2$ with the second-order difference-operator Δ^2, recursively defined by

$$\begin{aligned}\Delta^1\gamma_j &= \gamma_j - \gamma_{j-1}\\ \Delta^2\gamma_j &= \Delta^1\Delta^1\gamma_j = \Delta^1\gamma_j - \Delta^1\gamma_{j-1} = \gamma_j - 2\gamma_{j-1} + \gamma_{j-2}.\end{aligned}$$

Analogously, the rth-order differences, defined by $\Delta^r\gamma_j = \Delta^{r-1}\gamma_j - \Delta^{r-1}\gamma_{j-1}$, are the approximations of the rth-order derivative penalty in $J_r(x)$.
For the construction of the P-Splines' penalty matrix $\boldsymbol{R} = \boldsymbol{K}^r$, difference matrices D_r are used, yielding $\boldsymbol{R} = D_r'D_r$. As with the difference-operator Δ^r, the difference matrices are recursively defined by first considering the first-order difference matrix

$$\underset{(d-1)\times d}{D_1} = \begin{pmatrix} -1 & 1 & & \\ & -1 & 1 & \\ & & \ddots & \ddots \\ & & & -1 \quad 1 \end{pmatrix}.$$

With D_1 we can then write the vector of first differences as

$$D_1\boldsymbol{\gamma} = \begin{pmatrix} \gamma_2 - \gamma_1 \\ \vdots \\ \gamma_d - \gamma_{d-1} \end{pmatrix}.$$

Higher differences can be expressed as $D_r = D_1D_{r-1}$. Accordingly, the P-Splines' penalty term is defined as

$$\sum_{k=r+1}^{d}(\Delta^r\gamma_k)^2 = \boldsymbol{\gamma}'D_r'D_r\boldsymbol{\gamma} = \boldsymbol{\gamma}'\boldsymbol{K}^r\boldsymbol{\gamma}$$

(Fahmeir et al., 2013).

2.3. Tensor Product Smooths

The concepts of subsection 2.2 can be extended to multivariate functions. In this work, we are especially interested in estimating an individual's two-dimensional covariance function $x(s,t)$. Therefore, we will introduce the representation of bivariate functions using tensor product bases. To construct a bivariate basis, we first consider the univariate bases of s and t, respectively. From the previous szbsection we know, that we can represent the smooth functions $x(s)$ and $x(t)$ by the basis expansion

$$x(s) = \sum_{i=1}^{d_s}\beta_iB_i^{(1)}(s) \quad \text{and} \quad x(t) = \sum_{l=1}^{d_t}\gamma_lB_l^{(2)}(t),$$

where β_i and γ_l are the parameters and $B_i^{(1)}(s)$ and $B_l^{(2)}(t)$ are the known marginal basis functions. To obtain a joint smooth for s and t, we let $x(s)$ vary smoothly with t, by allowing its parameters β_i to vary smoothly with t, yielding

$$\beta_i(t) = \sum_{l=1}^{d_t} \gamma_{il} B_l^{(2)}(t).$$

This gives us the following representation for $x(s,t)$:

$$x(s,t) = \sum_{i=1}^{d_s} \sum_{l=1}^{d_t} \gamma_{il} B_i^{(1)}(s) B_l^{(2)}(t) \tag{2.4}$$

with the tensor product $B_i^{(1)}(s)B_l^{(2)}(t) = B_{il}(s,t)$ (Wood, 2006).
Given the usual Kronecker product $\otimes$ we can rewrite (2.4) into a matrix notation using the d-vector of basis functions $\boldsymbol{b}$:

$$\left[\boldsymbol{b}(t)' \otimes \boldsymbol{b}(s)'\right] \boldsymbol{\gamma}, \tag{2.5}$$

with $\boldsymbol{\gamma} = (\gamma_{11}, ..., \gamma_{d_s 1}, ..., \gamma_{1 d_t}, ..., \gamma_{d_s d_t})'$ and

$$\boldsymbol{b}(t)' \otimes \boldsymbol{b}(s)' = (B_{11}(s,t), ..., B_{d_s 1}(s,t), ..., B_{1 d_t}(s,t), ..., B_{d_s d_t}(s,t)).$$

We refer to the basis (2.4) as *tensor product basis*. When using polynomial splines such as B-Splines as basis functions, the tensor product basis is known as *tensor product splines* (Fahmeir et al., 2013).

For a given set of observations for s and t we can compute values for $x(s,t)$ and store them in a vector $\boldsymbol{v}$. As with univariate functions, we estimate the parameter vector $\boldsymbol{\gamma}$ by setting up a linear model of the form $\boldsymbol{v} = \boldsymbol{Z}\boldsymbol{\gamma}$ with response vector $\boldsymbol{v}$ and a model matrix $\boldsymbol{Z}$. The jth row of $\boldsymbol{Z}$ corresponds to the jth observation and is defined as

$$\boldsymbol{z}_j = (B_{11}(s_j,t_j), ..., B_{d_s 1}(s_j,t_j), ..., B_{1 d_t}(s_j,t_j), ..., B_{d_s d_t}(s_j,t_j)) = \boldsymbol{b}(t_j)' \otimes \boldsymbol{b}(s_j)'$$

(Reiss and Xu, 2019, p. 6; Fahmeir et al., 2013, p. 505). Consider again the marginal smooths of $x(s)$ and $x(t)$, respectively. Given the same set of observations for s, the model matrix for the estimation of parameters β_i is denoted as $\boldsymbol{Z_s}$. The model matrix $\boldsymbol{Z_t}$ for the parameters in the marginal smooth of $x(t)$ is similarly defined. It is straightforward to see, that we can write $\boldsymbol{z}_j$ as:

$$\boldsymbol{z}_j = \boldsymbol{z_s}_j \otimes \boldsymbol{z_t}_j,$$

where $\boldsymbol{z_s}_j$ and $\boldsymbol{z_t}_j$ is the jth row of $\boldsymbol{Z_s}$ and $\boldsymbol{Z_t}$, respectively. It is worth pointing out, that we can extend this construction for as many covariates as required (Wood, 2006, p. 164).

Tensor Product Penalties

Similar to the spline-based-representation of univariate functions, we need to address for the "roughness" of the estimated bivariate function. Therefore, recall the penalty term $J_r(x) = \gamma' \boldsymbol{R} \gamma$ for the roughness of the univariate smooth of $x(t)$.

We consider $x_{s|t}(s)$ being $x(s,t)$ as a function of s only, when t is held constant. We define $x_{t|s}(t)$ similarly. Then, we can measure the bivariate function's roughness through Wood's tensor product:

$$\lambda_s \int J_{r_s}(x_{s|t}) dt + \lambda_t \int J_{r_t}(x_{t|s}) ds. \tag{2.6}$$

Setting $r_s = r_t = 2$ yields to the popular second-derivative penalty of the form

$$\lambda_s \iint \left[\frac{\partial^2}{\partial s^2} x(s,t)\right]^2 dsdt + \lambda_t \iint \left[\frac{\partial^2}{\partial t^2} x(s,t)\right]^2 dtds$$

(Wood, 2006, p. 165). Additionally, Wood (2006) derives an approximation to the integrals of (2.6) through a reparametrization of the penalty terms. As an alternative, Reiss et al. (2014) provide an exact evaluation of the integrals and suggest to re-express the penalty term into a matrix notation:

$$\mathcal{P} = \lambda_s(\boldsymbol{R}_s \otimes \boldsymbol{Q}_t) + \lambda_t(\boldsymbol{Q}_s \otimes \boldsymbol{R}_t), \tag{2.7}$$

with the $d_t \times d_t$-Gram matrix $\boldsymbol{Q}_t$ having (l, m) entries $q_{lm} = \int B_l^{(2)}(t) B_m^{(2)}(t) dt$.
Wood's penalty is currently implemented in the package `mgcv` (Wood, 2017) for R (R Core Team, 2022).

In the case of tensor product B-Splines, Eilers and Marx (2003) propose bivariate P-Splines as an extension of the univariate P-Splines, described in subsection 2.2 to the bivariate case. The penalty term is defined as

$$\boldsymbol{I}_{d_s} \otimes \boldsymbol{K}_t^{r_t} + \boldsymbol{K}_s^{r_s} \otimes \boldsymbol{I}_{d_t},$$

where $\boldsymbol{K}_s^{r_s} = D'_{r_s} D_{r_s}$ and $\boldsymbol{K}_t^{r_t} = D'_{r_t} D_{r_t}$ are univariate difference-penalty matrices of respective order r_s and r_t. $\boldsymbol{I}_{d_t}$ and $\boldsymbol{I}_{d_s}$ are identity matrices (Fahmeir et al., 2013, p. 508). It is straightforward to see, that we can express this penalty term with the notation (2.7) by setting $\boldsymbol{R}_s = \boldsymbol{K}_s^{r_s}$, $\boldsymbol{R}_t = \boldsymbol{K}_t^{r_t}$, $\boldsymbol{Q}_s = \boldsymbol{I}_{d_s}$, and $\boldsymbol{Q}_t = \boldsymbol{I}_{d_t}$.

With the theoretical background provided in this chapter, we are now equipped to focus on the centerpiece of this thesis: the covariance estimation.

3. Functional Data Covariance Estimators

Much like the functions $x_i(t)$, we assume the covariance function $C(t_{ij}, t_{il}) \in L^2(\mathcal{T} \times \mathcal{T})$, with $t_{ij}, t_{il} \in \mathcal{T}$ of an individual curve to be smooth and in a Hilbert space. Geometrically, the covariance can be viewed as a surface defined on the domain $\mathcal{T} \times \mathcal{T} = \mathcal{T}^2$.
Before we start with a deliberation of the covariance estimates, we impose a further restriction on the measurement errors ϵ_{ij} in model (2.1): We assume ϵ_{ij} to be homoscedastic, yielding $\sigma_{ij}^2 = \sigma_i^2$ for all $i = 1, ..., n$.
The covariance is then defined as

$$C(t_{ij}, t_{il}) = \text{cov}(x_i(t_{ij}), x_i(t_{il})) + \sigma_i^2 \delta_{jl}, \tag{3.1}$$

where $\delta = 1$ if $j = l$ and 0 otherwise. The additional term on the surface's diagonal accounts for the measurement errors in the observed data. Because the white noise is independent across all observations, it does not contaminate the covariance between two distinct observations. Only if we consider an observation's variance, i.e. $j = l$, we will have to take the error variance σ_i^2 into account (Yao et al., 2005).

In contrast to regular and irregular data, which are observed at time points on the whole domain $\mathcal{T}$, the time points of fragmentary data are only defined on a subinterval $\mathcal{T}_i$ of $\mathcal{T}$. Thus, in the case of fragmentary data, the points (t_{ij}, t_{il}) at which we observe $x_i(t)$ only cover a fragment $\bigcup_{i=1}^n (\mathcal{T}_i \times \mathcal{T}_i)$ of the covariance's entire domain $\mathcal{T}^2$. Geometrically, the fragment $\bigcup_{i=1}^n (\mathcal{T}_i \times \mathcal{T}_i)$ can be interpreted as a band around the main diagonal of $\mathcal{T}$, whose size varies with the size of the functions' fragments. Consequently, the estimators we are about to present, can only estimate the covariance function on this subset, rather then the whole domain. Delaigle et al. (2021) address for this issue and propose a nonparametric method based on tensor product smooths to estimate $C(t_{ij}, t_{il})$ on $\mathcal{T}^2$ for data, where only a fragment of $\mathcal{T}^2$ is observed, even for increasing sampling size. Alternatively, Lin and Wang (2022) provide an approach to estimate the mean and covariance function of relatively short fragments, which often occur in longitudinal studies.

The usual procedure begins with computing the empirical estimates of the covariance: Let $\hat{\mu}$ be the smooth estimator for the mean function. Well known approaches on estimating the mean function are P-Splines implemented in Xiao et al. (2018) or local polynomial estimates (Fan and Gijbels, 1996) used by Yao et al. (2005).
The empirical estimates are evaluated through

$$G(t_{ij}, t_{il}) = (Y_{ij} - \hat{\mu}(t_{ij}))(Y_{il} - \hat{\mu}(t_{il})). \tag{3.2}$$

We see, that $E[G(t_{ij}, t_{il})] \approx C(t_{ij}, t_{il})$ and assuming that $E[\mu(\hat{t}_{ij})] = \mu(t_{ij})$ holds for all $j = 1, ..., N_i$, we get $E[G(t_{ij}, t_{il})] = C(t_{ij}, t_{il})$ by the definition of the covariance. Thus, we can obtain an estimation by smoothing the empirical estimates using tensor product splines. Since $C(t_{ij}, t_{il})$ is contaminated by white noise, we only smooth $G(t_{ij}, t_{il})$ for $j \neq l$ and treat the diagonal separately, yielding $G(t_{ij}, t_{il}) = \text{cov}(x_i(t_{ij}), x_i(t_{il}))$ (Reiss and Xu, 2019, Yao et al., 2005).

The following subsections present two approaches for the described procedure, both of which are based on penalized tensor product splines. In the first subsection we discuss the effect of smoothing over different domains on the penalization, presented by Reiss and Xu (2019).
As an alternative to the approach of Reiss and Xu (2019), an estimation algorithm developed by Xiao et al. (2018) is presented in subsection 3.2.

3.1. "Square" vs. "Triangle" Smoothing

Being a bivariate function of t_{ij} and t_{il}, we use (2.4) and (2.5) to express the covariance matrix into

$$C(t_{ij}, t_{il}) = \sum_{k=1}^{d} \sum_{p=1}^{d} \gamma_{kp} B_k(t_{ij}) B_p(t_{il}) = \left[\boldsymbol{b}(t_{il})' \otimes \boldsymbol{b}(t_{ij})'\right] \boldsymbol{\gamma}. \tag{3.3}$$

The symmetry of the covariance function requires us to model it in a symmetric way. That is, we demand $\gamma_{kp} = \gamma_{pk}$ to hold for the parameters in (3.3).

Our goal is to estimate the coefficient vector $\boldsymbol{\gamma}$ using a regression model.
Following the principles of the previous chapter, we compute the value of (3.2) of all curves $x_i, ..., x_n$ and distinct time points $t_{ij}, t_{il} \in \mathcal{T}^2$ and store them into a response vector $\boldsymbol{v}$. Then, we set up a model matrix $\boldsymbol{Z}$ having rows $\boldsymbol{b}(t_{ij}) \otimes \boldsymbol{b}(t_{il})$. The regression model amounts into $\boldsymbol{v} = \boldsymbol{Z}\boldsymbol{\gamma}$. The estimated parameter vector $\boldsymbol{\gamma}$ are obtained by minimizing the penalized least squares criterion $\|\boldsymbol{v} - \boldsymbol{Z}\boldsymbol{\gamma}\|^2 + \boldsymbol{\gamma}\mathcal{P}\boldsymbol{\gamma}$, yielding

$$\hat{\gamma} = \left(\boldsymbol{Z}'\boldsymbol{Z} + \mathcal{P}\right)^{-1} \boldsymbol{Z}'\boldsymbol{v},$$

where $\mathcal{P} = \lambda_s(\boldsymbol{R} \otimes \boldsymbol{Q}) + \lambda_t(\boldsymbol{Q} \otimes \boldsymbol{R})$ is a penalty matrix with symmetric $d \times d$ matrices $\boldsymbol{R}$ and $\boldsymbol{Q}$. Note, that this penalty matrix has the same form as (2.7), only with $\boldsymbol{R}_s = \boldsymbol{R}_t$ and $\boldsymbol{Q}_s = \boldsymbol{Q}_t$.
Choosing a second-derivative or second-difference penalty for $\mathcal{P}$, results into the penalty null space, that is spanned by bilinear functions of s and t, having the form

$$C(s, t) = a + b \cdot (s + t) + c \cdot s \cdot t, \tag{3.4}$$

where a, b, c are arbitrary parameters. Analogously, a first-derivative penalty shrinks the estimated covariance toward the two-dimensional space of constant functions on $\mathcal{T}$.
The approach described above is called *"square"* smoothing, because $E[G(t_{ij}, t_{il})]$ is smoothed for all values $t_{ij}, t_{il},\ j \neq l$ on the square region $\mathcal{T}^2$ (Reiss and Xu, 2019).

Yet, the symmetry of the covariance implies $G(t_{ij}, t_{il}) = G(t_{il}, t_{ij})$ and $\gamma_{kp} = \gamma_{pk}$, wherefore smoothing on the entire $\mathcal{T}^2$ seems redundant. Instead of smoothing on the entire square region, we can estimate C on the triangular region $\mathcal{H} = \{(s, t) \in \mathcal{T} : s > t\}$, i.e. we only consider the values of (3.2) for which $t_{ij} > t_{il}$ in the estimation process. Smoothing on $\mathcal{H}$ rather than on $\mathcal{T}^2$ is referred to as *"triangle"* smoothing. With triangle smoothing we obtain an estimate $\hat{C}(t_{ij}, t_{il})$ for $\mathcal{H}$, which is extended on $\mathcal{T}^2$ by taking $\hat{C}(t_{ij}, t_{il}) = \hat{C}(t_{il}, t_{ij})$ for $t_{ij} < t_{il}$.
The geometrical interpretation of triangle smoothing is a bisection of the covariance surface's domain $\mathcal{T}^2$ along the line $s = t$. The covariance surface is then estimated on one half $\mathcal{H}$ of the domain $\mathcal{T}^2$ and subsequently reflected on the diagonal.
Smoothing on $\mathcal{H}$ results in a dimension reduction of the regression model components, wherefore triangle smoothing is computationally more efficient than square smoothing.Yet, smoothing only on the triangular region $\mathcal{H}$ changes the penalty null space. Completely smooth functions on $\mathcal{H}$ have the form $a + bs + ct + dst$. Extending this function to the entire domain $\mathcal{T}^2$ yields

$$C(s, t) = a + b \cdot min(s, t) + c \cdot max(s, t) + d \cdot s \cdot t, \tag{3.5}$$

where a, b, c, d are again arbitrary parameters. Geometrically, such a function has a ridge along the diagonal and two bilinear pieces on either side (Reiss and Xu, 2019).

We see, that square smoothing and triangle smoothing shrink the estimated covariance in different directions. A simulation study to compare these two smoothing methods is provided in chapter 4. However, we first describe a computational efficient alternative to the methods in the next subsection.

3.2. "FACE" Algorithm

Besides square and triangle smoothing, Xiao et al. (2018) developed another approach to estimate the covariance using tensor product splines. Their so called "Fast Covariance Estimation"-, in short "FACE", -Algorithm provides a joint estimation of the covariance function C and the measurement error variance σ_i^2, hence including values t_{ij}, t_{il} with $j = l$ in the smoothing step. In the following we give a summary of the estimation method of FACE.

Recall the model (2.1) for functional data. We impose a further restriction on the white noise and assume the measurement errors to be i.i.d. Gaussian distributed. Note that the measurement errors are independent across all curves as well as from each other, yielding $\epsilon_{ij} \overset{iid}{\sim} N(0, \sigma^2)$.
Now, consider again the expression (3.3) for $j \neq l$, in which we want to estimate the parameter vector $\boldsymbol{\gamma}$. Let $\Gamma = (\gamma_{kp})_{1 \leq k,p \leq d}$ be the according symmetric $d \times d$ parameter matrix. With Γ we can re-write (3.3) into a matrix notation of the form

$$[\boldsymbol{b}(t_{ij}) \otimes \boldsymbol{b}(t_{il})]' \operatorname{vec}(\Gamma). \tag{3.6}$$

Similar to triangle smoothing, we can exploit the symmetry of the covariance function and Γ, such that we only estimate the lower triangle of Γ. Let $\boldsymbol{\gamma}^* = \operatorname{vech}(\Gamma)$ be a vector, that contains the stacked columns of the lower triangle of Γ. Moreover, let V be a duplication matrix, such that $\operatorname{vec}(\Gamma) = V\boldsymbol{\gamma}^*$.

Similar to square and triangle smoothing, FACE estimates Γ through a penalized least squares criterion. Hence, we compute the values of crossproducts (3.2) for every $t_{il} \geq t_{ij}$ for every curve $i = 1, ..., n$. We denote the cross product $G(t_{ij}, t_{il}) = G_{ijl}$. To get a better overview over all pairwise crossproducts, we consider the following notation:
$\boldsymbol{G}_{ij} = \left[G_{ijj}, G_{ij(j+1)}, G_{ij(j+2)}, ..., G_{ijN_i}\right]' \in \mathbb{R}^{N_i - j + 1}$ is computed for every $j = 1, ..., N_i$ and stored in a vector of vectors $\boldsymbol{G}_i = \left[\boldsymbol{G}_{i1}', ..., \boldsymbol{G}_{iN_i}'\right]' = [G_{i11}, G_{i12}, ..., G_{i1N_i}, G_{i22}, G_{i23}, ..., G_{iN_iN_i}]'$.
So, $\boldsymbol{G}_i$ contains all empirical estimates for some curve i. Note that $\boldsymbol{G}_i$ also contains the terms G_{ijj}, that are polluted by measurement error.
In the same spirit, $[\boldsymbol{b}(t_{ij}) \otimes \boldsymbol{b}(t_{il})]$ is evaluated for all grid points $t_{il} \geq t_{il}$, yielding $\boldsymbol{B}_{ij} = [\boldsymbol{b}(t_{ij}), \boldsymbol{b}(t_{ij+1}), ..., \boldsymbol{b}(t_{iN_i})] \otimes \boldsymbol{b}(t_{ij})$ for all $j = 1, ..., N_i$. The evaluations for all $j = 1, ..., N_i$ are stored in the vector of vectors $\boldsymbol{B}_i = [\boldsymbol{B}_{i1}, ..., \boldsymbol{B}_{iN_i}]$.

The response vector is then given by $\boldsymbol{G} = [\boldsymbol{G}_1', ..., \boldsymbol{G}_n']'$. We see, that the response vector is similarly constructed as in triangle smoothing, with the important difference, that $\boldsymbol{G}$ contains polluted values. Besides the response vector, the model matrix must account for the measurement errors as well. Therefore, consider the unit vector $\delta_{ij} = (1, \boldsymbol{0}_{N_i - j}')' \in \mathbb{R}^{N_i - j + 1}$ and $\boldsymbol{\delta}_i = [\delta_{i1}', ..., \delta_{iN_i}']'$. By the definition of the covariance and the form (3.6), we obtain a linear model of the form

$$E[\boldsymbol{G}_i] = [\boldsymbol{B}_i V, \boldsymbol{\delta}_i] \cdot \begin{bmatrix} \boldsymbol{\gamma}^{*\prime} \\ \sigma^2 \end{bmatrix} = \boldsymbol{Z}_i \boldsymbol{\alpha},$$

where $\boldsymbol{Z}_i = [\boldsymbol{B}_i V, \boldsymbol{\delta}_i]$ is the model matrix and $\boldsymbol{\alpha} = [\boldsymbol{\gamma}^{*\prime} \sigma^2]'$ is the coefficient vector. Let $W_i \in \mathbb{R}^{N_i \times N_i}$ be a weight matrix for capturing the correlation of $\boldsymbol{G}_i$. A weighted least squares formula is then defined through

$$WLS = \sum_{i=1}^{n} (\boldsymbol{G}_i - \boldsymbol{Z}_i \boldsymbol{\alpha})' W_i (\boldsymbol{G}_i - \boldsymbol{Z}_i \boldsymbol{\alpha}).$$

Taking $\boldsymbol{B} = [\boldsymbol{B}_1', ..., \boldsymbol{B}_n']'$, $\boldsymbol{\delta} = [\boldsymbol{\delta}_1', ..., \boldsymbol{\delta}_n']'$ and acknowledging that $\boldsymbol{Z} = [\boldsymbol{Z}_1', ..., \boldsymbol{Z}']' = [\boldsymbol{B}V, \boldsymbol{\delta}]$ results in a full matrix notation of the WLS-formula:

$$WLS = (\boldsymbol{G} - \boldsymbol{X}\boldsymbol{\alpha})' \boldsymbol{W} (\boldsymbol{G} - \boldsymbol{X}\boldsymbol{\alpha}),$$

where $\boldsymbol{W} = \text{blockdiag}(W_1, ..., W_n)$.
Consequently, Γ and σ^2 can be estimated simultaneously by minimizing the penalized weighted least squares criterion

$$WLS + \lambda \|\Gamma D\|_F^2,$$

where $\lambda \geq 0$ is the smoothing parameter which controls the extend of the penalty, $\|\cdot\|_F$ denotes the Frobenius norm, and D_r is the rth-order difference matrix which has been introduced in subsection 2.3. The default for FACE is $r = 2$. The Frobenius norm is essentially the trace of $(\Gamma D_r)'(\Gamma D_r)$, thus the penalty term can be re-expressed into

$$\begin{aligned}
\|\Gamma D\|_F^2 &= \text{vec}(\Gamma)'(\boldsymbol{I}_d \otimes D_r D_r')\text{vec}(\Gamma) \\
&= \boldsymbol{\gamma}^* V'(\boldsymbol{I}_d \otimes D_r D_r')V'\boldsymbol{\gamma}^* \\
&= (\boldsymbol{\gamma}^* \ \sigma^2)\boldsymbol{\Theta} \begin{pmatrix} \boldsymbol{\gamma}^* \\ \sigma^2 \end{pmatrix} \\
&= \boldsymbol{\alpha}\boldsymbol{\Theta}'\boldsymbol{\alpha},
\end{aligned}$$

where $\boldsymbol{\Theta} = \begin{pmatrix} \boldsymbol{P} & \boldsymbol{0} \\ \boldsymbol{0} & 0 \end{pmatrix}$ and $\boldsymbol{P} = V'(I_d \otimes D_r D_r')V'$. Note that the term $(\boldsymbol{I}_d \otimes D_r D_r')$ is equivalent to the row penalty of bivariate P-Splines.

The weight matrices W_i are specified as

$$W_i^{-1} = 0.5 \ \text{cov}(\boldsymbol{G}_i^*) + 0.5 \ \text{diag}\{\text{diag}\{\text{cov}(\boldsymbol{G}_i^*)\}\},$$

where $\boldsymbol{G}_i^*$ is defined almost in the same way as $\boldsymbol{G}_i$, except that $\boldsymbol{G}_i^*$ uses the real mean function in the computation of the crossproducts (3.2). Xiao et al. (2018) offer a derivation of $\text{cov}(\boldsymbol{G}_i)$ in terms of $C(t_{ij}, t_{il})$ and σ^2.
Consequently, Xiao et al. (2018) have implemented d a two-stage estimation for FACE:
In the first step, we set $W_i = I$ for all i and minimize a penalized ordinary least squares criterion, in order to obtain a first estimate for $C(t_{ij}, t_{il})$ and σ^2. With this first estimate, we can derive a suitable value for W_i. Having derived W_i, we estimate C and σ^2 again, using a penalized weighted least squares criterion (Xiao et al., 2018).

3.3. Summary of Methods

All three covariance estimators presented in this Chapter utilize the principles of bivariate smoothing through tensor product splines. Square smoothing can be interpreted as the application of the general penalized tensor product bivariate spline smooths, described in subsection 2.3, on the covariance function. Exploiting the symmetry of the covariance, triangle smoothing serves as a computationally efficient alternative, by simply estimating only half of the parameters and consequently reducing the dimension of the regression model. The presence of measurement errors requires square and triangle smoothing to exclude the diagonal values $t_{ij} = t_{il}$ from the estimation and treat them separately. FACE also considers the symmetry of the covariance and reduces the regression model similar to triangle smoothing. Yet, in contrast to triangle

smoothing, FACE includes the values on the diagonal to carry out a joint estimation of both the covariance function as well as the measurement error variance. Moreover, FACE uses a special case of difference penalty and utilizes only one summand of the penalty term (2.7), presented in subsection 2.3.
A crucial part of penalized tensor product smoothing is the estimation of the smoothing parameter λ. Possible techniques for choosing λ include the Un-Biased Risk Estimator (UBRE) as well as Generalized Cross Validation (GCV) (see e.g. Wood, 2017, p. 255 ff.), which are the default methods in `mgcv`. Xiao et al. (2018) propose a tuning algorithm for the leave-one-subject-out cross-validation of the smoothing parameter in FACE.
In the next Chapter we compare the accuracy of the three estimators in a simulation study.

4. Simulation Study

We provide a two-part simulation study to evaluate the performance of the covariance estimators discussed in Chapter 3. In Simulation A, we assess the difference between square and triangle smoothing (Reiss and Xu, 2019) empirically. Simulation B aims to compare the FACE-algorithm (Xiao et al., 2018) from section 3.2 with square and triangle smoothing. Note that by performance we mean the accuracy of the estimation rather than the computational time.

Data Generating Process

We generate N_i realizations of a Gaussian process for either $n = 15$, $n = 30$ or $n = 50$ curves, with a covariance function that will be specified in the following subsections. We consider data of all three types, as discussed in section 2.1, and choose the grid length N_i such that $E(N_i) \in \{20, 35, 70\},\ i = 1, ..., n$. Note that since the grid lengths differ from curve to curve in the case of irregluar and fragmentary data, the expected value is used rather than the absolute value. Fragmentary data is generated in analogy to Delaigle et al. (2021), who sample the intervals of each curve's fragment by generating its length from a uniform distribution.
Furthermore, the data is generated either weakly noisy, i.e. the variance of the measurment errors is relatively low, or very noisy, yielding the measurement errors' variance to be relatively high. To quantify the noise level, we use the signal-to-noise-ratio (SNR) defined as $SNR = E\|X - \mu\|^2/\sigma^2$ (Lin and Wang, 2022, p. 354), i.e. the lower we choose the SNR, the greater the noise becomes in the data. After examining the data and the estimation for different noise levels visually, we decide the SNR to have the value 8 for weakly noisy data and 3 for highly noisy data. We use the R-package `tidyfun` (Scheipl et al., 2022) for the data generation step.
Next, we estimate a covariance matrix with the respective smoothing method. Since the data was generated with a certain covariance function, we can simply derive a true covariance matrix, by evaluating the underlying covariance function for the same grid which we have used in the smoothing step.
To assess the performance of a method, we consider the Frobenius-distance and the Wasserstein-distance for normal distributions between the true covariance matrix and the estimated one. Consequently, one method is assumed to perform better than another, when its corresponding distance measure is smaller than the other's.
Remember that in the case of fragmentary data, the covariance cannot be estimated on its entire domain, but only on a subset. We account for this by only considering those values of the true covariance matrix, where the estimation values are present. Hence, the comparison over the Wasserstein-distance is redundant, and only the Frobenius-distance is considered for fragmentary data.

4.1. Simulation A

In subsection 3.1 we discussed the differences of square and triangle smoothing in terms of their respective penalty null spaces. Moreover, Reiss and Xu (2019) present two real-data examples, which display large differences between the eigendecomposition of the estimated covariance functions. This raises the question of whether triangular smoothing is just as accurate an estimate or even a more accurate estimate than square smoothing and how much the respective estimations differ in practice.
To answer this, we choose two settings for the data generating process. One setting is optimal

for square smoothing with a second-order penalty, while the other setting is optimal for triangle smoothing with a second-order penalty. This choice amounts to choosing an appropriate covariance function for the data generation step, that lies in the second-order penalty null space of the respective method. The form of such a function is given by (3.4) for square smoothing and (3.5) for triangle smoothing in subsection 3.1. We set the arbitrary parameters to $a = c = d = 1$ and $b = 2$. Note that if $b = c = d$, the triange-optimal covariance function has the same form as the square-optimal covariance function. We consider a second-order penalty as well as a first-order penalty. Furthermore, 10 basis functions are used for the estimation step.
The estimation methods are implemented by Fabian Scheipl based on the `fpca.sc` function from the R-package `refund` (Goldsmith et al., 2022). In order to boost computational time, one can choose to either smooth all products of the centered data $G(t_{ij}, t_{il}) = Y_{ij}Y_{il}$ for every curve $i = 1, ...n$, which is the usual procedure yet very slow for large data sets, or to smooth the mean of all products of the centered data over every curve, yielding a reduction of the response vector's dimension and an increase in computational speed. We refer to the latter as "aggregated"-smoothing.
In order to provide a clearer and straightforward presentation of the results, the values in the following tables are rounded to two decimal digits.

Results of Simulation A

Table 1: Overview over the average Wasserstein-distance for different combinations

			Regular				Irregular			
			Sq. Optim.		Tri. Optim.		Sq. Optim.		Tri. Optim.	
SNR	Cuves	Grid Length	Square	Triangle	Square	Triangle	Square	Triangle	Square	Triangle
3	15	20	34.78	35.36	13.41	13.49	472.12	449.11	243.76	215.31
		35	56.17	37.29	24.42	30.27	816.48	848.55	341.52	347.22
		70	106.22	99.63	49.53	38.79	1494.77	1485.23	810.82	659.78
	30	20	29.97	34.96	15.31	13.76	849.24	935.69	389.21	381.99
		35	58.91	52.99	26.10	24.74	1627.51	1420.75	685.44	793.88
		70	110.79	109.04	51.58	52.70	2992.44	3009.72	1013.58	1501.25
	50	20	27.86	31.27	14.85	13.28	1502.29	1422.55	666.45	663.65
		35	60.53	58.62	28.67	24.52	2458.90	2731.50	1258.20	1066.43
		70	117.09	112.53	44.40	52.02	5030.64	4940.25	2294.72	2677.64
8	15	20	24.60	28.75	14.92	12.92	415.00	373.71	172.48	184.70
		35	52.97	58.86	26.06	23.64	824.65	796.94	340.23	251.30
		70	118.95	95.51	54.86	39.66	1400.27	1378.70	480.39	953.43
	30	20	30.29	31.62	13.87	12.98	794.94	811.53	418.31	469.52
		35	57.79	52.71	24.91	24.19	1550.49	1596.79	898.62	728.61
		70	122.42	110.70	58.61	49.43	2969.38	3307.83	1287.77	1195.13
	50	20	35.25	29.66	14.88	14.91	1354.83	1419.39	633.00	603.60
		35	53.11	59.99	24.90	25.86	2538.28	2681.83	1130.51	1218.04
		70	111.44	109.71	50.84	53.87	4759.22	5385.27	2263.51	2162.13

To get a quick overview over the results, we consider Table 1 which displays the average Wasserstein-distance for different settings. It is immediately noticeable that that the average Wasserstein-distance for irregular data is much higher compared to regular data, for every setting. In fact, the average Wasserstein-distance for smoothing irregular data is greater than the average

Wasserstein-distance for smoothing regular data by the value 1344.69. Therefore, a separate analysis of data types is conducted. In the following we will examine the components of Table 1 further.
As mentioned before, our main interest is the comparison between square and triangle smoothing. To quantify the differences between the two smoothing methods, we consider the percentage differences of the average Wasserstein- and Frobenius-distances displayed in Table 2, where a negative value indicates that triangle smoothing had a smaller distance than square smoothing and thus performed better.

Table 2: Percentage difference of average Wasserstein- and Frobenius-distances between triangle smoothing and square smoothing, grouped by penalty order

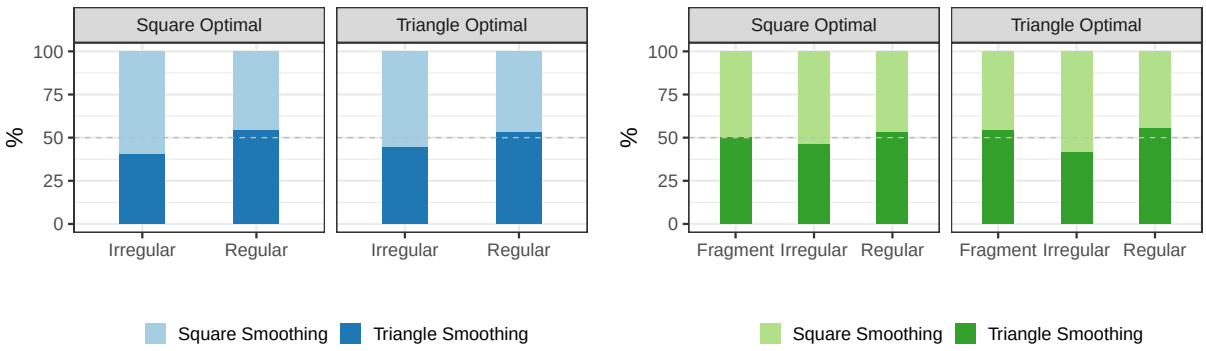
Setting	Type	Wasserstein	Frobenius
Sq. Optim.	Fragment	NaN	0.46
	Irregular	3.38	0.13
	Regular	-4.96	-0.91
Tri. Optim	Fragment	NaN	-1.03
	Irregular	4.86	0.90
	Regular	-5.63	-1.79

Note: percentage difference $= \frac{\text{avg. distance square smooth} - \text{avg. distance triangle smooth}}{\text{avg. distance square smooth}} \times (-100)$

According to the Wasserstein-distance, square smoothing performs better than triangle smoothing for irregular data in both settings. On the other hand, we see that square smoothing has a worse perfromance than triangle smoothing for regular data in both settings. The same observation is made when considering the percentage differences of the Frobenius-distances. Nevertheless, it should be noted that the percentage differences between the Frobenius-distances are relatively low, implying an almost equal performance of triangle and square smoothing. Furthermore, we see, that square smoothing outperforms triangle smoothing for square-optimal fragmentary data, while triangle smoothing performs on average 1.03% better than square smoothing for triangle-optimal fragmentary data.

A graphical visualization is given by the barplot in Figure 1. It shows the percentage of cases where the respective smoothing method could prevail against the other, with the graphic and its bars being grouped by setting and data type respectively. The dark shade of the bar depicts the corresponding percentage of cases where triangle smoothing had a smaller distance than square smoothing, while the light shade corresponds to the corresponding percentage of cases where square smoothing outperformed triangle smoothing.
We see, that the shades of the bars change around the 50% threshold, indicating that the percentages of cases, where the respective smoothing method could prevail against the other, are similar. In Figure 1a), the percentage of cases, where square smoothing outperformed triangle smoothing for irregular data in terms of the Wasserstein-distance, is slightly higher in the square-optimal setting compared to the triangle-optimal setting. This excess of square smoothing in the irregular data and triangle smoothing in the regular data in Figure 1 reflects the result in Table 2.

Having examined the columns of Table 1, we now turn to its rows, which correspond to the main features of the simulated functions. We begin with a combined analysis of the number of



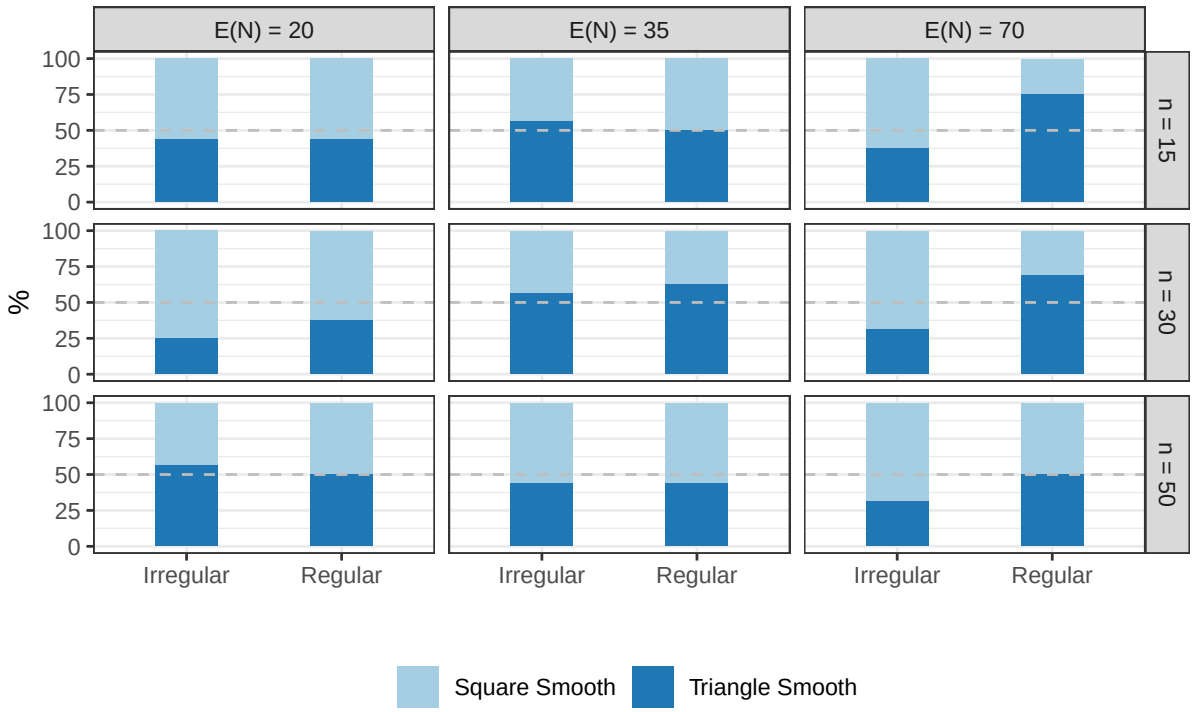
(a) Wasserstein distance

(b) Frobenius distance

Figure 1: Percentage of cases where the respective smoothing method outperformed the other

observed curves and grid lengths in Figure 2, which displays the percentage of cases, where the respective method prevailed against its alternative, grouped by data type. The columns of the figure correspond to the grid length and the rows correspond to the number of curves.
Considering the Wasserstein-distances in Figure 2a), we notice that with an increasing grid length, triangle smoothing performs better than square smoothing for more and more regular cases, where $n = 15$ or $n = 30$. Considering irregular cases with $n = 50$, we observe a decrease in the percentage where triangle smoothing performs better than square smoothing with an increasing grid length. Furthermore, the percentage of regular cases with $E(N) = 70$, where triangle smoothing outperformed square smoothing, decreases with the increasing number of curves.
To analyze fragmentary data, we examine Figure 2b). We obtain the same patterns for regular and irregular data as before in 2a) and there seems to be no pattern for fragmentary data.

The next feature we consider is the noise level, defined by the SNR. Square smoothing performs on average 2.20% better for data with high noise and 4.90% for data with low noise.
In Table 3, the results from Table 2 are further grouped by their SNR values. At first glance, we observe that triangle smoothing performs better than square smoothing for regular data in every setting. This becomes particularly clear for and triangle-optimal regular data with SNR=8, since the average Wasserstein-distance obtained with triangle smoothing is 9.29% smaller than the average Wasserstein-distance obtained with square smoothing.Moreover, triangle smoothing outperforms square smoothing for fragmentary data in every setting, but the square-optimal one with high noise. According to both Wasserstein- and Frobenius-distance, the two smoothing methods have nearly an equal performance for square-optimal irregular data with high noise.
Although square smoothing outperforms triangle smoothing for irregular data with SNR = 8 in both settings, the percentage difference of the Wasserstein-distances is smaller in the triangle-optimal irregular data than in the square-optimal irregular data. Hence, the extend of the outperformance of square smoothing is larger for square-optimal data than for triangle-optimal data. The opposite is the case for triangle smoothing, where the extend of the outperfromance is larger for triangle-optimal data than for square optimal data. Note that this pattern is not to be seen for square-smoothing when considering the Frobenius-distance.
Furthermore we do not see this trend the noisy case with SNR=3, since triangle smoothing even performs even better for square-optimal regular data than it does for triangle-optimal regular data.



(a) Wasserstein-distance

(b) Frobenius-distance

Figure 2: Percentage of cases where the respective smoothing technique outperformed the other, grouped by grid length and number of observed curves

Table 3: Percentage difference of average Wasserstein- and Frobenius-distances between triangle smoothing and square smoothing, grouped by SNR

SNR	Setting	Type	Wasserstein	Frobenius
3	Sq. Optim.	Fragment	NaN	1.60
		Irregular	-0.01	0.00
		Regular	-5.09	-1.29
	Tri. Optim	Fragment	NaN	-0.53
		Irregular	7.83	0.55
		Regular	-1.74	-0.56
8	Sq. Optim.	Fragment	NaN	-0.63
		Irregular	6.89	0.00
		Regular	-4.82	-0.65
	Tri. Optim	Fragment	NaN	-1.57
		Irregular	1.86	1.69
		Regular	-9.29	-2.76

Note: percentage difference $= \frac{\text{avg. distance square smooth} - \text{avg. distance triangle smooth}}{\text{avg. distance square smooth}} \times (-100)$

Additionally, we assess whether the pattern from Figure 2 becomes clearer, when a combined analysis of grid length and SNR is considered. Therefore, the results from Table 3 are further grouped by the expected grid length and displayed in Table 4. The values of the differences between the average Frobenius-distances seem to be proportionate to the differences between the average Wasserstein-distance. For each combination where the percentage difference of the mean Wasserstein-distances is relatively high, the corresponding Frobenius-distance is relatively high as well. Nevertheless, it is noticeable that the conclusion as to whether one method works better than the other can deviate. For example, consider noisy (SNR = 3) regular data in the triangle-optimal case, that has been observed on 20 grid points. According to the Wasserstein-distance triangle smoothing perfroms on average 6.99% better than square smoothing, yet according to the Frobenius-distance square smoothing outperfroms triangle smoothing by 0.07%. What also stands out, are the relatively large percentage differences of the average Wasserstein-distances for data with low noise and $E(N) = 70$. Beside these findings, there are no patterns to be seen.

Table 4: Percentage difference of average Wasserstein- and Frobenius-distances between triangle and square smoothing, grouped by expected grid length and SNR

			Wasserstein			Frobenius		
SNR	Setting	Type	$E(N_i) = 20$	$E(N_i) = 35$	$E(N_i) = 70$	$E(N_i) = 20$	$E(N_i) = 35$	$E(N_i) = 70$
3	Sq. Optim.	Fragment	NaN	NaN	NaN	-0.45	3.27	1.66
		Irregular	-0.58	2.00	-0.87	0.07	0.16	0.06
		Regular	9.70	-15.21	-3.86	2.08	-4.73	-1.20
	Tri. Optim	Fragment	NaN	NaN	NaN	0.45	-1.11	-1.60
		Irregular	-2.96	-3.40	17.47	-1.55	0.01	2.74
		Regular	-6.99	0.43	-1.38	0.07	-0.63	-1.60
8	Sq. Optim.	Fragment	NaN	NaN	NaN	-3.39	0.71	1.07
		Irregular	1.55	3.30	10.33	-0.74	-0.53	1.77
		Regular	-0.11	4.69	-10.46	0.18	0.94	-2.63
	Tri. Optim	Fragment	NaN	NaN	NaN	-1.69	-1.17	-1.01
		Irregular	2.78	-7.23	6.92	1.55	-3.56	6.63
		Regular	-6.54	-2.88	-13.00	-1.75	-1.60	-5.10

Note: percentual difference $= \frac{\text{avg. distance square smoothing} - \text{avg. distance triangle smoothing}}{\text{avg. distance square smoothing}} \times (-100)$

Bear in mind that the estimation in the square-optimal setting is not penalized only when the smoothing is conducted using a second-order penalty. Triangle-smoothing is similarly defined. Hence, we must distinguish our analysis by the penalty orders. Table 5 displays the mean distances of square and triangle smoothing for the two penalty orders first. Surprisingly, the Wasserstein-distance for square smoothing with a second-order penalty is slightly higher than smoothing with a first-order penalty. In what follows, we examine the difference between square and triangle smoothing of square-optimal as well as triangle-optimal data for different penalty orders.

Table 5: Average Wasserstein- and Frobenius-distance for triangle and square smoothing, grouped by penalty order

	Wasserstein		Frobenius	
Pen. Order	Square Smooth	Triangle Smooth	Square Smooth	Triangle Smooth
2	712.94	716.51	2.470	2.4516
1	702.09	748.47	2.472	2.4752

Table 6: Percentage difference of average Wasserstein- and Frobenius-distances between triangle smoothing and square smoothing for square- and triangle-optimal data, grouped by penalty order

Pen. Order	Setting	Type	Wasserstein	Frobenius
2	Sq. Optim.	Fragment	NaN	0.00
		Irregular	0.53	-0.32
		Regular	-0.33	0.00
	Tri. Optim	Fragment	NaN	-2.09
		Irregular	0.85	-1.11
		Regular	-9.59	-3.30
1	Sq. Optim.	Fragment	NaN	0.95
		Irregular	6.27	0.65
		Regular	-9.49	-1.93
	Tri. Optim	Fragment	NaN	0.00
		Irregular	8.94	2.79
		Regular	-1.64	-0.56

Note: percentage difference $= \frac{\text{avg. distance square smooth} - \text{avg. distance triangle smooth}}{\text{avg. distance square smooth}} \times (-100)$

Table 6 further groups the results from Table 2 by penalty order. In Table 6, we see that, compared to a frist-order penalty, the percentage differences of the average Wasserstein-distances are smaller for a second-order penalty. Only for the triangle-optimal regular case, the average Wasserstein-distances differ more than 1%, where triangle smoothing outperforms square smoothing on average by 9.59% for triangle-optimal regular data.
We depict the percentage of cases, where the respective smoothing technique outperformed the other, in terms of the Wasserstein-distance in Figure 3. Figure 2 and Table 4 indicate that the percentage difference of the average Wasserstein-distances between square and triangle smoothing increase with grid length for data with low noise, hence the results in Figure 3 are grouped by SNR and grid length.
Figure 3a) corresponds to smoothing with a second-order penalty while Figure 3b) corresponds to smoothing with a first-order penalty. Much like Figure 2, the lower panels of Figure 3a) show an increase of cases where triangle smoothing prevails against square smoothing with increasing grid length, for regular data with low noise. This trend is not continued for data with SNR = 3. Furthermore, we observe for irregular data on both noise levels, the percentage of cases, where triangle smoothing outperformed square smoothing, is higher for data with $E(N) = 35$ than for data with other expected grid lengths. It is also noticeable, that the percentage of irregular cases with $E(N) = 50$, where triangle smoothing performed better than square smoothing, is lower than 50% for data with low noise, while it is above 50% for noisy data. For regular data, observed on roughly 70 grid points, this observation is the other way round.
In Figure 3b) we notice that, for irregular data with a high noise level, the percentage of cases, where triangle smoothing performed better than square smoothing, decreases with an increasing grid length for both noise levels. For regular data, observed on a grid of expected length of 70, much more cases have triangle smoothing prevailing against square smoothing. On the other hand, for the major part of irregular noisy data, observed on this large grid, square smoothing with a first-order penalty performed better than triangle smoothing with a first-order penalty. Additionally the shade change in the bar is much further away from the 50% threshold in Figure

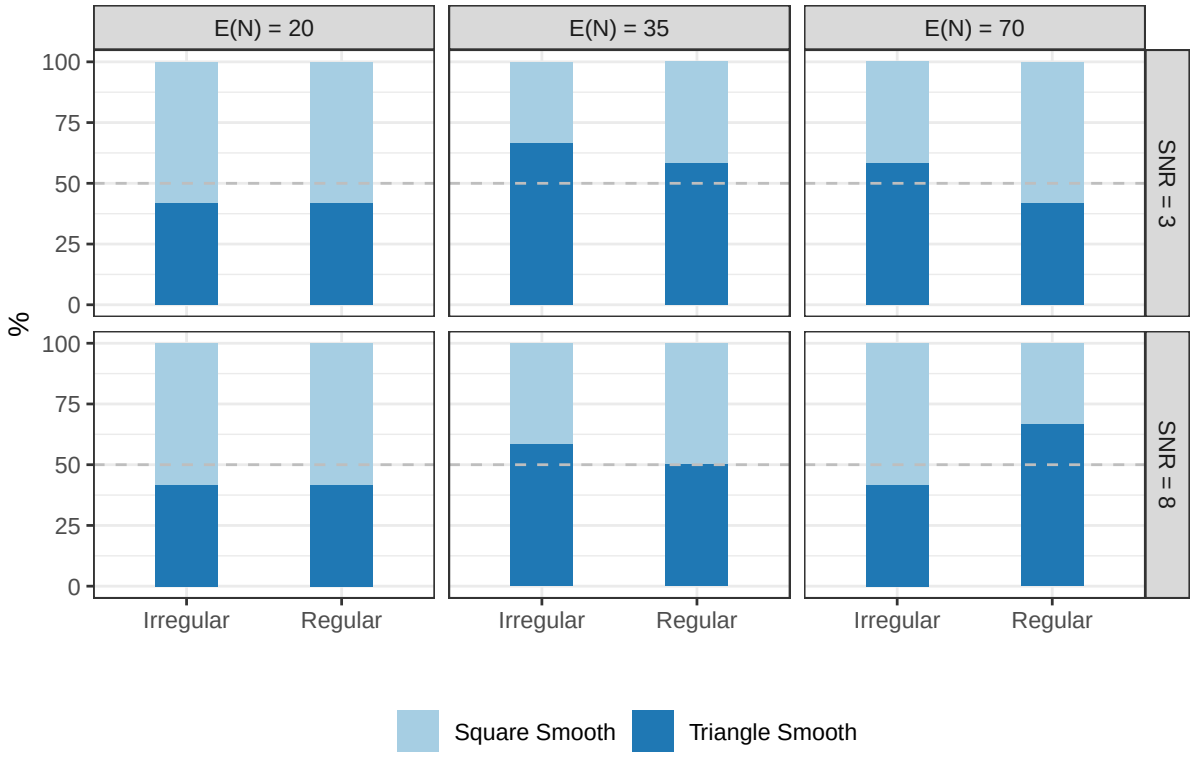
3b) than in Figure 3a), indicating stronger deviations of the smoothing methods' performance. The corresponding illustration of the Frobenius-distances is given in the Appendix B.

As mentioned before, we can smooth the mean of crossproducts across every curve and thus reduce the dimension of the regression model.
In addition to the computational benefit, aggregated smoothing has a positive effect on the performance of both smoothing techniques, as shown by the average Wasserstein- and Frobenius-distance in Table 7.

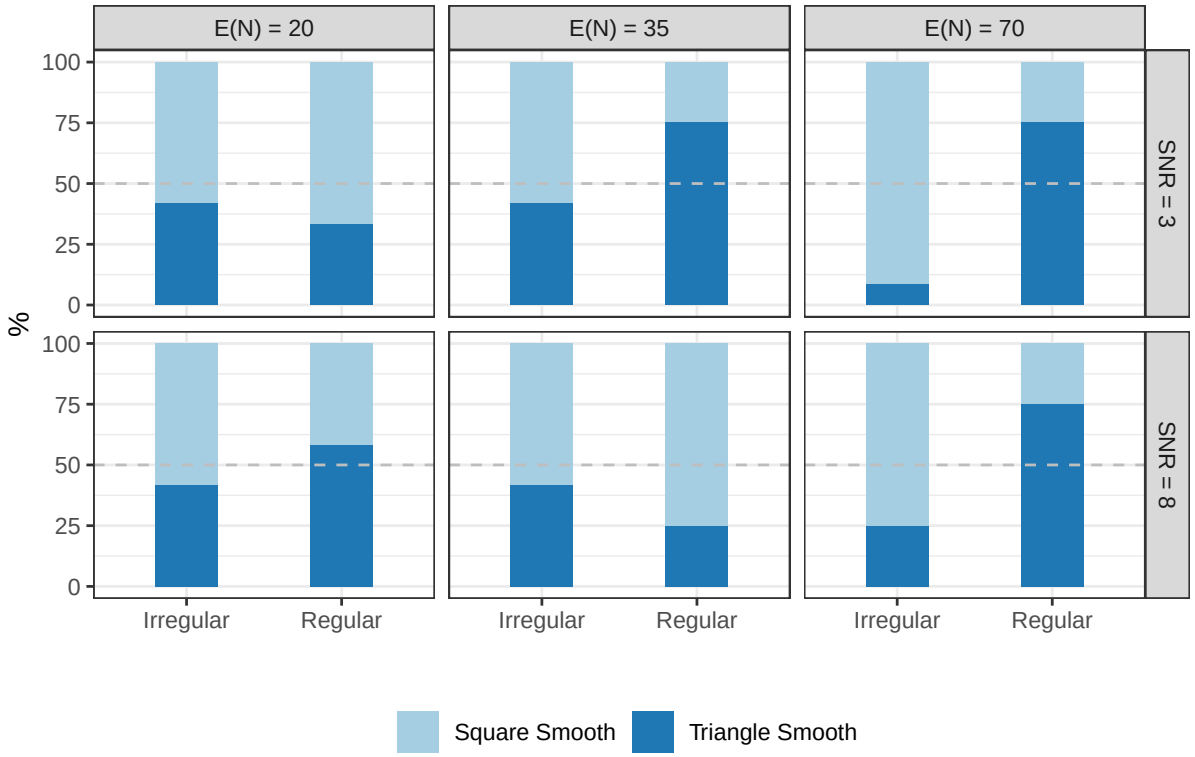
Table 7: Average Wasserstein-distance for aggregated triangle and square smoothing

Aggregated	Square Smooth	Triangle Smooth
False	722.65	741.51
True	692.39	723.47

We see that there are only minor differences between square smoothing and triangle smoothing in general. In the next part of the simulation, we will compare triangle and square smoothing with the FACE-algorithm.



(a) Second-Order Penalty



(b) First-Order Penalty

Figure 3: Percentage of cases where the respective smoothing technique outperformed the other in terms of the Wasserstein-distance, grouped by grid length and SNR

4.2. Simulation B

In this part of the simulation study, we compare the FACE-algorithm, described in subsection 3.2, with square and triangle smoothing. As done before in Simulation A, we generate N_i realizations of a Gaussian process for either $n = 15$, $n = 30$ or $n = 50$ curves such that $E(N_i) \in \{20, 35, 70\}$. The domain of the curves is $\mathcal{T} = [0, 1] \in \mathbb{R}$. Instead of a triangle-optimal or square-optimal setting, we choose only one covariance function, namely the squared exponential covariance in the form of

$$C(s,t) = \exp\left(\frac{-(s-t)^2}{\beta}\right) + \sigma^2\delta_{st},$$

where σ^2 is the variance of the measurement errors and $\delta_{st} = 1$ if $s = t$ and 0 otherwise. The parameter β is an arbitrary scale parameter. Similar to the default implemented in `tidyfun`, we set $\beta = 1/10$.
The data is then smoothed with either square smoothing, triangle smoothing or the FACE-algorithm. As in Simulation A, we consider smoothing with a second-order penalty as well as a first-order penalty. Furthermore, 10 basis functions are used for the spline basis. Table 7 indicates, that aggregated square and triangle smoothing has no performance issues compared to full smoothing. Hence, we use only aggregated square and triangle smoothing, in order to boost computational time. Smoothing with the FACE-algorithm is carried out for irregular and fragmentary data with the `face.sparse` function, which is implemented in the R-package `face` (Xiao et al., 2018). In case of regular data, the function `fpca.face` from the `refund` package (Goldsmith et al., 2022) is used.

Results of Simulation B

Table 8 shows the average Wasserstein- and Frobenius-distances for the three data types, obtained with the three methods. For every data type the average Wasserstein- and Frobenius-distance is smaller for FACE than for square and triangle smoothing. Much like in Simulation A, the Wasserstein-distances obtained with smoothing irregular data are much higher in comparison to regular data. Consequently, a separate analysis is carried out.

Table 8: Average Wasserstein- and Frobenius-distance

Type	Method	Wasserstein	Frobenius
Fragment	FACE	NaN	0.5552
	Square	NaN	0.6499
	Triangle	NaN	0.6260
Irregular	FACE	342.36	0.5482
	Square	483.58	0.5990
	Triangle	566.66	0.6370
Regular	FACE	16.04	0.6064
	Square	17.71	0.6104
	Triangle	19.66	0.6483

In the following, we will assess in which settings the FACE-algorithm prevails against square and triangle smoothing and vice versa. Therefore, we again consider the percentage differences of the

average distances between FACE and the respective alternative.
Similar to Simulation A, we start with the main features of the generated functions, such as grid length, number of curves, and SNR, before grouping the results by the penalty-order that was used in the estimation step.
In Simulation A we observed that grouping the results by grid length and SNR yields quite different outcomes. Hence, we begin with examining the percentage differences of the average Wasserstein- and Frobenius-distances between FACE and square smoothing or FACE and triangle smoothing respectively, in Table 9. The values in Table 9 show by how much percent the average distance obtained with FACE is smaller than the average distance obtained with either square or triangle smoothing. A negative value indicates that FACE is outperformed.

Table 9: Percentage difference of average Wasserstein- and Frobenius-distances between FACE and square smoothing / triangle smoothing, grouped by SNR and grid length

			Wasserstein		Frobenius	
SNR	Grid Length	Type	FACE - Square	FACE - Triang.	FACE - Square	FACE - Triang.
3	20	Fragment	NaN	NaN	-11.99	-8.90
		Irregular	13.74	28.97	5.10	3.93
		Regular	-1.36	41.16	-3.38	9.64
	35	Fragment	NaN	NaN	52.01	10.71
		Irregular	39.91	77.36	7.46	24.41
		Regular	-9.67	51.85	-5.32	9.79
	70	Fragment	NaN	NaN	8.68	25.89
		Irregular	58.17	64.09	14.59	20.71
		Regular	22.73	-6.23	10.34	4.38
8	20	Fragment	NaN	NaN	13.39	17.19
		Irregular	47.18	76.64	6.87	17.14
		Regular	10.78	3.21	0.37	-4.02
	35	Fragment	NaN	NaN	23.55	10.77
		Irregular	6.58	48.21	9.12	26.69
		Regular	26.05	20.06	3.53	11.00
	70	Fragment	NaN	NaN	26.65	28.04
		Irregular	54.73	81.00	13.06	5.49
		Regular	2.68	40.21	-1.11	11.36

Note: percentage difference $= \frac{\text{avg. distance FACE} - \text{avg. distance triangle or square}}{\text{avg. distance FACE}} \times (-100)$

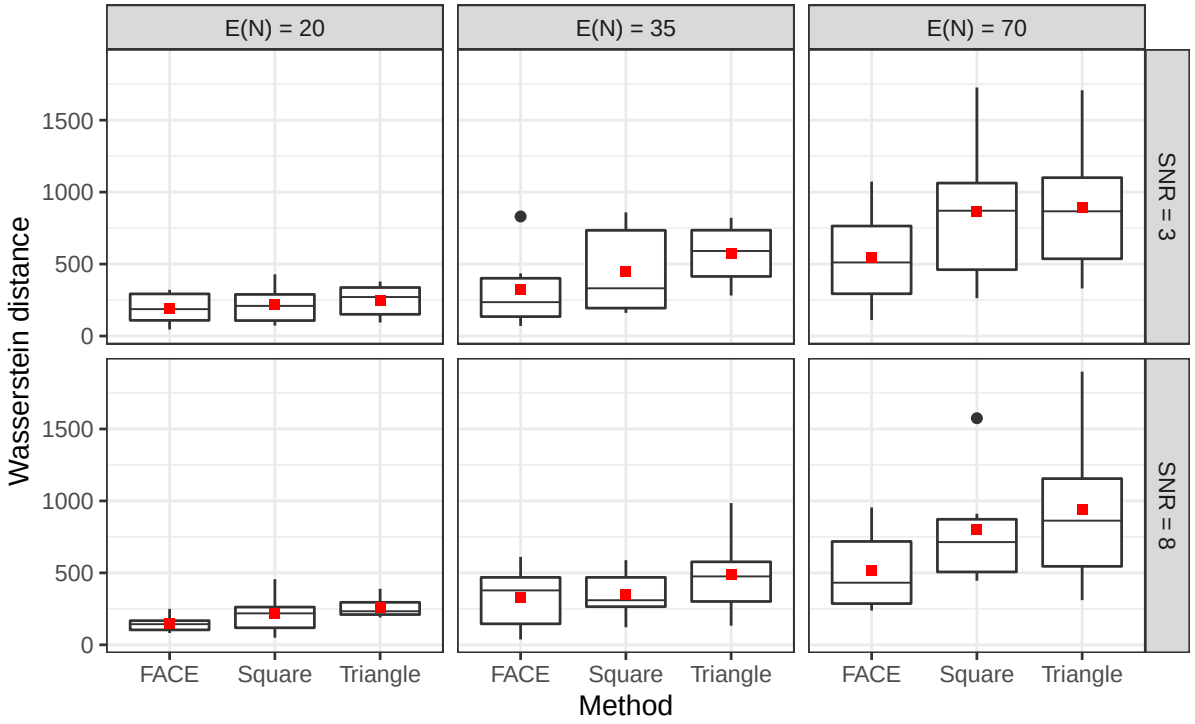
At first glance, the values in Table 9 seem relatively high. For example, FACE perform on average 81% better than triangle smoothing of irregular data with low noise, recorded on a grid of length 70.
In general, FACE performs better than both triangle and square smoothing for most settings. This is especially apparent for irregular data, where we can see FACE clearly prevailing against triangle smoothing in every setting. For data with low noise, i.e. SNR = 8, FACE outperforms square and triangle smoothing in every setting according to the Wasserstein-distance. Yet, the Frobenius-distance implies that triangle smoothing on regular data with grid length 20 and square smoothing on regular data with grid length 70 performed on average better than FACE. In the noisy case we see some more settings where FACE was outperformed. Consider e.g. noisy (SNR = 3) regular data, recorded on 35 and 20 grid points. Both the Wasserstein- and Frobenius-distance indicate that square smoothing performs on average better than FACE. Furthermore, according to the Wasserstein-distance, FACE performs on average 6.23% worse

than triangle smoothing for regular data with SNR = 3 recorded on 70 grid points, although the corresponding Frobenius-distance indicates that FACE outperforms triangle smoothing.
In the case of fragmentary data, FACE performs better than square and triangle smoothing for every grid length where SNR = 8 and the Frobenius-distance even increases with increasing grid length. Yet, for noisy fragmentary data with $E(N) = 20$, square smoothing outperforms FACE on average by 11.99% and triangle smoothing outperforms FACE on average by 8.90%. Considering the two larger grid lengths, FACE performs best for noisy fragmentary data.

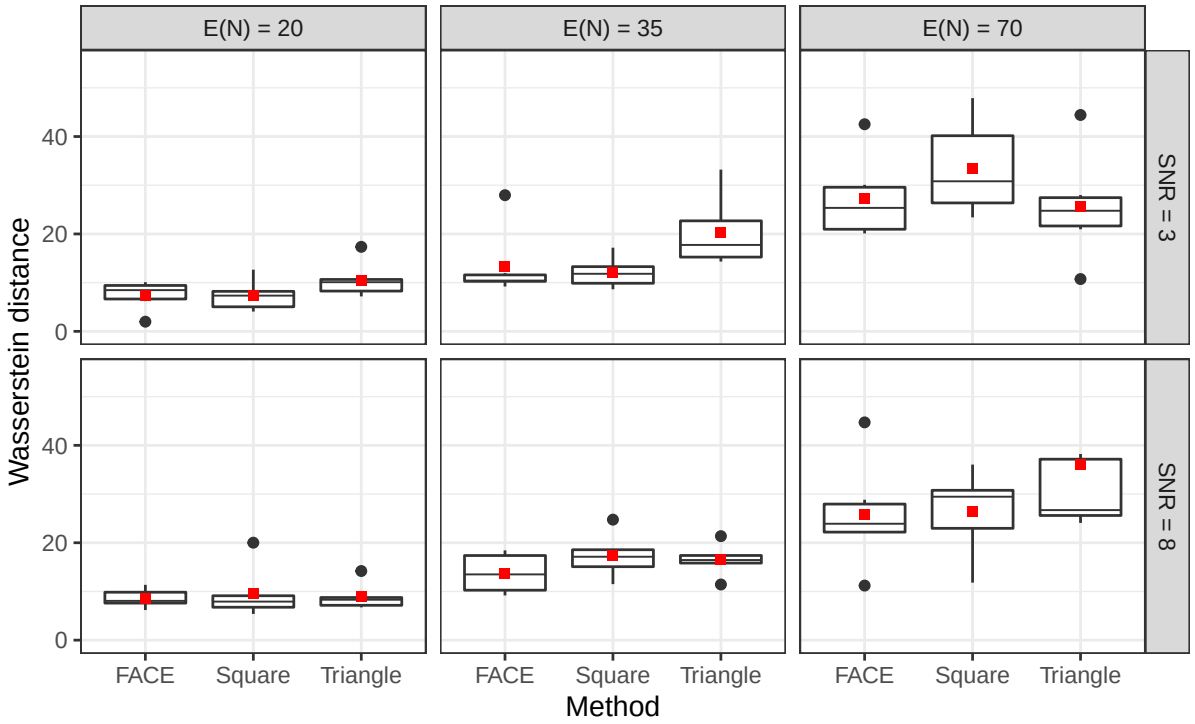
We display the Wasserstein-distances in Figure 4, where Figure 4a) shows the boxplots for irregular data and Figure 4b) shows the boxplots for regular data. The red dots depict the corresponding mean values from Table 9. In the first panel of Figure 4b), we can see that the box, as well as the median which corresponds to square smoothing, is slightly lower than the other two boxes and medians. This reflects the negative value in Table 9.
In the second panel, we observe an outliner of the boxplot corresponding to FACE, that distorts the mean value. The median of FACE is a little bit below the median of square smoothing and the box corresponding to FACE is narrower than the box corresponding to square smoothing, hence the value in Table 9 must be treated with caution. The third panel in Figure 4b) displays the boxplots for noisy regular data, measured on 70 grid points. The box corresponding for triangle smoothing is slightly lower and slimmer than the box corresponding to FACE. Moreover the median of the Wasserstein-distance obtained with triangle smoothing is lower than the corresponding median of FACE.

The boxplots in Figure 5 display the distribution of the Frobenius-distances obtained with each method for different SNR values and grid lengths for fragmentary data in 5a) as well as for regular data in 5b). The red dots are again the corresponding mean values shown in Table 9. According to the mean values, square and triangle smoothing perform better than FACE for fragmentary data, measured on roughly 20 grid points. This case is depicted as the first panel in 5a). For FACE, we see an outliner, which distorts the mean value. Additionally, the boxplot corresponding to FACE as well as the median, are slightly lower than the boxplot and median of square and triangle smoothing. Thus, this result from Table 9 should also be treated with caution. For regular data, measured on 20 grid points, the results from Table 9 are reflected in the left panels of Figure 5b), where the boxplots and medians corresponding to square smoothing are lower than the boxplots and medians corresponding to the other two smoothing methods.

A crucial ingredient in the estimation process is the order of the penalization. Table 10 shows the percentage differences of the Wasserstein- and Frobenius-distance between FACE and square smoothing as well as FACE and triangle smoothing, grouped by penalty order and grid length. According to the Wasserstein-distance, FACE with a first-order penalty performs better than square and triangle smoothing with a first-order penalty in every setting. Except for square smoothing of regular data with $E(N) = 70$, FACE has the best performance, according to the Frobenius-distance, in the case of first-order penalty.
When smoothing with a second-order penalty, we can see that FACE is often outperformed in the regular case. Moreover, the Frobenius-distance indicates that square and triangle smoothing perform better than FACE for fragmentary data with $E(N) = 20$.
These findings are reflected in the subfigures of Figure 6, which display the percentage of cases where the respective smoothing method outperformed the other, grouped by grid length and penalty order, according to either the Wasserstein- or Frobenius-distance.

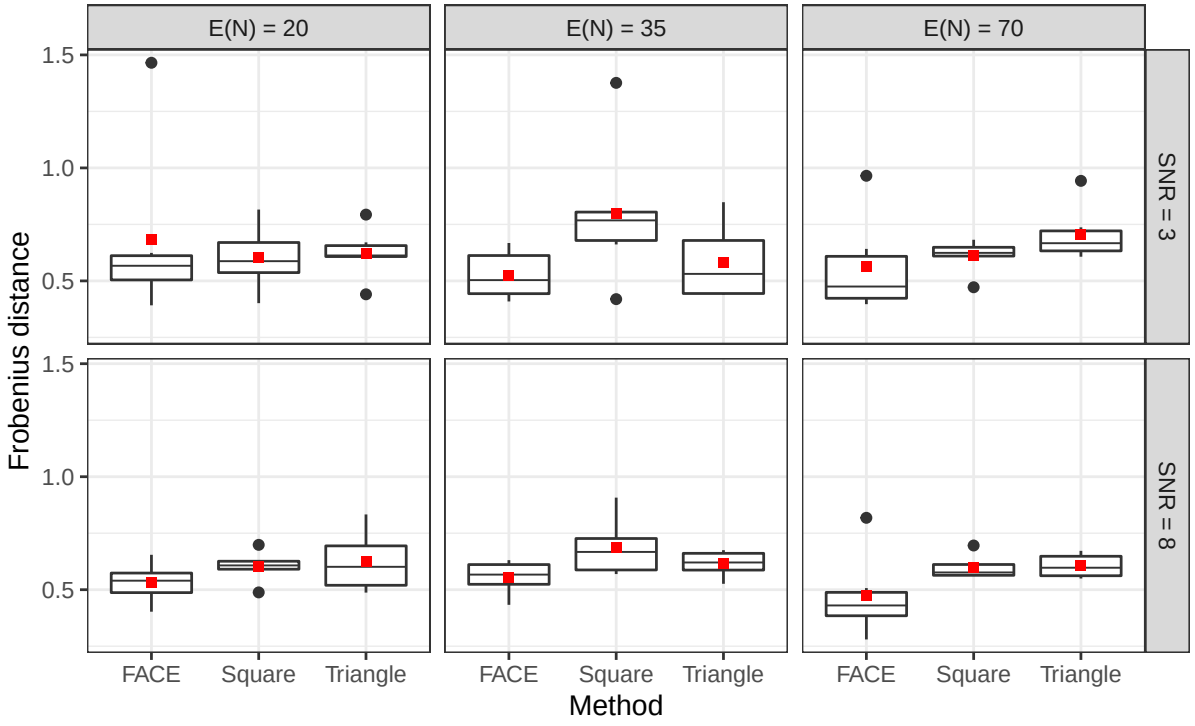


(a) Irregular Data

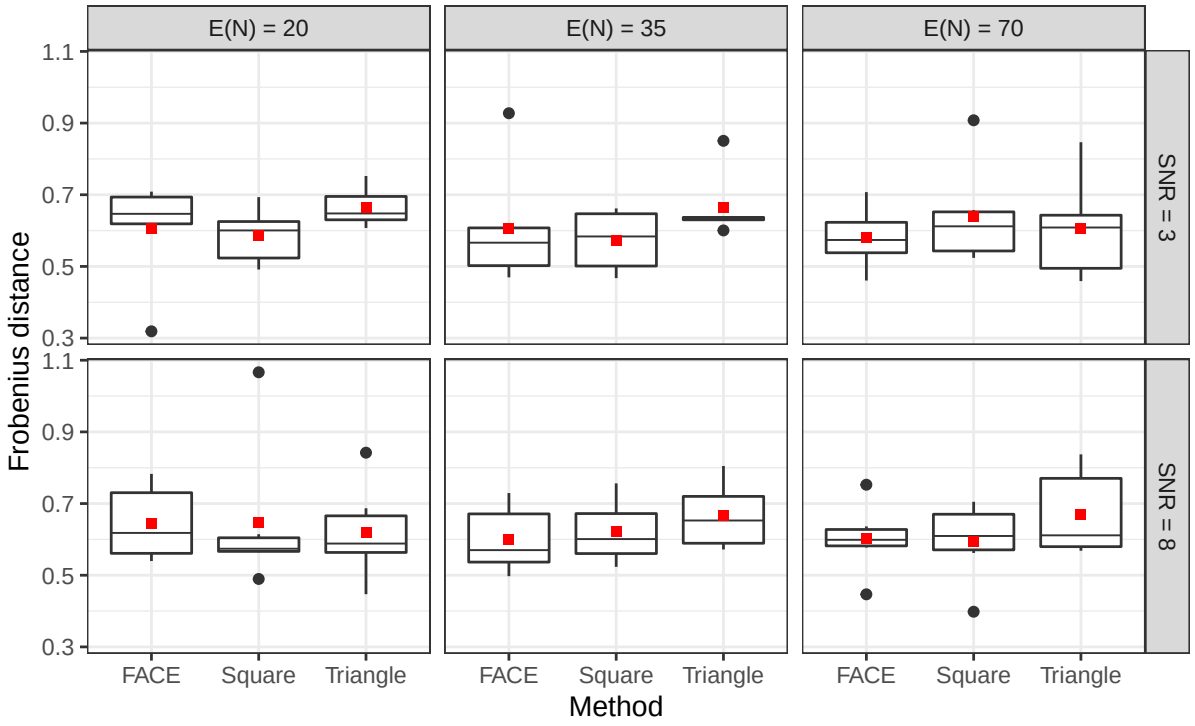


(b) Regular Data

Figure 4: Boxplots of Wasserstein-distances for irregular and regular data, grouped by grid length and SNR



(a) Fragmentary Data



(b) Regular Data

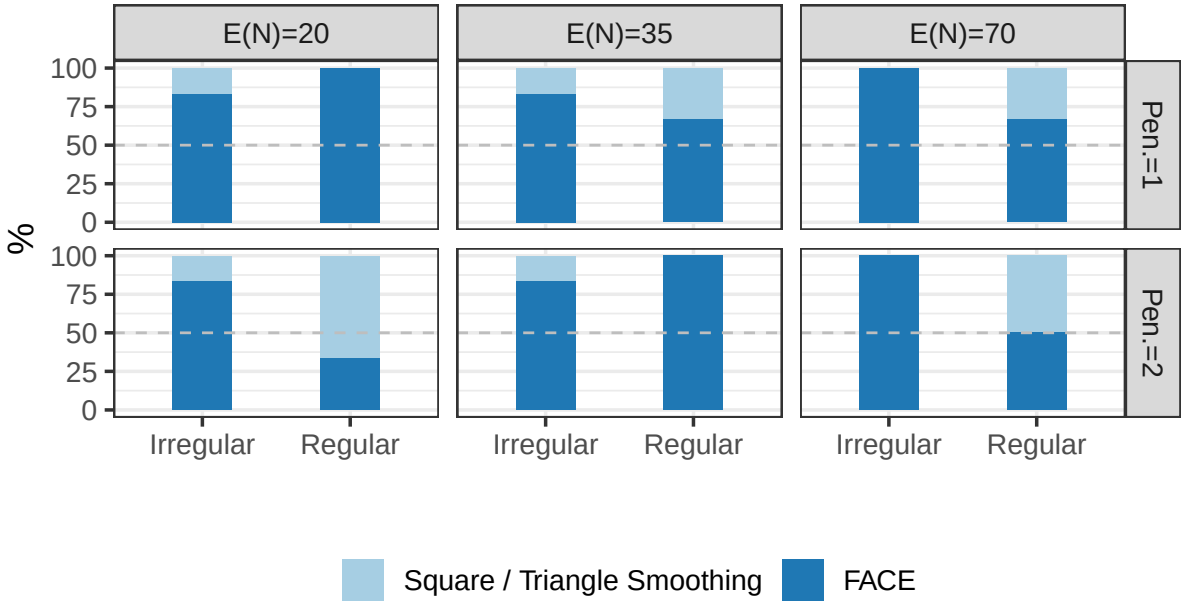
Figure 5: Boxplots of Frobenius-distances for fragmentary and regular data, grouped by grid length and SNR

Table 10: Percentage difference of average Wasserstein- and Frobenius-distances between FACE and square smoothing / triangle smoothing, grouped by penalty order and grid length

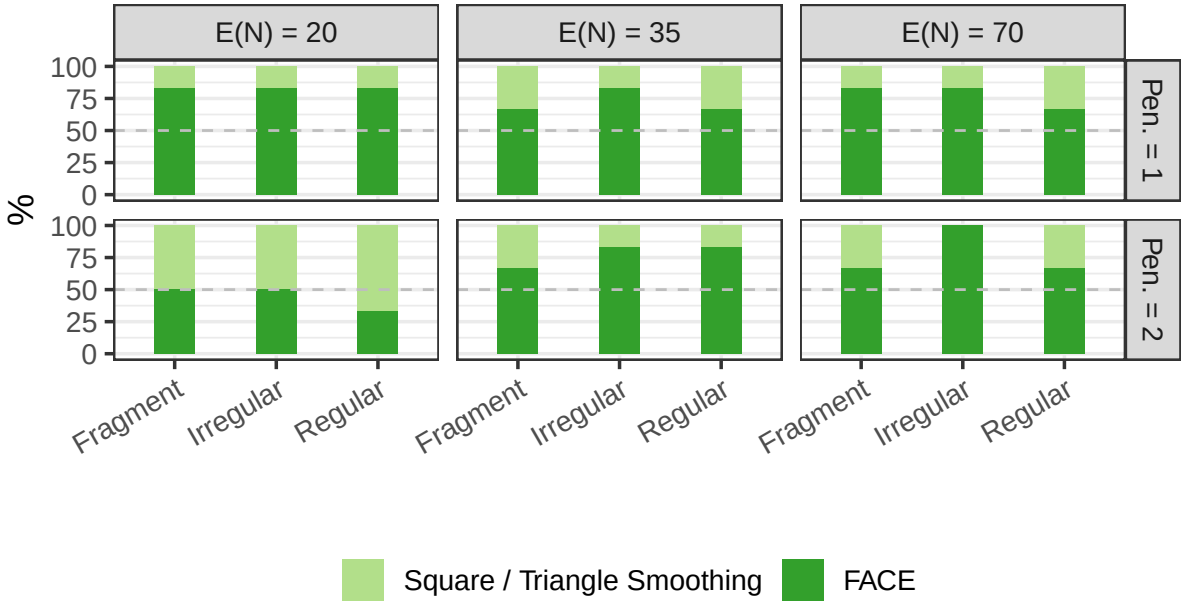
			Wasserstein		Frobenius	
Pen. Order	Grid Length	Type	FACE - Square	FACE - Triang.	FACE - Square	FACE - Triang.
2	20	Fragment	NaN	NaN	-11.99	-15.80
		Irregular	33.95	52.22	0.43	-1.04
		Regular	-12.50	14.86	-9.69	-5.28
	35	Fragment	NaN	NaN	50.57	12.10
		Irregular	33.53	60.54	13.60	18.18
		Regular	9.41	56.95	-3.03	17.82
	70	Fragment	NaN	NaN	2.70	3.33
		Irregular	65.03	78.55	18.70	18.45
		Regular	16.82	-11.39	15.03	0.60
1	20	Fragment	NaN	NaN	14.23	27.42
		Irregular	22.13	46.96	12.13	23.19
		Regular	23.75	27.18	6.84	10.53
	35	Fragment	NaN	NaN	24.55	9.42
		Irregular	13.52	64.55	3.21	32.57
		Regular	7.53	15.90	1.19	3.01
	70	Fragment	NaN	NaN	36.83	59.94
		Irregular	48.84	66.74	9.33	7.79
		Regular	7.74	54.43	-6.74	15.78

Note: percentage difference $= \frac{\text{avg. distance FACE} - \text{avg. distance triangle or square}}{\text{avg. distance FACE}} \times (-100)$

Additionally, the percentage difference grouped by penalty order and SNR, are displayed in Table 11. What clearly stands out are the high values for irregular data compared to regular data. Consider the second row, which indicates that the second-order penalty FACE-algorithm performed on average 83.69% better than square smoothing with a second-order penalty and 90.01% better than triangle smoothing with a second-order penalty for irregular data with SNR = 3, according to the Wasserstein-distance. Yet, for irregular data with SNR = 8, FACE is on average "only" 17.47% better than square smoothing and 47.64% better than triangle smoothing. While the percentage differences for irregular data decrease with increasing SNR in the case of second-order penalty, they increase with increasing SNR for a first-order penalty.



(a) Wasserstein distance



(b) Frobenius distance

Figure 6: Percentage of cases where the respective smoothing method outperformed the other, grouped by grid length and penalty order

Table 11: Percentage difference of average Wasserstein- and Frobenius-distances between FACE and square smoothing / triangle smoothing, grouped by penalty order and SNR

			Wasserstein		Frobenius	
Pen. Order	SNR	Type	FACE - Square	FACE - Triang.	FACE - Square	FACE - Triang.
2	3	Fragment	NaN	NaN	13.28	-5.86
		Irregular	83.69	90.01	17.81	17.23
		Regular	13.66	9.00	4.63	8.10
	8	Fragment	NaN	NaN	8.48	3.51
		Irregular	17.47	47.64	3.62	5.79
		Regular	7.12	10.87	-2.99	0.57
1	3	Fragment	NaN	NaN	13.92	24.52
		Irregular	12.09	38.29	0.85	14.94
		Regular	5.90	26.48	-3.76	7.88
	8	Fragment	NaN	NaN	36.96	36.83
		Irregular	58.96	92.51	16.31	28.03
		Regular	15.38	48.94	5.04	11.47

Note: percentage difference $= \frac{\text{avg. distance FACE} - \text{avg. distance triangle or square}}{\text{avg. distance FACE}} \times (-100)$

4.3. Summary and Discussion of Results

Before we begin with a summary and discussion of the simulation results, a remark about the distance measures is required. For most cases, the percentage differences of the Frobenius-distances between square smoothing and triangle smoothing are smaller than the corresponding percentage differences of the Wasserstein-distances. Sometimes even the conclusion as to whether one method works better than the other can deviate, as shown in Table 4 of Simulation A. As a consequence, the results must be interpreted with respect to the distance measure.

The main conclusion we obtain from Simulation A, is that triangle smoothing and square smoothing produce estimators with quite similar accuracy. Both methods perform a lot better for regular data than for irregular data, which is not surprising since regular data has been measured on the same equidistant time points for every curve, in contrast to irregular data where the individual grid differs from curve to curve. Square smoothing performs slightly better than triangle smoothing for irregular data, while triangle smoothing outperforms square smoothing on average by 5.17% for regular data, according to the Wasserstein-distance. Furthermore, the average Frobenius-distances obtained with square and triangle smoothing of fragmentary data only differ 0.1% from each other.
Grouping the results further by noise-level yields larger differences of the Wasserstein-distances, especially for a grid length $E(N) = 70$, compared to the whole set of results that is only grouped by data type. However, a clear pattern indicating whether a smoothing method performs better than the other with increasing grid length or SNR is not obvious.
For almost every setting, smoothing data with a second-order penalty results in a percentage difference of less than 1% between the Wasserstein- and Frobenius-distances, obtained with the respective method. Regular triangle-optimal data forms the noticeable exception, where triangle smoothing performs on average 9.59% better than square smoothing. This exception is no surprise, since triangle smoothing is not penalized in the triangle-optimal setting, so it has an "advantage" against square smoothing. Considering a first-order penalty, the percentage differences between the distance measures increase. These findings are expected, since a first-order

penalty results in a penalty null space spanned by constant functions, resulting in square and triangle smoothing being penalized in both the square- and triangle-optimal setting. The increase in the percentage distances becomes especially visible for data sampled on large grids, where square smoothing clearly prevails against triangle smoothing for irregular data and triangle smoothing prevails against square smoothing for regular data.

Simulation B compared square and triangle smoothing with FACE using data generated with the squared exponential function. We observe that in nearly every setting FACE outperforms square and triangle smoothing. Especially for irregular data, we can see FACE clearly prevailing against square and triangle smoothing. Yet, for regular data with high noise and small grids of length 20 and 35, square smoothing seems to work better than FACE. At first glance, this exception seems surprising, since FACE estimates the measurement-error-variance as opposed to square smoothing, hence we would expect FACE to provide a more accurate estimation of the covariance surface. Conversely, we compare the true covariance matrix with the estimated covariance matrix, which are both evaluated on all grid points. Because the true covariance matrix contains the evaluations of the covariance function without the additional nugget term, estimating the measurement-error-variance is redundant.
In the case of a second-order penalty, we observe the absolute percentage differences of the distance measures decreasing with increasing SNR. However, in the case of a first-order penalty the absolute percentage differences of the distance measures increase with increasing SNR. This is an astonishing contrast, as the penalty order changes how well FACE performs in comparison to the other methods for different noise levels.
A possible explanation for these high values could be, that FACE utilizes only the row penalty of bivariate P-Splines, while square and triangle smoothing use both the column and the row penalty. Although the covariance's symmetry implies that the column penalty is the same as the row penalty (Xiao et al., 2018, p. 213), FACE is presumably less penalized than square and triangle smoothing. This might also relate to the percentage differences of the distance measures decreasing with increasing SNR for a second-order penalty estimation.

In summary, we conclude that triangle and square smoothing yield similar results, thus triangle smoothing is an appropriate alternative to square smoothing with a computational benefit, especially for regular data. However, both square and triangle smoothing do not provide as accurate estimates as FACE does.

5. Conclusion

In this thesis we explored the application of penalized tensor product splines to the covariance estimation of functional data. Analogously to the approximation of univariate functions, the general approach of covariance smoothing is to set up a regression model, where the response vector consists of all empirical estimates for the covariance. Because the covariance is smoothed for the squared set of all empirical estimates, this method is known as square smoothing. Yet, the symmetry of the covariance deems smoothing the entire squared set of all empirical estimates superfluous, as the response vector will then include double values. By smoothing only half of the covariance and mirroring it on the diagonal, triangle smoothing reduces the dimension of the response vector and provides a computational alternative to square smoothing. Hence, we expect triangle and square smoothing to produce similar estimations. This expectation was empirically studied in Simulation A and indeed the differences between both methods are in general quite small, with triangle smoothing performing slightly better than square smoothing for regular data and square smoothing slightly outperforming triangle smoothing for irregular data. Considering fragmentary data the differences between the methods are infinitesimally small. However, when a first-order penalty is applied rather than a second-order penalty, the differences increase and become more significant. Conclusively, triangle smoothing can be viewed as a computationally beneficial alternative to square smoothing with similar estimation accuracy. In practice, the measurements are often polluted by errors, which are reflected on the variance at the corresponding measurement value. In contrast to square and triangle smoothing which exclude values on the diagonal in the response vector, FACE provides a two-stage algorithm to estimate the covariance function as well as the measurement-error-variance. Moreover, FACE uses a special case of the bivariate P-Spline-penalty and provides its own algorithm to choose the smoothing parameter. Comparing the performance of FACE with square and triangle smoothing in Simulation B, shows FACE clearly outperforming the other two methods for most settings.

A. Directory Structure of the GitLab Repository

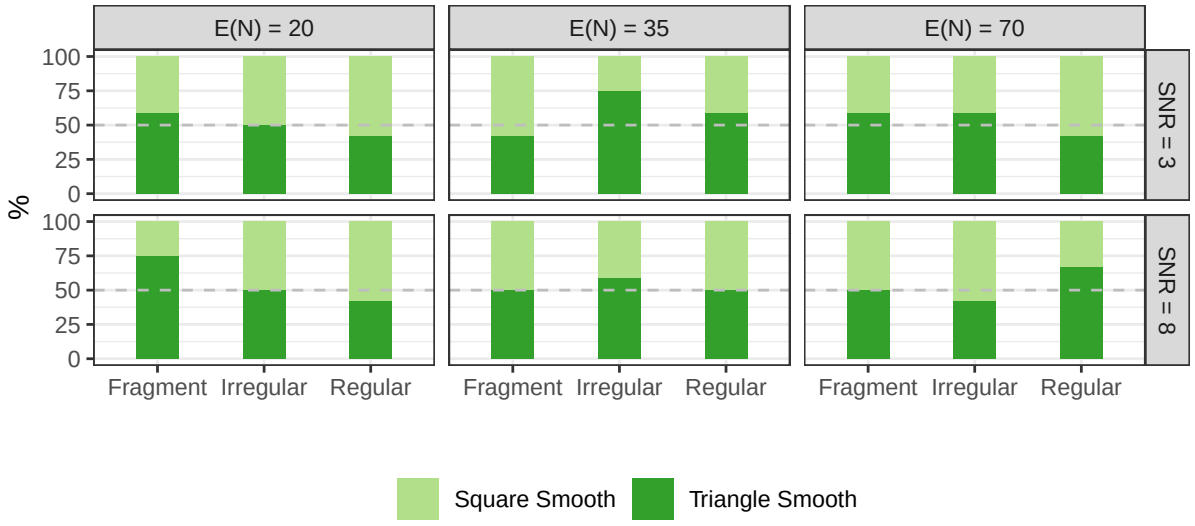
The submission of this bachelor thesis includes all codes, for generating and analyzing the simulation results, in the corresponding GitLab Repository. In the following a brief overview over the repository and its directory structure is given. The template for this thesis was provided by Moritz Herrmann.

`thesis`: includes chapter-wise Rmarkdown files. `00_thesis.Rmd` knits all chapters and produces the output.

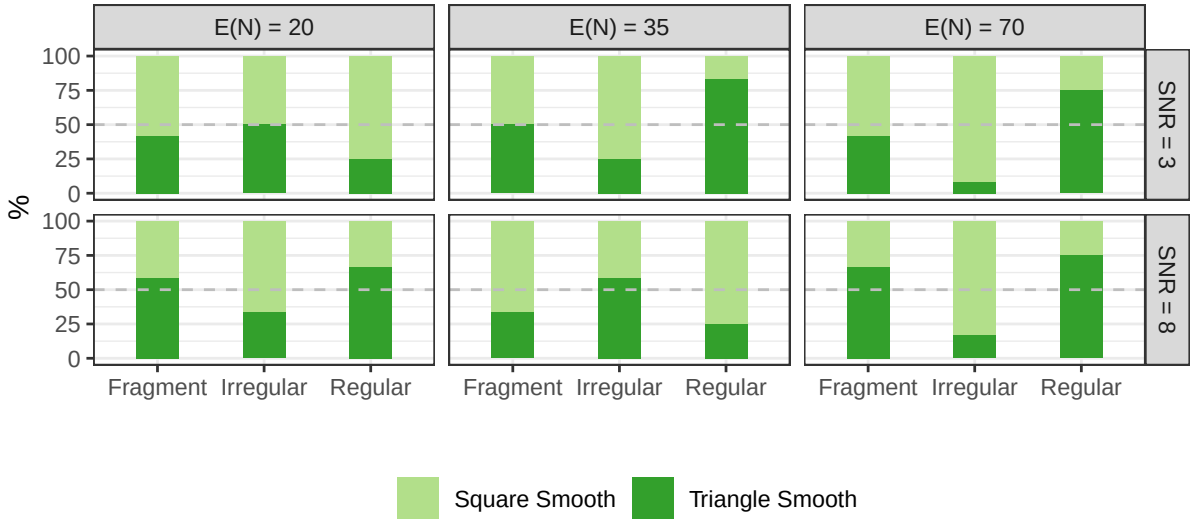
`code`: contains functions that are used by both parts of the simulation

- Simulation A:
  - Run `main_a.R` to start the simulation. Input data is then loaded and `simulator_a` is called on `input_data`
    - `simulator_a` generates a `tfd`-object using data_generator and estimates the covariance with the `smooth_covariance` function
    - `simulator_b` then computes the Wasserstein- and Frobenius-distances and stores the values as new columns of the input data
  - Results stored in `Simulation_Results` folder
- Simulation B:
  - Run `main_b.R` to start the simulation. Input data is then loaded and `simulator_a` is called on `input_data`
    - `simulator_b` generates a tfd-object using data_generator and estimates the covariance with the face_smooth_covariance function
    - `simulator_b` then computes the Wasserstein- and Frobenius-distances and stores the values as new columns of the input data
  - Results stored in `Simulation_Results` folder
- Evaluation:
  - `tables_a.R`: creates dataframes that are sourced and used in `04_experiments_a.Rmd` for tables (Chapter 4)
  - `figures_a.R`: creates dataframes that are sourced and used in `04_experiments_a.Rmd` for figures (Chapter 4)
  - `tables_b.R`: creates dataframes that are sourced and used in `04_experiments_b.Rmd` for tables (Chapter 4)
  - `figures_b.R`: creates dataframes that are sourced and used in `04_experiments_b.Rmd` for figures (Chapter 4)
- `Simulation_Results`: contains results in form of csv files
- temp: small pilot study to determine suitable values for the SNR
- Old: contains old files

B. Additional Plots for Simulation A

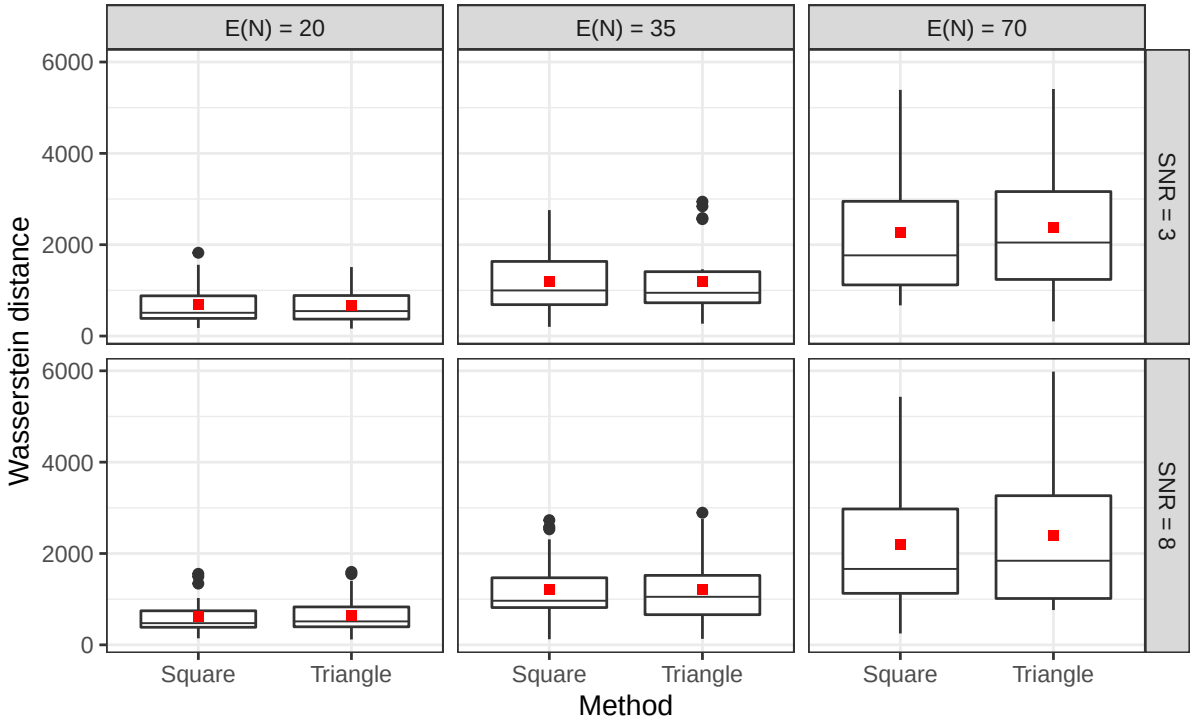


(a) Second-Order Penalty

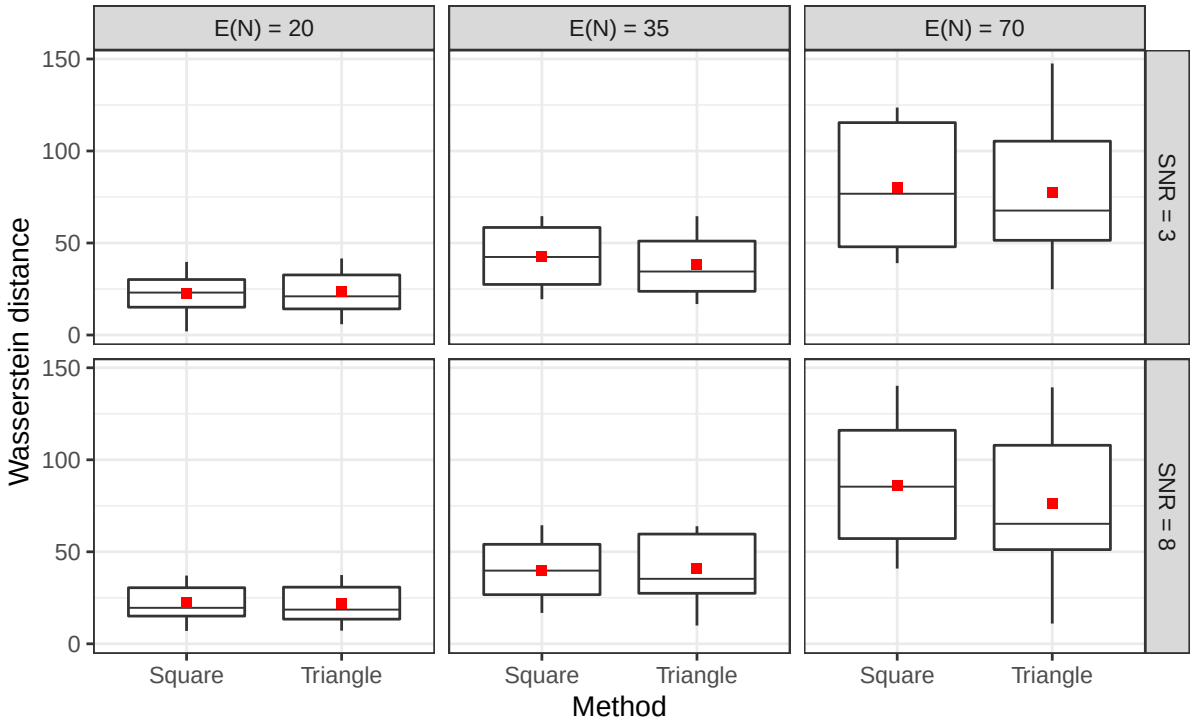


(b) First-Order Penalty

Figure 7: Percentage of cases where the respective smoothing technique outperformed the other in terms of the Frobenius-distance. Grouped by grid length and SNR.

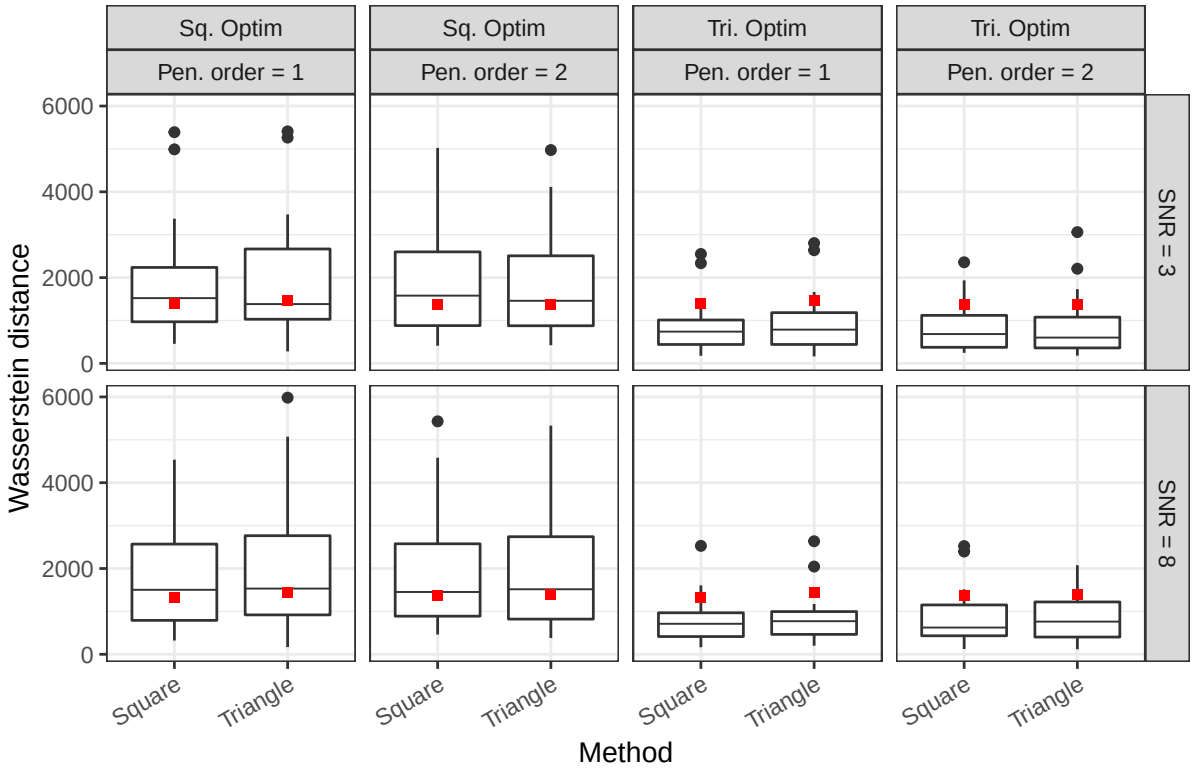


(a) Irregular Data

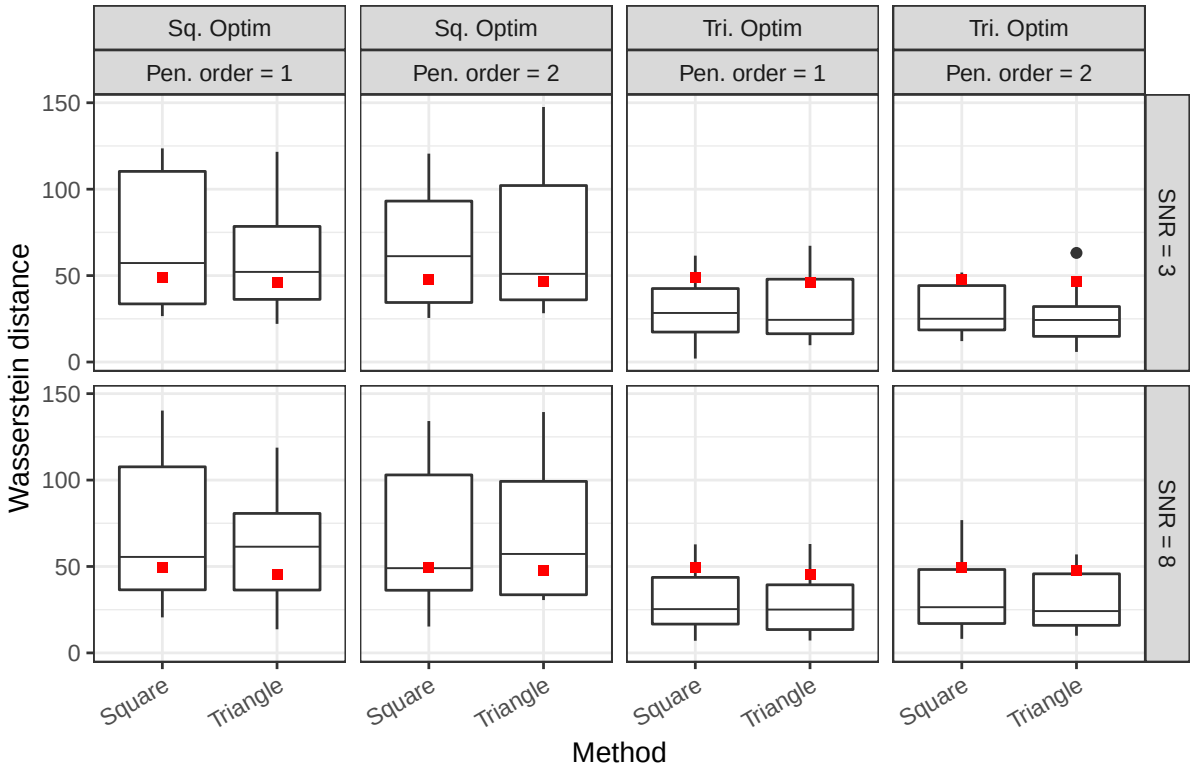


(b) Regular Data

Figure 8: Boxplots of Wasserstein-distances for regular and irregular data, grouped by grid length and SNR



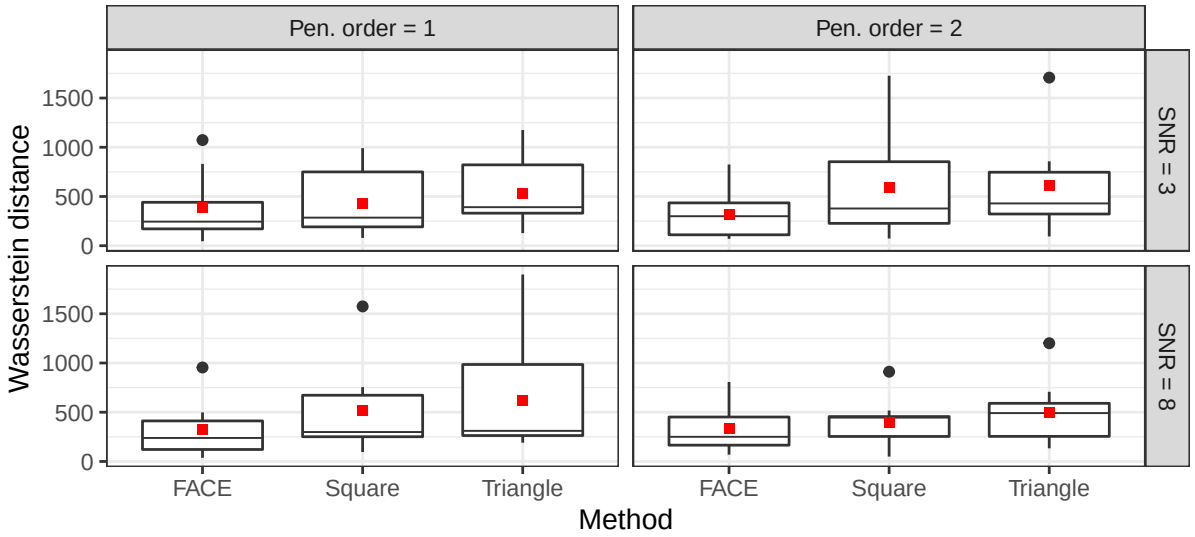
(a) Irregular Data



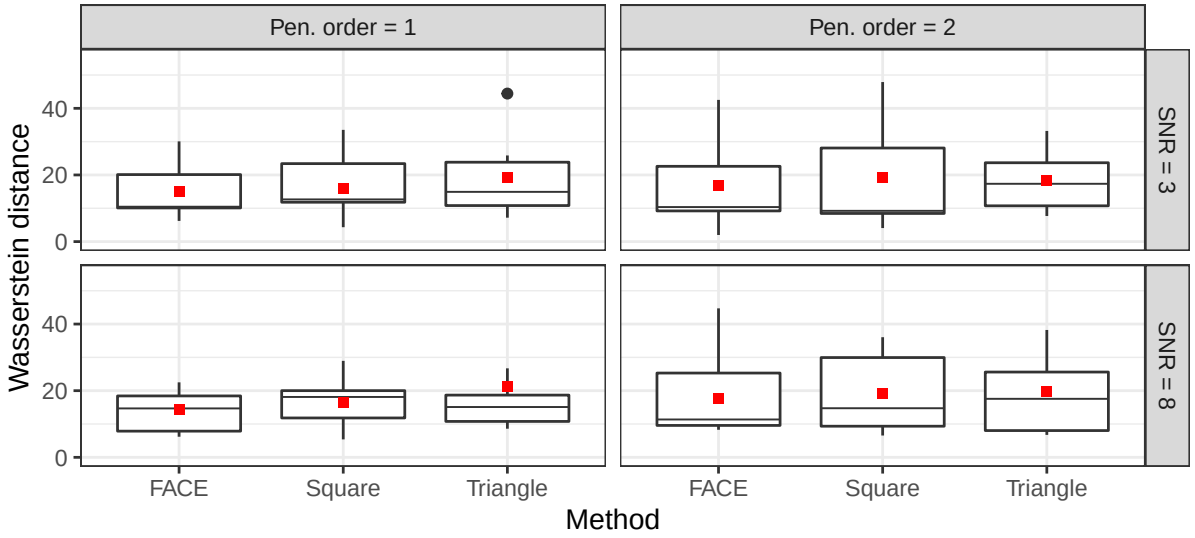
(b) Regular Data

Figure 9: Boxplots of Wasserstein-distances for regular and irregular data, grouped by grid length and SNR

C. Additional Plots for Simulation B



(a) Irregular Data



(b) Regular Data

Figure 10: Boxplots of Wasserstein-distances for regular and irregular data, grouped by penalty order and SNR. Red dots represent the mean value of corresponding method.

References

Jona Cederbaum, Fabian Scheipl, and Sonja Greven. Fast symmetric additive covariance smoothing. *Computational Statistics and Data Analysis*, 120:25–41, 2018. doi: 10.1016/j.csda.2017.11.002.

Aurore Delaigle, Peter Hall, Wei Huang, and Alois Kneip. Estimating the covariance of fragmented and other related types of functional data. *Journal of the American Statistical Association*, 116(535):1383–1401, 2021. doi: 10.1080/01621459.2020.1723597.

Paul H. C. Eilers and Brian D. Marx. Flexible smoothing with *b*-splines and penalties. *Statistical Science*, 11(2):89–102, 1996. doi: 10.1214/ss/1038425655. URL http://www.jstor.org/stable/2246049.

Paul H.C. Eilers and Brian D. Marx. Multivariate calibration with temperature interaction using two-dimensional penalized signal regression. *Chemometrics and Intelligent Laboratory Systems*, 66(2):159–174, 2003. doi: https://doi.org/10.1016/S0169-7439(03)00029-7.

Ludwig Fahmeir, Thomas Kneib, Stefan Lang, and Brian Mark. *Regression: Models, Methods and Applications*. Springer Berlin Heidelberg, 1. edition, 2013. ISBN 978-3-642-34333-9. doi: https://doi.org/10.1007/978-3-642-34333-9.

Jianquing Fan and Irene Gijbels. *Local Polynomial Modelling and Its Applications*. Chapman and Hall/CRC, 1. edition, 1996. ISBN 9780412983214.

Jeff Goldsmith, Fabian Scheipl, Lei Huang, Julia Wrobel, Chongzhi Di, Jonathan Gellar, Jaroslaw Harezlak, Mathew W. McLean, Bruce Swihart, Luo Xiao, Ciprian Crainiceanu, and Philip T. Reiss. *refund: Regression with Functional Data*, 2022. URL https://CRAN.R-project.org/package=refund. R package version 0.1-28.

Zhenhua Lin and Jane-Ling Wang. Mean and covariance estimation for functional snippets. *Journal of the American Statistical Association*, 117(537):348–360, 2022. doi: 10.1080/01621459.2020.1777138.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2022. URL https://www.R-project.org/.

J. O. Ramsay and B. W. Silverman. *Functional Data Analysis*. Springer New York, NY, 2. edition, 2005. doi: https://doi.org/10.1007/b98888.

Philip Reiss and Meng Xu. Tensor product splines and functional principal components. *Journal of Statistical Planning and Inference*, 208:1–12, 10 2019. doi: 10.1016/j.jspi.2019.10.006.

Philip Reiss, Lei Huang, Huaihou Chen, and Stan Colcombe. Varying-smoother models for functional responses, 2014. URL https://arxiv.org/abs/1412.0778.

Fabian Scheipl, Jeff Goldsmith, and Julia Wrobel. *tidyfun: Tools for Tidy Functional Data*, 2022. https://github.com/tidyfun/tidyfun, https://tidyfun.github.io/tidyfun/.

Jane-Ling Wang, Jeng-Min Chiou, and Hans-Georg Müller. Functional data analysis. *Annual Review of Statistics and Its Application*, 3(1):257–295, 2016. doi: 10.1146/annurev-statistics-041715-033624.

Simon N. Wood. *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC, 1. edition, 2006. ISBN 9780429093159. doi: https://doi.org/10.1201/9781420010404.

Simon N. Wood. *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC, 2. edition, 2017. ISBN 9781315370279. doi: https://doi.org/10.1201/9781315370279.

Simon N. Wood, Fabian Scheipl, and Julian J. Faraway. Straightforward intermediate rank tensor product smoothing in mixed models. *Statistics and Computing*, 23(3):341–360, 2013. doi: https://doi.org/10.1007/s11222-012-9314-z.

Luo Xiao, Cai Li, William Checkley, and Ciprian Crainiceanu. Fast covariance estimation for sparse functional data. *Statistics and Computing*, 28(3):511–522, 2018. doi: 10.1007/s11222-017-9744-8.

Fang Yao, Hans-Georg Müller, and Jane-Ling Wang. Functional data analysis for sparse longitudinal data. *Journal of the American Statistical Association*, 100(470):577–590, 2005. doi: 10.1198/016214504000001745.

Declaration of authorship

I hereby declare that the report submitted is my own unaided work. All direct or indirect sources used are acknowledged as references. I am aware that the Thesis in digital form can be examined for the use of unauthorized aid and in order to determine whether the report as a whole or parts incorporated in it may be deemed as plagiarism. For the comparison of my work with existing sources I agree that it shall be entered in a database where it shall also remain after examination, to enable comparison with future Theses submitted. Further rights of reproduction and usage, however, are not granted here. This paper was not previously presented to another examination board and has not been published.

Munich, 22 March 2023 Anna Nazarova