HAL
open science

Gaussian process metamodeling of functional-input code for coastal flood hazard assessment

José Daniel Betancourt, François Bachoc, Thierry Klein, Déborah Idier, Rodrigo Pedreros, Jeremy Rohmer

To cite this version:

José Daniel Betancourt, François Bachoc, Thierry Klein, Déborah Idier, Rodrigo Pedreros, et al.. Gaussian process metamodeling of functional-input code for coastal flood hazard assessment. 2019. hal-01998724v1

HAL Id: hal-01998724

https://hal.science/hal-01998724v1

Preprint submitted on 29 Jan 2019 (v1), last revised 25 Nov 2019 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Gaussian process metamodeling of functional-input code for coastal flood hazard assessment

José Betancourt[a,b], François Bachoc[a], Thierry Klein[a,b,*], Deborah Idier[c], Rodrigo Pedreros[c], Jérémy Rohmer[c]

[a] *Institut de Mathématiques de Toulouse, UMR 5219, Université de Toulouse, CNRS, UPS IMT, 31062 Toulouse Cedex 9, France*
[b] *ENAC - Ecole Nationale de l'Aviation Civile, Université de Toulouse, France*
[c] *BRGM, 3, av. Claude Guillemin, BP 36009, 45060 Orleans Cedex 2, France*

Abstract

In this article we investigate the construction of metamodels when the numerical code has functional inputs. We compare diverse Gaussian process metamodeling approaches that take into account the functional structure of the data. We discuss two methods to tune the dimension of the projection when functional inputs are decomposed on a functional basis: (i) based on the error of the projection; (ii) based on the performance of the metamodel. We further propose a methodology that allows to detect the optimal projection of the input function. We apply this methodology to the real case study of coastal flooding in the peninsula of Gâvres. Results show that the approach based on the error of the projection, being the common practice nowadays, may lead to unnecessarily large projection dimensions. In contrast, the approach based on metamodel performance presents the virtue of directly pointing to the final objective of building a fast and accurate metamodel.

Keywords: Computer code experiments, Metamodeling, Functional inputs, Gaussian processes

1. Introduction

The use of computer codes for the study of complex systems is, nowadays, a well extended practice. On the one hand, they offer the possibility of simulating realizations of the system under study at a lower resource expense/risk than if observations were taken from the real system. On the other hand, computer codes provide a solution for cases when the real system is a natural process (i.e., it makes part of the environment) and some conditions of the inputs or the outputs rarely occur. In the coastal flooding domain, for instance, by focusing on flooding on sites never or rarely flooded, it is not possible to obtain a sufficient number of observations from historical registers [1, 2, 3]. In

*Corresponding author
Email addresses: jbetanco@math.univ-toulouse.fr (José Betancourt), francois.bachoc@math.univ-toulouse.fr (François Bachoc), thierry.klein@math.univ-toulouse.fr (Thierry Klein), D.Idier@brgm.fr (Deborah Idier), r.pedreros@brgm.fr (Rodrigo Pedreros), j.rohmer@brgm.fr (Jérémy Rohmer)

those cases, computer codes can be used to produce the required observations to complement historical data. Despite the aforementioned advantages, computer codes for environmental and industrial applications often happen to be too time-consuming for direct application (e.g., for uncertainty quantification or fast prediction within an early warning system) [4, 5]. Despite the steady and continuing growth of computing power and speed, the complexity of these codes seems to keep pace with computing advances [6]. This issue is usually addressed by creating quick-to-evaluate mathematical emulators of those numerical codes, based on limited collection of runs of the original computational model [7, 8, 9]; such emulators are often called *surrogate models* or *metamodels*. In this study, we illustrate the development of a metamodel for the emulation of a complex hydrodynamic code used in the context of early warning for coastal flooding hazards, using a simplfied fast running model for the metamodel developement purpose.

The computer code studied here receives four inputs and delivers a single output, all of them functional (time-dependent). The focus of this article is on the management of functional inputs, therefore we use a simplified scalar representation of the output but keep the full complexity of the inputs into account. Among all the metamodel-based solutions (polynomials, splines, neural networks, etc.), we focus on Gaussian processes [10, 11, 12]. These are one of the most popular metamodeling alternatives, partly due to their ability to provide both an interpolation of the data and an uncertainty quantification in the unexplored regions. Although Gaussian processes for scalar-valued inputs and outputs have been studied for almost 30 years, the functional framework is still a relatively new and much less developed research area [4]. A common approach to deal with functional data is to project the input/output onto a functional basis of lower dimensionality while preserving the main statistical or geometric properties of the variable. The basis functions can be of various forms, such as Legendre polynomials, trigonometric functions, or wavelet bases [13]. In the case of functional inputs, one can either use the coefficients of the decomposition as independent scalar inputs of the metamodel, or implement adapted covariance functions able to measure the similarity among pairs of projections. The first and more common approach was applied for wavelets in [14] and for principal component analysis (PCA) in [5]. A comparison of PCA and partial least squares (PLS) was also carried out by [15]. The second approach was recently adopted by [4], who projected the inputs of the model onto a B-pline basis [16] and used a Gaussian process metamodel with an adapted covariance function depending on the decomposition to fit the output of the code.

A common limitation for most of the studies cited above is that the projection dimension is chosen beforehand. Usually, the new dimension p is chosen so that most information is concentrated in the p first basis functions, e.g. so that the variance in the set of functional inputs is explained at a minimum level of 80%; yet, for forecast purposes, it is the accuracy on the prediction of the output which matters. Therefore, dimension reduction of the inputs should be primarily constrained by metamodel predictability (objective-based dimension reduction). An interesting solution lies in the field of (linear) regression under the name of scalar-on-function regression (see e.g. [17] for a review article). This method deals with cases of high-dimensional regression, where the number of covariates is relatively large compared to the amount of observations. It consists on the use of penalizations to control the number of predictors in the model and

consequently, the overall sparsity of the coefficient matrix. In the context of this article, scalar-on-function techniques could be applied directly to the original input funtion (under a time series representation) or to a functional projection of it, in order to determine which elements of the series or coefficients of the projection to keep in the model. Applications of this approach can be found for instance in [18] using a PCA basis and [19] using a kernel-based principal component basis (KPCA). Despite the advantages that scalar-on-function regression offers in terms of metamodel fitting, developments concerning this method are mainly related to the linear regression framework. The present study extends the state of the art by exploring non-linear regression models (Gaussian processes). Moreover, the methodology proposed here allows to simultaneously set up the projection dimension and diverse other characteristics of the metamodel such as the projection method (PCA, B-splines, wavelets, etc), the type of covariance function or the distance used to measure similarity among observation points. This types of metamodeling choices, herein called *structural parameters* of the metamodel, are often fixed arbitrarily or based on results from previous articles. However, as we prove through the case studies conducted here, the ideal metamodel configuration depends on the particular application. Thus, this kind of setting should be optimized each time a metamodel is to be built in order to get the best results. Although simple, the methodology presented here seems to be an effective tool to perform such an optimization task.

The remainder of this paper is organized as follows. Section 2 describes the motivating case which lies in the domain of coastal flooding. As mentioned above, this case study requires the application of special metamodeling techniques designed for scenarios with functional inputs. The particular methods used here are described in Section 3. Then, Section 4 illustrates the exploration approach that we follow in order to setup structural parameters of the metamodel while setting a good dimension for the inputs. Section 4 also includes an analytic case study that allows to illustrate the functioning of the methodology. In Section 5 we apply the exploration methodology to the coastal flooding case study. The contrast among the solutions of the two case studies evidences that the ideal levels of the structural parameters of a metamodel strongly depends on the application, which justifies the use of our methodology. This and some other conclusions, as well as further research lines are pointed out in Section 6.

2. Motivating case: coastal flooding prediction at Gâvres, France

This study is motivated by the Gâvres coastal flooding case study extracted from the ANR research project RISCOPE [20]. RISCOPE focuses on the development of risk-based methods relying on metamodeling, for forecasting, early warning and prevention of coastal flooding. Our case study considers the coastal French municipality of Gâvres, located on a peninsula at the Blavet river mouth, in the conurbation of Pays de Lorient (Morbihan). This region is representative of a significant part of French mainland coasts in terms of variety and complexity of flooding processes, as well as available offshore data. Since 1864, Gâvres has had to deal with more than ten coastal flooding events, two of the most dramatic ones taking place in the 21st century. Flooding processes at Gâvres are known to be complex enough (tide influence and overtopping) to cover most of the flooding cases along the French mainland coasts. This ensures the scalability of the methods presented here, to any coastal flooding type.

2.1. Hydrodynamic code

Here we consider a simplified fast running code defined on cross-shore transect model (see Figure 1, and the next paragraph for the description). The code takes four variables with physical interpretability as inputs. Those are the tide (Tp), atmospheric storm surge (Sg), significant wave height (Hs) and wave peak period (Td). Each input should be provided to the system in a time series format, so that $\boldsymbol{Td} = (Td_t)_{t=1,\ldots,T}$, and similarly for the other three inputs. As the output, the code delivers a time series of the same length of the inputs, with the value of the average amount of water entering inland at each time instant $t \in \{1, \ldots, T\}$, expressed as a volume. From that series, it is also possible to compute the cumulative amount of water entered inland along this transect at each time instant t. We note that quantity as CV_t. In this paper we focus on the management of functional inputs and try to keep the output as simple as possible. Therefore, we study a scalar output instead of a functional one. In particular, we consider as the output the total amount of water entering inland, corresponding to the last value of the cumulative amount of water series, CV_T. From here on, we will note CV_T as FCV, referring to *final cumulative volume*.

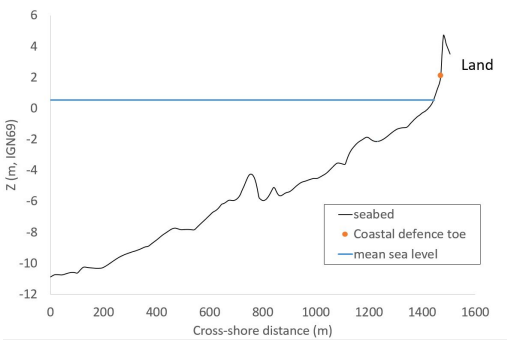


Figure 1: Illustration of the cross-shore transect considered in the RISCOPE application.

Computations inside the computer code involve the statistical model SWAN [21] and the EurOtop equations [22], both described below.

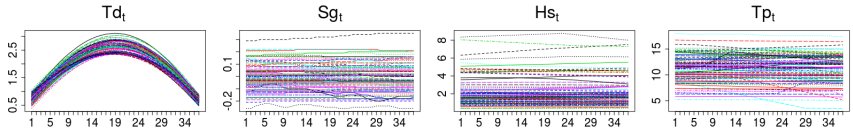
Inputs → **SWAN** → **EurOtop** → Outputs

- **SWAN** is a spectral wave model which allows computing the wave conditions at the coastal defence toe, accounting for water level variations induced by tide and surge.
- **EurOtop** refers to the use of the overtopping and overflow discharges formulas provided in the Eurotop (2018) manual ([22], Eq. 5.11 and 5.20). These formulas require as input the wave conditions at the coastal defence toe, but also the crest freeboard (water height above the coastal defence crest, including the wave setup computed by SWAN plus the tide and surge) and coastal defence characteristics. Based on the computed discharge, the volume of water entering inland along the transect (Figure 1) is finally computed.

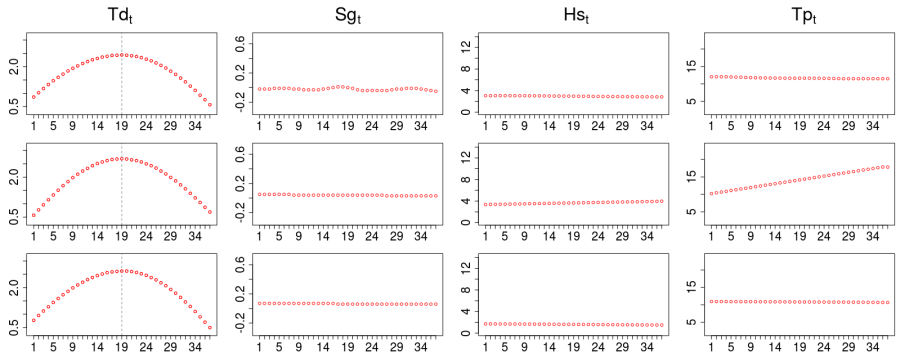
We are aware that the adoption of a cross shore configuration does not allow to properly model all the complexities of the phenomenon under study. However, in the frame of the RISCOPE project, the analysis presented here is considered an intermediary step in the development of methodologies for functional metamodeling, that could be later implemented for more realistic computer codes. The use of a simplified computer code at this stage enables a wider exploration and understanding of the physical phenomenon, before dealing with more detailed - and thus, more computationally time consuming - hydrodynamic models. We remark that FCV is equal to the sum over t of the volume entering inland at each instant CV_t, estimated from scalar inputs. Thus, for this simplified code, a metamodel based on a scalar representation of the inputs may provide relatively good predictions. However, in a further stage of the RISCOPE project we will address intrinsically functional problems such as the estimation of the water height on land (i.e., at different points in the space between the back of the coastal defense and inland). At that point, we naturally expect the functional metamodels to be able to better reproduce the shape of the output than the scalar ones.

2.2. Data set

For purposes of construction and validation of the metamodel, we count on a data set composed of hindcasts of past conditions for tide, atmospheric storm surge, significant wave height and wave peak period, offshore of the study site over the period 1900-2016. The data set is constituted by the concatenation of hindcasts of different sources (see Appendix A), but with bias corrections between the hindcasts through a quantile-quantile correction method. The various hindcasts have different time steps. As the main driver (the tide) significantly changes in 10 minutes, the other three inputs are also interpolated at a 10 min time step. Then, the long data set was split into a collection of tidal events, each covering a period of $\pm$ 3 hours around a high tide. A time series of 37 elements (corresponding to a time lapse of 6 hours with the time step of 10 min) was used to represent each functional input at each event (see Figure 2). Only events where the tide reached at least 2.342m (IGN69) were kept. This value corresponds to the mean spring high tide level below which no flooding event ever happened in Gâvres. As a result, a total of 20557 events were obtained.



(a) 100 randomly sampled events.



(b) 3 sample events on independent plots.

Figure 2: Illustration of functional inputs. *Td*, *Sg* and *Hs* are given in meters and *Tp* in seconds.

The tide series has a characteristic parabolic shape, which is consistent among events. In fact, its peak (always located at time istant $t = 19$) is known to be highly influential on the output of the code. In contrast, the majority of *Sg*, *Hs* and *Tp* curves are almost constant or linear with small slope. It means that the range of variation of those three inputs within each event is relatively small compared to their range of variation among events. Based on that, one could presume that just one or two scalar parameters associated to the height and/or slope would be enough to characterise those curves. However, beyond any conclusion that we could reach by visual inspection, the ideal dimension will depend on the sensitivity of the output of the code to changes of each input. Even quite small and visually negligible changes on some input could cause important changes in the ouput depending on the interactions that happen within the code.

3. Theoretical background

3.1. Metamodeling using Gaussian processes in case of scalar inputs

Let us consider a regression framework, where an expensive computer code f_{code} is available to model the relationship between a group of input variables $\boldsymbol{x} \in \mathbb{R}^d$ and an output variable of interest $y \in \mathbb{R}$. As the evaluation of f_{code} is computationally costly, it is desired to build a light-to-run mathematical model to approximate it. To this end, there is available a learning set $D = \{(\boldsymbol{x_1}, y_1), \ldots, (\boldsymbol{x_n}, y_n)\}$, with $y_i = f_{\text{code}}(\boldsymbol{x_i})$. In this context, Gaussian processes are nonparametric regression models which treat the fixed function f_{code} as a realization of a Gaussian process ξ, specified by its mean and covariance functions m and k. The Gaussian process model can be written as:

$$f_{\text{code}}(\cdot) \sim \xi(m(\cdot), c(\cdot,\cdot)), \tag{1}$$

with

$$m(\boldsymbol{x}) = \mathbb{E}[\xi(\boldsymbol{x})], \tag{2}$$
$$k(\boldsymbol{x_1}, \boldsymbol{x_2}) = \mathbb{E}[(\xi(\boldsymbol{x_1}) - m(\boldsymbol{x_1}))(\xi(\boldsymbol{x_2}) - m(\boldsymbol{x_2}))]. \tag{3}$$

Gaussian processes present diverse attributes that have contributed to their popularity in many applications. They provide a mean estimate along with an indication of the uncertainty attached to it. They are able to reproduce the observations exactly, but there is a simple way to switch from interpolation to smoothing by means of a nugget effect, if required (see [12] for more details). Furthermore, the Gaussian process model often has a very high prediction power compared to other approaches [4]. In adition, the conditional distribution of Gaussian processes, given observed values, is particularly tractable in practice and closed form expressions exist for the conditional mean and variance. We discuss them below.

Let $\boldsymbol{X} = (\boldsymbol{x_1}, \ldots, \boldsymbol{x_n})^\top$ be the $n \times d$ inputs matrix extracted from the learning set (where $\boldsymbol{x_i}$ for $i = 1, \ldots, n$ is a column vector), and let $\boldsymbol{y} = (y_1, \ldots, y_n)^\top$ be the vector of corresponding output values. Similarly, let $\boldsymbol{X_*} = (\boldsymbol{x_{*,1}}, \ldots, \boldsymbol{x_{*,n_*}})^\top$ be a $n_* \times d$ inputs matrix of prediction points. The Gaussian conditioning theorem (see e.g., [12]) implies that, conditionally to $\boldsymbol{y}$, ξ is a Gaussian process with mean and covariance functions m_n and k_n defined by

$$m_n(\boldsymbol{X_*}) := \mathbb{E}[\xi(\boldsymbol{X_*}) | \boldsymbol{X}, \boldsymbol{y}] = K(\boldsymbol{X_*}, \boldsymbol{X}) K(\boldsymbol{X}, \boldsymbol{X})^{-1} \boldsymbol{y} \tag{4}$$

and

$$\begin{aligned} k_n(\boldsymbol{X_*}, \boldsymbol{X_*}) &:= \mathrm{Cov}[\xi(\boldsymbol{X_*}), \xi(\boldsymbol{X_*}) | \boldsymbol{X}, \boldsymbol{y}] \\ &= K(\boldsymbol{X_*}, \boldsymbol{X_*}) - K(\boldsymbol{X_*}, \boldsymbol{X}) K(\boldsymbol{X}, \boldsymbol{X})^{-1} K(\boldsymbol{X}, \boldsymbol{X_*}). \end{aligned} \tag{5}$$

Here $K(\boldsymbol{X}, \boldsymbol{X})$ denotes the $n \times n$ matrix of covariances $(k(\boldsymbol{x_i}, \boldsymbol{x_j}))_{1 \le i,j \le n}$ among all pairs of observation points, and similarly for the other entries $K(\boldsymbol{X}, \boldsymbol{X_*})$, $K(\boldsymbol{X_*}, \boldsymbol{X})$ and $K(\boldsymbol{X_*}, \boldsymbol{X_*})$. We remark that $m_n(\boldsymbol{X_*})$ and $k_n(\boldsymbol{X_*}, \boldsymbol{X_*})$ are of the form

$$\begin{aligned} m_n(\boldsymbol{X_*}) &= (\mathbb{E}[\xi(\boldsymbol{x_{*,i}}) | \xi(\boldsymbol{x_1}) = y_1, \ldots, \xi(\boldsymbol{x_n}) = y_n])_{1 \le i \le n_*}, \\ k_n(\boldsymbol{X_*}, \boldsymbol{X_*}) &= (\mathrm{Cov}[\xi(\boldsymbol{x_{*,i}}), \xi(\boldsymbol{x_{*,j}}) | \xi(\boldsymbol{x_1}) = y_1, \ldots, \xi(\boldsymbol{x_n}) = y_n])_{1 \le i,j \le n_*}. \end{aligned}$$

In practice the conditional mean (see Equation (4)) is used as an estimation of the true function f_{code} at the test points $\boldsymbol{X_*}$, while the conditional variance (see Equation (5)) is often interpreted as a measure of the local error of the prediction [9].

Gaussian process models are flexible by incorporating diverse types of covariance functions, being aware that only functions that yield symmetric positive semidefinite covariance matrices are valid kernels [12]. The choice of the covariance function encodes assumptions such as the degree of regularity of the

underlying process [23]. Examples of standard covariance functions are given for instance in [24] and [25]. The covariance between any given pair of input coordinates $(\boldsymbol{x_1}, \boldsymbol{x_2})$, depends on the variance of the stochastic process σ^2 and on the correlation function R which governs the degree of correlation through the use of the vector of length-scale parameters $\boldsymbol{\theta}$ (see Equation (6) for its general form). Two of the most popular correlation functions are the Gaussian kernel (see Equation (7)) and the Matérn 5/2 kernel (see Equation (8)).

$$\underline{\textit{General:}} \qquad k(\boldsymbol{\tau}; \sigma^2, \boldsymbol{\theta}) = \sigma^2 \; R(\boldsymbol{\tau}; \boldsymbol{\theta}), \tag{6}$$

$$\underline{\textit{Gaussian:}} \qquad k(\boldsymbol{\tau}; \sigma^2, \boldsymbol{\theta}) = \sigma^2 \; exp\left(-\frac{\|\boldsymbol{\tau}\|^2_{\boldsymbol{\theta}}}{2}\right), \tag{7}$$

$$\underline{\textit{Matérn 5/2:}} \quad k(\boldsymbol{\tau}; \sigma^2, \boldsymbol{\theta}) = \sigma^2 \left(1 + \sqrt{5}\; \|\boldsymbol{\tau}\|_{\boldsymbol{\theta}} + \frac{5\; \|\boldsymbol{\tau}\|^2_{\boldsymbol{\theta}}}{3}\right) exp\left(-\sqrt{5}\; \|\boldsymbol{\tau}\|_{\boldsymbol{\theta}}\right). \tag{8}$$

Here $\boldsymbol{\tau} = \boldsymbol{x_1} - \boldsymbol{x_2}$, σ^2 and $\boldsymbol{\theta}$ are the so called hyperparameters of the model which have to be estimated, and $\|\boldsymbol{\tau}\|_{\boldsymbol{\theta}} = \sqrt{\frac{\tau_1^2}{\theta_1^2} + \cdots + \frac{\tau_d^2}{\theta_d^2}}$ is the anisotropic L^2 norm of $\boldsymbol{\tau}$. Intuitively, if $\boldsymbol{x_1} = \boldsymbol{x_2}$, then the correlation is 1, whereas if the distance between both vectors tends to infinity, then the correlation tends to 0.

3.2. Treatment of functional inputs

Gaussian process models were first developed for scalar inputs. That is the case of a model of the form $\boldsymbol{x} \to \xi(\boldsymbol{x})$. However, they can be also built for functional inputs or a combination of both, scalar and functional inputs, which would correspond to the models $\boldsymbol{f} \to \xi(\boldsymbol{f})$ and $(\boldsymbol{x}, \boldsymbol{f}) \to \xi(\boldsymbol{x}, \boldsymbol{f})$, respectively. The objective of this paper is the third type of model, which considers a combination of scalar and functional inputs.

3.2.1. Three distances complete

As for scalars inputs, Gaussian processes dealing with functional inputs require the selection of a proximity or similitude measure, enabling to discriminate among input coordinates. Here we describe the three main approaches adopted in the literature, taking as reference the L^2 norm for functions. Other types of norms such as weighted norms or general p norms, could also be chosen depending on the physical or geometric interpretation of the functions, as well as the notion of dissimilarity among functions for the particular application case. The main difference among the three approaches lies in the way of representing observations of the functional inputs. The first approach, for instance, works with a modified L^2 norm of the form:

$$\|f_1 - f_2\|_{f,\theta} = \sqrt{\frac{\int_{\mathrm{T}} (f_1(t) - f_2(t))^2 dt}{\theta^2}}, \tag{9}$$

with

$$f_1, f_2 \in L^2(T, \mathbb{R}) = \left\{ f : T \to \mathbb{R}, \int_{\mathrm{T}} f^2(t) dt < \infty \right\}. \tag{10}$$

If the computer code receives polynomial expressions as inputs, the integral could be solved by ordinary calculus. Otherwise, if a temporal series representation is used, the integral could be computed by numerical approximation over a large series of time instants. An alternative perspective could be to use a distance measure for time series such as the Manhattan, Minkowski or infinite norm (see e.g., [26]).

For the second approach, we consider a subspace of $L^2(T, \mathbb{R})$ of dimension p and let $\boldsymbol{B} = \{B_1, \ldots, B_p\}$ be a basis of this space. We let $\Pi_p(f)$ denote the projection of the functional input f onto the space generated by the basis $\boldsymbol{B}$, which can be written as:

$$\Pi_p(f)(t) = \sum_{k=1}^{p} \alpha_k B_k(t). \tag{11}$$

Here, the scalar expansion coefficients $\alpha_1, \ldots, \alpha_p$ indicate the weight (contribution) of each of the p basis functions. The dimension p has to be chosen strategically so that the functions are represented well enough and computations for the metamodel remain tractable. Diverse decomposition methods such as Wavelets, Fourier, B-splines or PCA could be used and different approaches to set the values of the expansion coefficients exist as well. We discuss those two aspects in Section 3.2.2 and Appendix B, respectively.

If each observation of the functional input is represented using a decomposition of the same dimension p, then (9) becomes:

$$\|\Pi_p(f_1) - \Pi_p(f_2)\|_{d,\theta} = \sqrt{\frac{\int_{\mathrm{T}} \left(\sum_{k=1}^{p} (\alpha_{1,k} - \alpha_{2,k}) B_k(t) \right)^2 dt}{\theta^2}} \tag{12}$$

where $\alpha_{i,k}$ is the coefficient of the k-th basis function for the projection of the i-th observation of the functional input.

The third approach is a variation of the second one, where the distance only considers the coefficients of the decomposition:

$$\|\Pi_p(f_1) - \Pi_p(f_2)\|_{s,\boldsymbol{\theta}} = \sqrt{\sum_{k=1}^{p} \frac{(\alpha_{1,k} - \alpha_{2,k})^2}{\theta_k^2}}. \tag{13}$$

This approach is equivalent to taking the coefficients of the decomposition as independent scalar inputs of the metamodel. Note that a similar distance is obtained from (12) if $\boldsymbol{B}$ is an orthonormal family. However, (12) involves a single correlation coefficient θ, while (13) involves p of them. The distance in (13) can be interpreted as that in (12) in the case where $B_1, \ldots, B_p$ are an orthogonal family with $\|B_k\| = \theta_k^{-1}$ and $\theta_k = 1$.

As general expressions to handle multiple functional inputs, we let

$$\|\boldsymbol{f_1} - \boldsymbol{f_2}\|_{F,\boldsymbol{\theta}} = \sqrt{\sum_{\ell=1}^{df} \|f_1^\ell - f_2^\ell\|_{f,\theta_\ell}^2}, \tag{14}$$

$$\|\Pi_p(\boldsymbol{f_1}) - \Pi_p(\boldsymbol{f_2})\|_{D,\boldsymbol{\theta}} = \sqrt{\sum_{\ell=1}^{df} \|\Pi_p(f_1^\ell) - \Pi_p(f_2^\ell)\|_{d,\theta_\ell}^2}, \tag{15}$$

and

$$\|\Pi_p(\boldsymbol{f_1}) - \Pi_p(\boldsymbol{f_2})\|_{S,\boldsymbol{\theta}} = \sqrt{\sum_{\ell=1}^{df} \|\Pi_p(f_1^\ell) - \Pi_p(f_2^\ell)\|_{s,\boldsymbol{\theta}_\ell}^2}. \tag{16}$$

Here df is an integer denoting the number of functional inputs in the model. In (16), $\boldsymbol{\theta} = (\boldsymbol{\theta_1}, \ldots, \boldsymbol{\theta_{df}})$ has dimension $p \times df$ as an independent correlation length coefficient is matched to each basis function of each functional input. In contrast, $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_{df})$ is a df-dimensional vector in (14) and (15), as a single correlation length coefficient is used for all the basis functions associated to the same functional input.

For applications of the three methods, the reader is referred to [27], [4] and [5], in the corresponding order. To the best of our knowledge, up to now there is no evidence of the superiority of any of the three methods over the others. In fact, the best working method seems to vary from one application case to the other. In this paper, we focus on the two approaches based on functional decomposition (Equations (15) and (16)), as they fit better to the application case that motivated this paper. Among those two approaches, the one where the coefficients of the projection are taken as independent scalar inputs (Equations (13) and (16)) is straightforward to implement, since the classic covariance functions for scalars can be directly applied. In contrast, the second method (Equations (12) and (15)), requires some developments in order to adaptate standard covariance funtions to deal with the projection. Such special treatement is the topic of the folllowing section.

3.2.2. Adapted covariance function for projections

In this study we focus on the B-splines and PCA projection methods, which are attractive tools for the modeling of functional inputs (see for instance [4, 28, 13] for applications of B-splines and [5, 15] for applications of PCA). In both cases, the functional input is represented by a linear combination of coefficients α_k and basis functions B_k, $k = \{1, \ldots, p\}$, as in (11). Recall that the projection dimension p has to be strategically chosen. Here we discuss the basics of each method. For a more complete introduction and full derivation of B-splines and PCA equations, the interested reader can refer to [16] and [29], respectively.

B-splines of order m are constructed from polynomial pieces of degree $m-1$, joined at a sequence of points in the domain of the input, called the knots. A B-spline requires a total of $p - m + 2$ knots to be constituted, with additional $m - 1$ replicates for the first and the last knot

$$\tau_1 = \cdots = \tau_{m-1} = \tau_m < \tau_{m+1} < \cdots < \tau_p < \tau_{p+1} = \tau_{p+2} = \cdots = \tau_{p+m}.$$

Given a knots vector, the B-splines basis of any desired order m are given by the following system of recursive equations:

$$B_{i,1}(t) = \mathbf{1}_{[\tau_i, \tau_{i+1}]}(t) \qquad \text{for } i \in \{1, \ldots, p + m - 1\}$$

and

$$B_{i,m}(t) = \frac{t-\tau_i}{\tau_{i+m-1}-\tau_i} B_{i,m-1}(t) + \frac{\tau_{i+m}-t}{\tau_{i+m}-\tau_{i+1}} B_{i+1,m-1}(t) \qquad \text{for } i \in \{1,\ldots,p\},$$

with $B_{i,m} = 0$ if $\tau_i = \cdots = \tau_{i+m} = 0$ to avoid division by zero.

For PCA we consider the $n \times T$ matrix $\boldsymbol{F}$ containing n observations of the functional input under study at T time instants, that is, the matrix composed of the set of discretized temporal curves $\boldsymbol{f_i}$ (with $i = 1,\ldots,n$). Similarly, we let $\boldsymbol{F_c}$ be the centered input matrix that has the respective column means subtracted from each element of $\boldsymbol{F}$:

$$\boldsymbol{F_c} = \boldsymbol{F} - \boldsymbol{1}_n \bar{\boldsymbol{f}}', \tag{17}$$

with $\bar{\boldsymbol{f}}' = (\bar{f}_1,\ldots,\bar{f}_T)$ denoting the vector of column means of $\boldsymbol{F}$, and $\boldsymbol{1}_n$ denoting a n-dimensional column vector of ones. Then, we define the variance–covariance matrix $\boldsymbol{\Sigma} = \frac{1}{n}\boldsymbol{F_c}' \cdot \boldsymbol{F_c}$ of the columns of $\boldsymbol{F}$. The PCA decomposition is based on the expansion of $\boldsymbol{\Sigma}$ as follows:

$$\boldsymbol{\Sigma} = \sum_{k=1}^{T} \lambda_k \boldsymbol{v}_k \boldsymbol{v}_k', \tag{18}$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_T$ are the eigenvalues of $\boldsymbol{\Sigma}$ and $\boldsymbol{v}_k, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_T$ are mutually orthogonal eigenvectors associated with these eigenvalues. In the PCA framework, the basis functions $B_1,\ldots,B_T$ correspond to the eigenvectors $\boldsymbol{v}_k$.

B-spline basis are independent of the data set and may be of any desired dimension. In contrast, PCA is purely data driven and the basis size is bounded by the number of observed time instants T, as basis functions correspond to eigenvectors of the matrix $\boldsymbol{\Sigma}$. An attractive property of PCA is the orthonormality of the basis. This characteristic simplifies some of the computations required for metamodeling. For instance, in the frame of Design of Experiments, optimizing a maximin criterion based on the quadratic distance of the coefficients is equivalent to optimizing a maximin criterion based on the L^2 distance of the corresponding functions [30, 5].

For both methods, an efficient expression for the L^2 norm of $f_1 - f_2$ can be implemented, whenever both functions are projected on the same basis. In the case of B-splines, the two projections should have the same size p, order m, and knots vector $\boldsymbol{\tau}$ in order to be considered as the same basis. For PCA, the eigenvectors (basis functions) of the two projections must come from the same covariance matrix $\boldsymbol{\Sigma}$ to have the same basis. If the stated conditions are met, the norm $\|f_1 - f_2\|_{L^2}$ approximated by $\|\Pi_p(f_1) - \Pi_p(f_2)\|_{L^2}$ reduces to a norm in $\mathbb{R}^p$, with $p \in \mathbb{N}$ representing the dimension of the decomposition [4]:

$$
\begin{aligned}
\|\Pi_p(f_1) - \Pi_p(f_2)\|_{L^2}^2 &= \int_{\mathrm{T}} \left(\sum_{k=1}^{p} \alpha_k^1 B_k(t) - \sum_{k=1}^{p} \alpha_k^2 B_k(t) \right)^2 dt \\
&= \int_{\mathrm{T}} \left(\sum_{k=1}^{p} (\alpha_k^1 - \alpha_k^2) B_k(t) \right)^2 dt \\
&:= \int_{\mathrm{T}} \left(\sum_{k=1}^{p} \delta_k^{1,2} B_k(t) \right)^2 dt \\
&= \int_{\mathrm{T}} \sum_{j,k=1}^{p} \delta_j^{1,2} \delta_k^{1,2} B_j(t) B_k(t) \; dt = \boldsymbol{\delta}' \boldsymbol{J} \boldsymbol{\delta}. \qquad (19)
\end{aligned}
$$

Here $\mathrm{T} \subset \mathbb{R}$ is the domain of the functional input and $\boldsymbol{J}$ is the $p \times p$ Gram matrix $\left(\int_{\mathrm{T}} B_j(t) B_k(t) dt\right)_{1 \leq j,k \leq p}$. The attractive fact of this derivation proposed in [4] is that the matrix $\boldsymbol{J}$ does not depend on the coefficients of the decomposition but only on the basis functions, and thus it can be stored and reused, saving computational time. Moreover, when the adopted basis is orthonormal (e.g., PCA basis), $\boldsymbol{J}$ is simply the identity matrix of dimension $p \times p$.

Now that the measure of proximity has been defined for functions, an extended version of the anisotropic, tensor-product kernel can be implemented in order to include functional inputs in the covariance function:

$$
\mathrm{Cov}(Z(\boldsymbol{x_1}, \boldsymbol{f_1}), Z(\boldsymbol{x_2}, \boldsymbol{f_2})) = \sigma^2 \; k\left(D_s(\boldsymbol{x_1} - \boldsymbol{x_2}; \boldsymbol{\theta_s})\right) \; k\left(D_f(\boldsymbol{f_1} - \boldsymbol{f_2}; \boldsymbol{\theta_f})\right), \quad (20)
$$

where $D_s(\boldsymbol{x_1} - \boldsymbol{x_2}; \boldsymbol{\theta_s})$ and $D_f(\boldsymbol{f_1} - \boldsymbol{f_2}; \boldsymbol{\theta_f})$ denote distances for the scalar and functional inputs, respectively, scaled by the corresponding length-scale parameters $\boldsymbol{\theta_s}$ and $\boldsymbol{\theta_f}$. For the Gaussian kernel, (20) becomes

$$
\begin{aligned}
\mathrm{Cov}(Z(\boldsymbol{x_1}, \boldsymbol{f_1}), Z(\boldsymbol{x_2}, \boldsymbol{f_2})) = \sigma^2 \; & exp\left(-\frac{1}{2} \sum_{\ell=1}^{ds} \left(\frac{x_1^\ell - x_2^\ell}{\theta_s^\ell} \right)^2 \right) \\
& exp\left(-\frac{1}{2} \sum_{\ell=1}^{df} \frac{(\boldsymbol{\delta}_{1,2}^\ell)' \boldsymbol{J} \boldsymbol{\delta}_{1,2}^\ell}{\left(\theta_f^\ell\right)^2} \right). \qquad (21)
\end{aligned}
$$

Here ds and df denote the number of scalar and functional input variables, respectively. Following Equation (19), the term $\boldsymbol{\delta}_{1,2}^\ell$ denotes the vector of differences among the coefficients of the decomposition of the ℓ-th functional input at observation points 1 and 2.

4. Exploration strategy

The construction of a surrogate model requires making a series of decisions that may have significant impact on its resulting performance. The distance used to measure similarity among functions (see Section 3.2), the projection method

used to represent functional inputs (e.g. B-splines, PCA, wavelets, polynomials), as well as the dimension of the projection, are all part of those decisions. To the best of our knowledge, there is no evidence of a best performing level for any of the parameters mentioned above. In fact, the suitability of different levels of those parameters may depend on the application, and thus, should be optimized each time a metamodel is to be built, in order to achieve the best possible performance. However, such an optimization is rarely performed in the literature and the levels of those parameters are often fixed based on results from previous studies. In particular, the projection dimension is often fixed using the accuracy of the projection itself as criterion. The deficiency of this approach is that the dimension that provides the highest accuracy in terms of projection of the inputs, does not necessarily translate into good metamodel predictability.

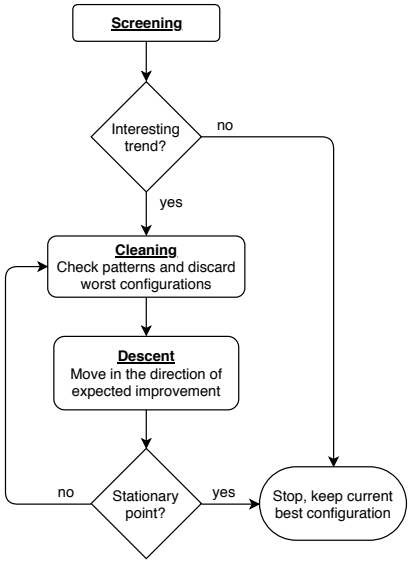


Figure 3: Exploration strategy flowchart.

To overcome the issues described above, here we propose a methodology that aims at exploring different configurations of some structural parameters of the surrogate model, while determining a good dimension to represent the functional inputs. A scheme of the proposed methodology is presented in Figure 3 and its main steps are briefly described below:

1. *Screening.* This step is intended to provide an overview of the effect of the parameters on the performance of the metamodel. The main objectives here are to:
    - Identify patterns, as could be the dominance of certain levels of a parameter over the other levels. For intance, determine if some projection method clearly outperforms the others.
    - Detect trend in numerical parameters, for instance, the projection dimension. See if the performance of the metamodel improves by increasing or decreasing the value of such parameters.

- Determine if the functional representation of functional inputs adds information to the metamodel or if a scalar representation is enough. To do so, a metamodel using a scalar representation of the functional inputs is used as a benchmark, which is compared to metamodels based on functional decompositions of the inputs.

In ideal circumstances, the performance of the metamodel can be assessed using an independent validation dataset from the one used for training. However, in most metamodeling applications, the number of simulations available is quite limited to proceed in that way. The application of cross validation / bootstrap methods (see e.g. [31]) is recomended for those cases.

As one of the main purposes of the exploration methodology is to reduce considerably the dimension of the inputs, we start by exploring configurations with the lowest possible dimensions. For instance, configurations of dimension 1, 2 and 3.

2. *Cleaning.* If the screening stage allows to detect a tendency to better performance on highest projection dimensions, the exploration is extended in such direction. However, depending on the number of structural parameters under study, the extension of the experiment could become too time consuming. Therefore, the cleaning stage consists on discarding those dominated levels of the parameters, identified through the patterns on the screening stage.
3. *Descent.* Once the dominated levels have been discarded, greater values of projection dimension are evaluated, building a new factorial experiment only with the non-dominated levels of the other parameters. Here we call this stage *descent* as we aim to find metamodels providing lower error as we increase the dimension size. Similarly to the response surface methodology (see [32] and [33]), this stage is repeated until a stationary point is identified.

Despite the fact that the proposed methodology may require the evaluation of several experimental conditions, all of them could be performed using the same set of observations of the expensive code. Thus, the computational cost of the methodology is expected to be reasonable and the potential gains in terms of metamodel performance could be worth it. The following example is used to illustrate the methodology.

4.1. Analytic case

In this section, we consider the second analytic case presented in [4], with a slight different domain for the functional inputs. In [4], a Gaussian process metamodel is built using B-spline decompositions of dimension 5 and order 4 to represent the functional inputs. Here, we use the exploration strategy presented above to find an attractive metamodel configuration among several combinations of decomposition dimension, projection method and type of covariance function.

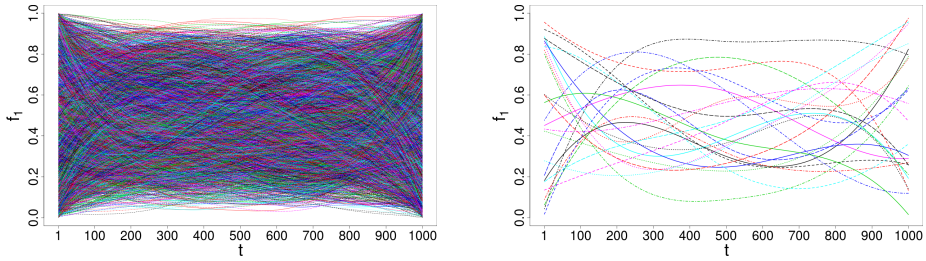
Let $\mathcal{F}$ be the set of continuous functions from $[0, 1000]$ to $\mathbb{R}$. Then, consider a black box computer code receiving the scalar inputs $\boldsymbol{x} = (x_1, x_2) \in [0, 1]^2$ and the continous functional inputs $\boldsymbol{f} = (f_1, f_2) \in \mathcal{F}^2$, defined as:

$$
\begin{aligned}
\mathcal{G}: \quad & [0,1]^2 \times \mathcal{F}^2 \to \mathbb{R}, \\
& (\boldsymbol{x}, \boldsymbol{f}) \mapsto \left(x_2 - \frac{5}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 \\
& \quad + 10 \left(1 - \frac{1}{8\pi} \right) \cos(x_1) + 10 \\
& \quad + \frac{4}{3}\pi \left(42 \int_0^{1000} 15 f_1(t)\,(1-t)\,dt \right. \\
& \quad \left. + \pi \left(\frac{x_1 + 5}{5} + 15 \right) \int_0^{1000} 15 t f_2(t) dt \right).
\end{aligned}
$$

Note that $\mathcal{G}$ is an instrinstic functional code, since the integrals over the domain of the inputs and the interactions between functional and scalar variables make it not possible to recover the output by means of independent computations on scalar representations of the input over its domain. That fact gives an insight on the type of metamodel that should be used; at least, we expect functional metamodels to be an interesting alternative here.

4.1.1. Data set

We started by creating a data set with 5000 observations of the code that could be used later to generate multiple independent training and validation sets. The coordinates of the 5000 observation points for the scalar inputs where uniformily sampled over their domain. For the functional part, we followed an approach proposed in [13], which is to make the design over the coefficients of a basis. Here we modeled each functional input as a B-spline of dimension 5 and order 4, and then we constructed a Latin Hypercube design (LHD) [34] with 5000 observation points for the coefficients of the basis. We remark that the order and dimension used for the constitution of the data set is independent of the order and dimension to be used later for the representation of the inputs in the metamodel. As the focus of this paper is not optimal design of experiments, we do not develop further on the construction of the data set, and we assign the 5000 scalar coordinates to the 5000 functional coordinates using a random permutation. For a more elaborated approach to combine the scalar and functional parts of the design, the reader is referred to [4]. The full data set and a sample of 25 observations of the function f_1 are shown in Figures 4a and 4b, respectively.



(a) Full data set. (b) 25 sample functions.

Figure 4: Illustration of the functional input f_1.

4.1.2. Screening

Once the data set was obtained, we set up the screening experiment, which implies the definition of a scalar metamodel to be used as a benchmark for the functional ones. Let $\ddot{\boldsymbol{f}} = (\ddot{f}_1, \ddot{f}_2)$ be the vector of scalar representations of the functional inputs f_1 and f_2. Different scalar parameters could be used to represent the inputs, depending on the geometry and/or the physical meaning of the curves. For simplicity, and as the functions in this theoretical example do not have any physical meaning, here we set the mean of each function as its scalar representation. Now, let us define the scalar metamodel as:

$$\begin{aligned}\mathcal{M}_{00}: \ & [0,1]^2 \times [0,1]^2 \rightarrow \mathbb{R}, \\ & (\boldsymbol{x}, \ddot{\boldsymbol{f}}) \mapsto \mathcal{M}_{00}(\boldsymbol{x}, \ddot{\boldsymbol{f}}).\end{aligned} \tag{22}$$

In order to define the funtional metamodels, let us consider a shifted version of $\boldsymbol{f}$, computed as $\tilde{\boldsymbol{f}} = \boldsymbol{f} - \ddot{\boldsymbol{f}}$. Then, let $\boldsymbol{\Pi_p} = (\Pi_{p,1}, \Pi_{p,2})$ denote the vector of functional decompositions of the elements in $\tilde{\boldsymbol{f}}$ onto a basis of dimension p. For every functional metamodel, we keep $\boldsymbol{x}$ and $\ddot{\boldsymbol{f}}$ as inputs and we add at least one element of $\boldsymbol{\Pi_p}$. This way, the difference in performance among the scalar metamodel and any functional metamodel will be attributed to the addition of the corresponding functional decompositions. As an example, metamodels with a) only $\Pi_{p,1}$ active, b) only $\Pi_{p,2}$ active, and c) both, $\Pi_{p,1}$ and $\Pi_{p,2}$ active, are defined in Equations (23), (24) and (25), respectively.

$$\begin{aligned}\mathcal{M}_{f0}: \ & [0,1]^2 \times [0,1]^2 \times \mathbb{R}^p \rightarrow \mathbb{R}, \\ & (\boldsymbol{x}, \ddot{\boldsymbol{f}}, \Pi_{p,1}) \mapsto \mathcal{M}_{f0}(\boldsymbol{x}, \ddot{\boldsymbol{f}}, \Pi_{p,1}).\end{aligned} \tag{23}$$

$$\begin{aligned}\mathcal{M}_{0f}: \ & [0,1]^2 \times [0,1]^2 \times \mathbb{R}^p \rightarrow \mathbb{R}, \\ & (\boldsymbol{x}, \ddot{\boldsymbol{f}}, \Pi_{p,2}) \mapsto \mathcal{M}_{0f}(\boldsymbol{x}, \ddot{\boldsymbol{f}}, \Pi_{p,1}).\end{aligned} \tag{24}$$

$$\begin{aligned}\mathcal{M}_{ff}: \ & [0,1]^2 \times [0,1]^2 \times (\mathbb{R}^p)^2 \rightarrow \mathbb{R}, \\ & (\boldsymbol{x}, \ddot{\boldsymbol{f}}, \boldsymbol{\Pi_p}) \mapsto \mathcal{M}_{0f}(\boldsymbol{x}, \ddot{\boldsymbol{f}}, \boldsymbol{\Pi_p}).\end{aligned} \tag{25}$$

The notation of the metamodels is such that the subscript indicates which functional decompositions are active. For instance, in $\mathcal{M}_{00}$ both functional decompositions are inactive, while in $\mathcal{M}_{0f}$ only $\Pi_{p,2}$ is active. Furthermore, the notation of metamodels (23), (24) and (25) is generic in the sense that two different decomposition methods can be used to represent the functional inputs (B-splines or PCA), the dimension of the basis p can take values from $\mathbb{N}$, and either the scalar or the projection-based covariance function ($\|\boldsymbol{f_1} - \boldsymbol{f_2}\|_{S,\boldsymbol{\theta}}$ and $\|\boldsymbol{f_1} - \boldsymbol{f_2}\|_{D,\boldsymbol{\theta}}$, respectively) can be used.

A total of 37 experimental conditions were included in the screening experiment, resulting from the scalar metamodel $\mathcal{M}_{00}$, plus all combinations of the levels of the parameters listed below, except for those cases where both $\Pi_{p,1}$ and $\Pi_{p,2}$ are inactive as those correspond to redundant counts of the scalar metamodel.

- State of $\Pi_{p,1}$: inactive or active;
- State of $\Pi_{p,2}$: inactive or active;
- Projection method (Pr): B-plines or PCA;
- Covariance function (Cv): $\|\boldsymbol{f_1} - \boldsymbol{f_2}\|_{S,\boldsymbol{\theta}}$ or $\|\boldsymbol{f_1} - \boldsymbol{f_2}\|_{D,\boldsymbol{\theta}}$;
- Projection dimension (p): 1, 2 or 3 basis functions.

In all cases, the optimization of the expansion coefficients $\alpha_1, \ldots, \alpha_p$ (see Equation (11)) was done using the ordinary least squares formulation proposed in Section Appendix B. The estimation of the parameters is done using the Maximum Likelihood method (see e.g., [35]) in a similar way to the estimation methods used in the R package DiceKriging [36]. In a first step a number of random points in the parameter space are checked for their log-likelihood value and the best is chosen as starting point for the optimization by the R-command *optim*. We used the Matérn 5/2 kernel (see Equation (8)) in every metamodel tested. The quality of the metamodels was assessed using the root mean square error RMSE (see e.g., [37]). For a validation sample $y_1, \ldots, y_{n*}$ with $n_* \in \mathbb{N}$, the RMSE is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{n_*} \sum_{i=1}^{n_*} (\hat{y_i} - y_i)^2}, \tag{26}$$

where $\hat{y_i}$ is the approximation of the observation y_i.

Figure 5 illustrates the performance of the 37 metamodel configurations, using 800 training points and 1500 validation points. A total of 30 independent pairs of training and validation sets were used and the natural logarithm of the average RMSE over the 30 runs (log(RMSE) hereon) is reported in the plot. For convenience in the analysis, the plot classifies metamodel configurations into three groups based on their performance:

(i) the scalar metamodel $\mathcal{M}_{00}$,
(ii) all metamodels with active functional representation of both functional inputs $\mathcal{M}_{ff}$,
(iii) all metamodels with at least one functional representation inactive (except for the scalar metamodel).

For a detailed list of the experimental conditions and corresponding results of the screening stage, the reader is referred to Appendix C, Table C.3.

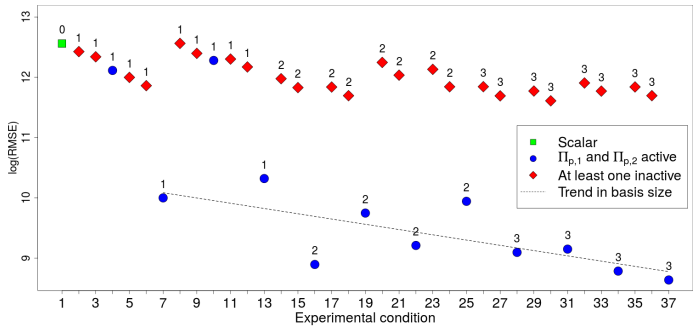


Figure 5: Analytic case: results of the screening experiment. The abscissa denotes the experimental condition (see Table C.3). Points are labeled by basis size p; the label 0 corresponds to the scalar metamodel.

As a first noticeable result, the scalar metamodel was the worst performing one. For $p = 2$ and 3, configurations with both functional representations active (i.e., configurations $16 + 3i$, $i = 0 \ldots, 7$) performed better than the others, while for $p = 1$ only those configurations with PCA representation of both functional inputs (configurations 7 and 13) had outstanding performance. On the other hand, the RMSE tends to decrease as the number of basis functions increases. In fact, the best performing metamodel (configuration 37) matches to a configuration with $p = 3$, the greatest p value tested in the screening experiment. That trend is more visible for the best performing configurations than for the worst performing ones. In Figure 6 we plot the log(RMSE) of all the experimental conditions considered in Figure 5, but this time using training sets of different size. Specifically, we evaluated metamodels trained with sets of size $n = 200, 400$ and 800. As in Figure 5, increasing the projection dimension p clearly delivers a better fit for the best performing metamodels. In addition, for the best configurations found, increasing the training sample size also improves the performance. On the other hand, performance of the worst configurations remains almost the same regardless the projection dimension or the training sample size.

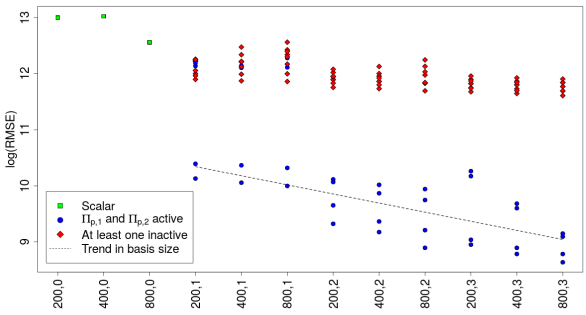


Figure 6: Analytic case: results of screening for different number of training points. The abscissa denotes the combination of training set size n and projection dimension p for each group of experimental conditions. For instance, $(200, 2)$ means a training dataset of size 200 with a projection of dimension 2. As in Figure 5, $p = 0$ corresponds to the scalar metamodel.

4.1.3. Cleaning and descent

As an interesting trend was detected during the screening stage, we proceed to the cleaning and descent stages. The screening stage allowed us to determine that the increase of p reduces the RMSE and also that configurations with the functional representation of the inputs being active dominate any other configuration. Therefore, only those configurations are kept and we start expanding the experiment by increasing the projection dimension p by steps of a unity. At each step, we inspect again for patterns or changes in trend, and cleaning is performed again if possible. The log(RMSE) of the new configurations is ploted in Figure 7. A detailed list of the experimental conditions and corresponding results of this stage is provided in Appendix C, Table C.4. Note that at $p = 7$, we removed the configurations using the scalar covariance function $\|\boldsymbol{f_1} - \boldsymbol{f_2}\|_{S,\boldsymbol{\theta}}$, as those are clearly dominated by configurations using the covariance based on functional decomposition. At $p = 6$, we observed a change in the trend as the RMSE stopped improving. As the difference in RMSE with the best known configuration up to that point was small, we kept increasing the dimension size to see if it improved again. At $p = 8$ we stopped as no further improvement was found.

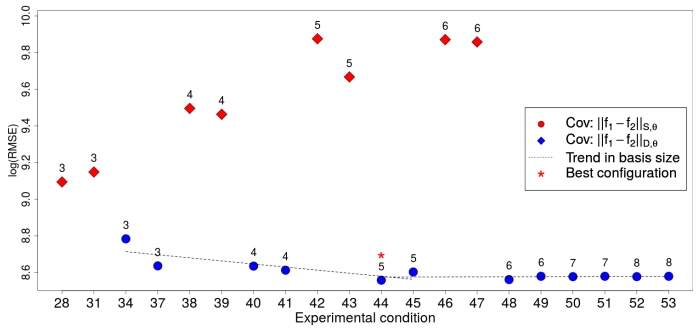
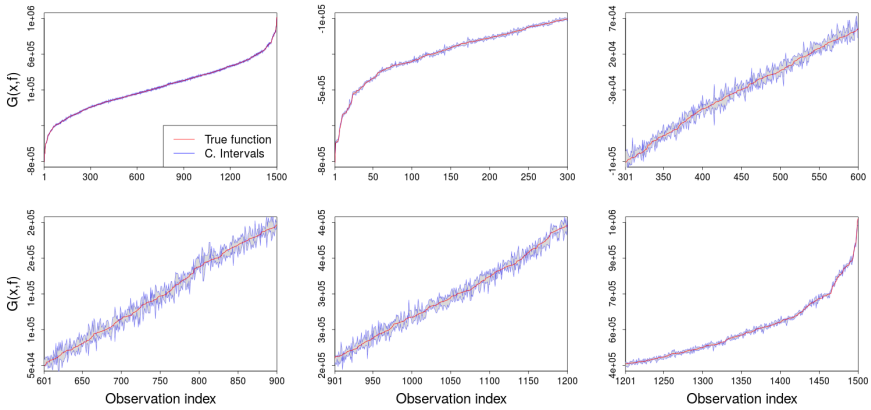
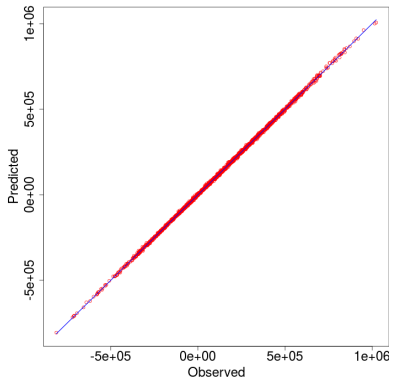


Figure 7: Analytic case: results of cleaning and descent. The abscissa denotes the experimental condition. Points are labeled by basis size p. Configurations with $p = 3$, evaluated in the screening experiment are plotted again here for comparison against configurations with higher p.

For this analytic case, the metamodel with an active B-splines representation of size $p = 5$ for both functional inputs, using the covariance function based on functional decomposition $\|\boldsymbol{f_1} - \boldsymbol{f_2}\|_{D,\boldsymbol{\theta}}$ (configuration 44) is the most attractive configuration found. Note that as we do not know the shape of the RMSE surface, we cannot warrantee that such configuration provides the global optimum. However, we know that its average RMSE corresponds to the 3.84% of the average RMSE of the worst configuration found (configuration 0). Considering that and based on the patterns found during the exploration, configuration 44 is likely one of the best metamodels in terms of RMSE.



(a) Analytic case: fitting of the best performing sample of best configuration. Left-top subplot illustrates the whole set of 1500 output observations in increasing order; this subplot is then subdivided to generate the remaining 5 subplots by splitting the abscissa into 5 sections and zooming the resulting figures.



(b) Analytic case: calibration plot.

Figure 8: Performance of best performing configuration for the analytic case. (a) 1500 observations of the true output sorted in increasing order and corresponding 95% confidence intervals produced by the metamodel; (b) Prediction plot for 1500 data points for the best metamodel configuration.

The fitting of the ordered output as well as the calibration plot of the best of the 30 samples corresponding to the best performing configuration, trained with 800 data points, are presented in Figures 8a and 8b, respectively. For this sample, the proportion of observation points lying withing confidence intervals at 99%, 95% and 90% was 89%, 77% and 65%, respectively. Based on this results and the plots, the metamodel delivers a good prediction with no evident fitting problems (e.g., skewness, heavy tails).

5. Coastal flooding case study

In this section, we address the application case introduced in Section 2. As for the analytic case, we implement our exploration methodology to find

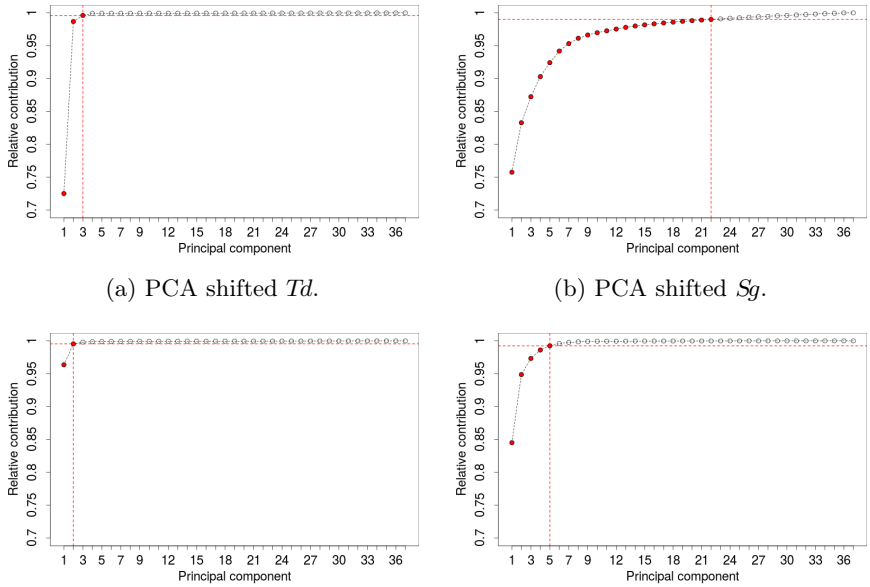
a convenient representation of the functional inputs and select a metamodel configuration of realively high prediction quality.

5.1. Screening

Following the exploration methodology (described in Section 4), we start by performing a screening test oriented to identify patterns, detect trend and determine if the functional representation of the inputs adds value to the metamodel. Similarly to the analytic case, here we denote $\mathbf{f} = (\mathbf{Td}, \mathbf{Sg}, \mathbf{Hs}, \mathbf{Tp})$ the vector of discretized functional inputs (see Section 2) and correspondingly, we denote $\ddot{\mathbf{f}} = (\ddot{\mathbf{Td}}, \ddot{\mathbf{Sg}}, \ddot{\mathbf{Hs}}, \ddot{\mathbf{Tp}})$ the vector of scalar representations of $\boldsymbol{f}$. The peak of $\boldsymbol{Td}$ (the value at time 19) is known to be the most critical and influencing part of the curve, therefore, we use it as its scalar representation. For $\boldsymbol{Sg}$, $\boldsymbol{Hs}$ and $\boldsymbol{Tp}$, we use the mean as their scalar representation, given the relatively constant behavior showed by them in Figure 2. Then we define the scalar benchmark metamodel as:

$$\begin{aligned} \mathcal{M}_{000} : \ \mathbb{R}^4 &\to \mathbb{R}, \\ \ddot{\boldsymbol{f}} &\mapsto \mathcal{M}_{000}(\ddot{\boldsymbol{f}}). \end{aligned} \tag{27}$$

As before, the funtional metamodels require the definition of a shifted version of $\boldsymbol{f}$, which can be computed as $\tilde{\boldsymbol{f}} = \boldsymbol{f} - \ddot{\boldsymbol{f}}$. However, the coastal flooding application has four functional inputs, in contrast to the analytic case (see Section 4.1) which has only two. As the number of experimental conditions to be tested grows exponentially with the number of functional inputs, it could be useful to make a preliminary analysis to determine if some functional inputs could be omitted in the metamodel. In order to decide which functional inputs to keep, we observed the variance explained by the PCA projection of the shifted inputs, as a function of the dimension of the basis (see Figure 9). We observe the plot for shifted inputs as those are the ones that will be used as functional inputs of the metamodel (see the set up for the theoretical example in Section 4.1).



(a) PCA shifted Td.

(b) PCA shifted Sg.

(c) PCA shifted Hs.

(d) PCA shifted Tp.

Figure 9: Coastal flooding case: PCA on the shifted inputs; the dotted red line indicates the number of principal components necessary to explain at least the 99% of the variability of the data.

According to the plots, $\tilde{\boldsymbol{Hs}}$ is the input requiring fewer components to be well described, while $\tilde{\boldsymbol{Sg}}$ and $\tilde{\boldsymbol{Tp}}$ require a considerable number of components. Taking into account this information, as well as observing the shapes of the time series (see Figure 2), we decided to discard $\tilde{\boldsymbol{Hs}}$ as a functional input of the metamodels. However, we keep its scalar representation $\ddot{\boldsymbol{Hs}}$ as part of the experiment, as it does not affect the number of experimental conditions and may help to improve the metamodel. Despite the fact that $\tilde{\boldsymbol{Td}}$ is also well described by just a few components, that input is known to be a main forcing factor and thus, we keep its functional representation in the experiment. Both, the mathematical analysis of the series and the knowledge about the process should be considered during this filtering process.

Based on the discussion above, we let $\boldsymbol{\Pi_p} = (\Pi_{p,1}, \Pi_{p,2}, \Pi_{p,3})$ denote the vector of functional decompositions of $\tilde{\boldsymbol{Td}}$, $\tilde{\boldsymbol{Sg}}$ and $\tilde{\boldsymbol{Tp}}$, respectively, onto a basis of dimension p. For every functional metamodel, we keep $\ddot{\boldsymbol{f}}$ as inputs and we add at least one element of $\boldsymbol{\Pi_p}$. The functional metamodels are defined in the same fashion as in the analytic case (see Section 4.1). For instance, the metamodels with a) only $\Pi_{p,1}$ active, b) $\Pi_{p,2}$ and $\Pi_{p,3}$ active, and c) all three functional decompositions active, are defined in Equations (28), (29) and (30), respectively.

$$\begin{aligned}\mathcal{M}_{f00}:\ & [0,1]^4 \times \mathbb{R}^p \to \mathbb{R},\\ & (\ddot{\boldsymbol{f}}, \Pi_{p,1}) \mapsto \mathcal{M}_{f00}(\ddot{\boldsymbol{f}}, \Pi_{p,1}).\end{aligned} \tag{28}$$

$$\mathcal{M}_{0ff}: \ [0,1]^4 \times \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R},$$
$$(\ddot{\boldsymbol{f}}, \Pi_{p,2}, \Pi_{p,3}) \mapsto \mathcal{M}_{0ff}(\ddot{\boldsymbol{f}}, \Pi_{p,2}, \Pi_{p,3}). \tag{29}$$

$$\mathcal{M}_{fff}: \ [0,1]^4 \times (\mathbb{R}^p)^3 \to \mathbb{R},$$
$$(\ddot{\boldsymbol{f}}, \boldsymbol{\Pi_p}) \mapsto \mathcal{M}_{fff}(\ddot{\boldsymbol{f}}, \boldsymbol{\Pi_p}). \tag{30}$$

The interpretation of the metamodels notation is analogous to that of the metamodels in the analytic case. The subscript indicates which functional decompositions are active. The generic aspect of the notation also holds for metamodels (28), (29) and (30) so that the functional decomposition can be done in a B-spline or PCA basis, whose dimension p can take values from $\mathbb{N}$, and either the scalar or the projection-based covariance function ($\|\boldsymbol{f_1} - \boldsymbol{f_2}\|_{S,\boldsymbol{\theta}}$ and $\|\boldsymbol{f_1} - \boldsymbol{f_2}\|_{D,\boldsymbol{\theta}}$, respectively) can be used.

A total of 85 experimental conditions were included in the screening experiment this time, corresponding to the scalar metamodel plus all combinations of the levels of the parameters listed below, except for those where $\Pi_{p,1}$, $\Pi_{p,2}$ and $\Pi_{p,3}$ are simultaneously inactive, as those cases are already considered in the scalar metamodel.

- State of $\Pi_{p,1}$: inactive or active;
- State of $\Pi_{p,2}$: inactive or active;
- State of $\Pi_{p,3}$: inactive or active;
- Projection method (Pr): B-plines or PCA;
- Covariance function (Cv): $\|\boldsymbol{f_1} - \boldsymbol{f_2}\|_{S,\boldsymbol{\theta}}$ or $\|\boldsymbol{f_1} - \boldsymbol{f_2}\|_{D,\boldsymbol{\theta}}$;
- Projection dimension (p): 1, 2 or 3 basis.

See Appendix D, Table D.5 for a detailed list of the experimental conditions and corresponding results of the screening stage in the coastal flooding case.

In the theoretical example, we used the ordinary least squares formulation to find the coefficients of the decompositions, as all points in the domain of the functions where considered equally important. In the RISCOPE application, the midpoint on the domain of the inputs is known to be of particular importance as it corresponds to the moment of maximum $\boldsymbol{Td}$. Therefore, in this case we used the weighted least squares formulation (see Section Appendix B) instead of the ordinary one. The constrained formulation and the weighted constrained one could also be good choices, and they will be considered in a posterior analysis on Section 5.2.

In the frame of the least squares formulations defined in Section Appendix B, we are considering discretized functional inputs defined over the temporal set $\boldsymbol{t} = \{t_1, \ldots, t_T\}$. For the coastal flooding application, $T = 37$ and the point of outstanding importance is always $t_* = t_{19}$ (see Section 2 and the figures therein). For the weighted least squares formulation we define the vector of weights $\boldsymbol{w} = (w_1, \ldots, w_T)$ in the following way (see Figure 10). We choose $0 \leq \delta \leq t_*$. Then we choose $0 \leq \omega \leq t_* - \delta$ and $(\lambda, \gamma) \in [0,1]^2$. We also define σ^2 as:

$$\sigma^2 = -\frac{\omega^2}{2\ln(\gamma)} \tag{31}$$

Then the weigths are giving the relative importance of the time series at each instant in $\boldsymbol{t}$, be defined by:

$$w_t = \begin{cases} 1 & : \text{ if } t = t_* \\ \lambda & : \text{ if } 0 < |t - t^*| \leq \delta \\ \lambda \exp\left(-\dfrac{(|t - t_*| - \delta)^2}{2 * \sigma^2}\right) & : \text{ if } |t - t^*| > \delta, \end{cases} \tag{32}$$

The weighting function used in the coastal flooding application is illustrated in Figure 10. This type of function, often used to model relevance of potential results regarding a given query on search engines (see e.g., [38]), retains some interesting properties from the Gaussian pdf. For instance, it is a non-negative function with at least one maximum located at the origin t_*. The first case of (32) and the space considered in the definition of the parameters ensures that the greatest possible score is 1.

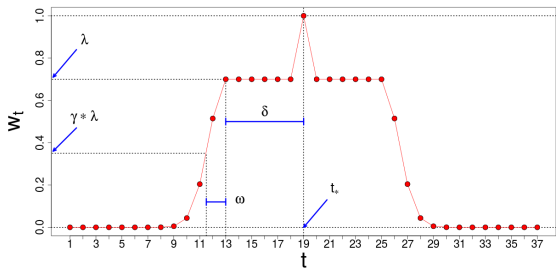


Figure 10: Weighting function for coastal flooding case. In this application, the parameters λ, γ, ω and δ controling the shape of the curve were set to 0.7, 0.5, 1.5 and 6, respectively.

Figure 11 illustrates the performance of the 85 metamodel configurations, using 800 training points and 1500 validation points. Once again, a total of 30 independent pairs of training and validation sets were used and the log(RMSE) over the 30 runs is reported in the results. For convenience in the analysis, the plot classifies metamodel configurations into three groups based on their performance:

(i) the scalar metamodel $\mathcal{M}_{000}$,
(ii) all metamodels using a distance decomposition-based distance $\|\boldsymbol{f_1} - \boldsymbol{f_2}\|_{D,\boldsymbol{\theta}}$,
(iii) all metamodels using a distance based only on the coefficients of the functional decomposition $\|\boldsymbol{f_1} - \boldsymbol{f_2}\|_{S,\boldsymbol{\theta}}$.

See Section 3.2 for the definition of the covariance functions.

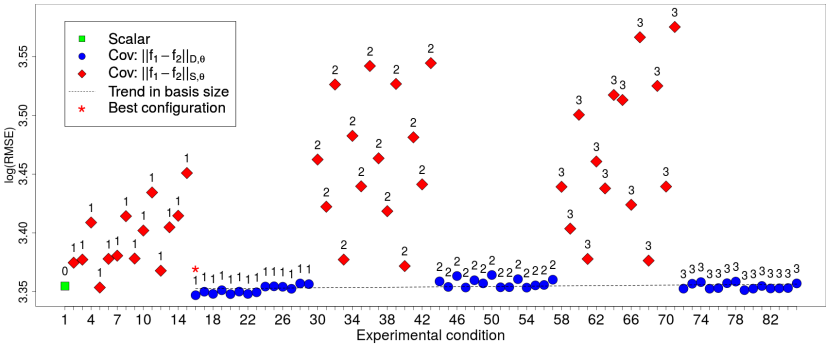
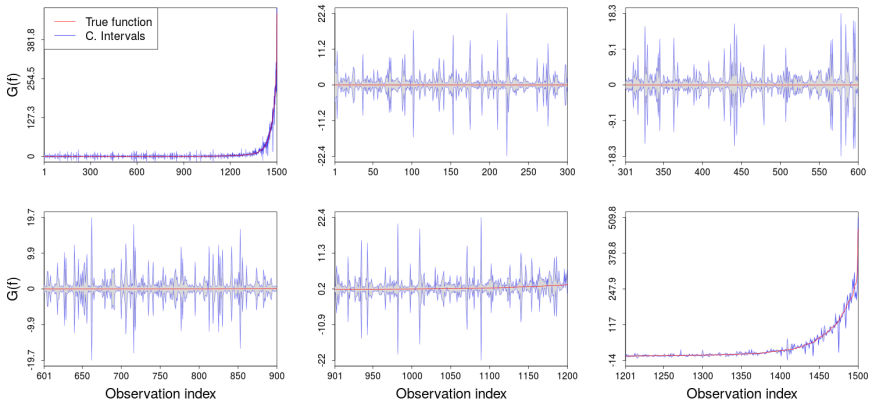
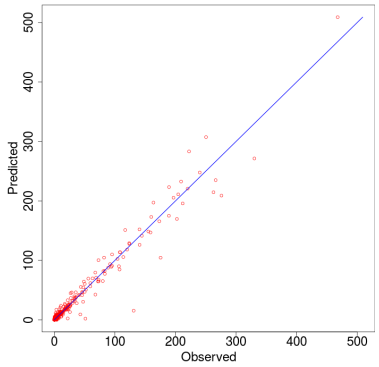


Figure 11: Coastal flooding case: results of the screening experiment. The abscissa denotes the experimental condition. Points are labeled by basis size p; the label 0 corresponds to the scalar metamodel.

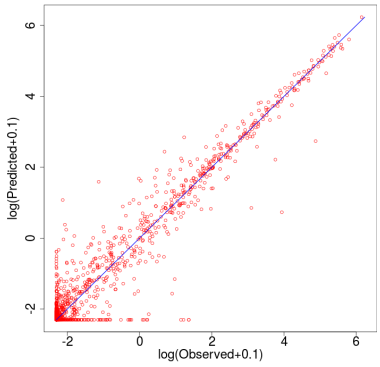
This time the scalar metamodel and the functional ones using the decomposition-based covariance performed similarly and outperformed almost every configuration based on the scalar covariance function. Here no trend of the RMSE regarding the projection size is clearly visible, at least for the best performing configurations. For the worst ones, it seems that a lower projection dimension p works better. Following the flowchart presented in Figure 3, we stop here and keep the current best configuration, as no interesting trend was detected. Strictly speaking, such a configuration corresponds to experimental condition number 16, which matches to a metamodel with an active B-spline representation of size $p = 1$ only for $\boldsymbol{Td}$, using the decomposition-based covariance function $\|\boldsymbol{f_1} - \boldsymbol{f_2}\|_{D,\boldsymbol{\theta}}$. However, from a practical perspective any other configuration using the decomposition-based covariance function could be also a good choice, since the performance of all that group of configurations was quite similar.



(a) Coastal flooding case: fitting of the best performing sample of best configuration. Left-top subplot illustrates the whole set of 1500 output observations in increasing order; this subplot is then subdivided to generate the remaining 5 subplots by splitting the abscissa into 5 sections and zooming the resulting figures.



(b) Coastal flooding case: calibration plot.



(c) Coastal flooding case: calibration plot in logarithmic scale.

Figure 12: Performance of best performing configuration for the coastal flooding case. (a) 1500 observations of the true output sorted in increasing order and corresponding 95% confidence intervals produced by the metamodel; (b) Prediction plot for 1500 data points for the best metamodel configuration.

The same type of plots built for the inspection of metamodel predictability in the analytic case, are used here (see Figure 12). For this sample, the proportion of observation points lying withing confidence intervals at 99%, 95% and 90% was 96%, 94% and 93%, respectively. In this case, the confidence intervals show more variability than in the analytic case. However, this is partially a visual effect since a significant part of the output series is almost constant here and thus, the span of the ordinates is determined by the width of the confidence intervals. As our input-output data set comes from hindcasts rather than a design of experiments, the distribution of the output seems to present a positive skew, and some outliers are visible (see Figure 12b). For this specific sample, it seems that the metamodel was able to make a reasonable prediction even for the largest outlier. Nonetheless, depending on the particular training and validation set used, this type of observations could be difficult to fit and significant errors

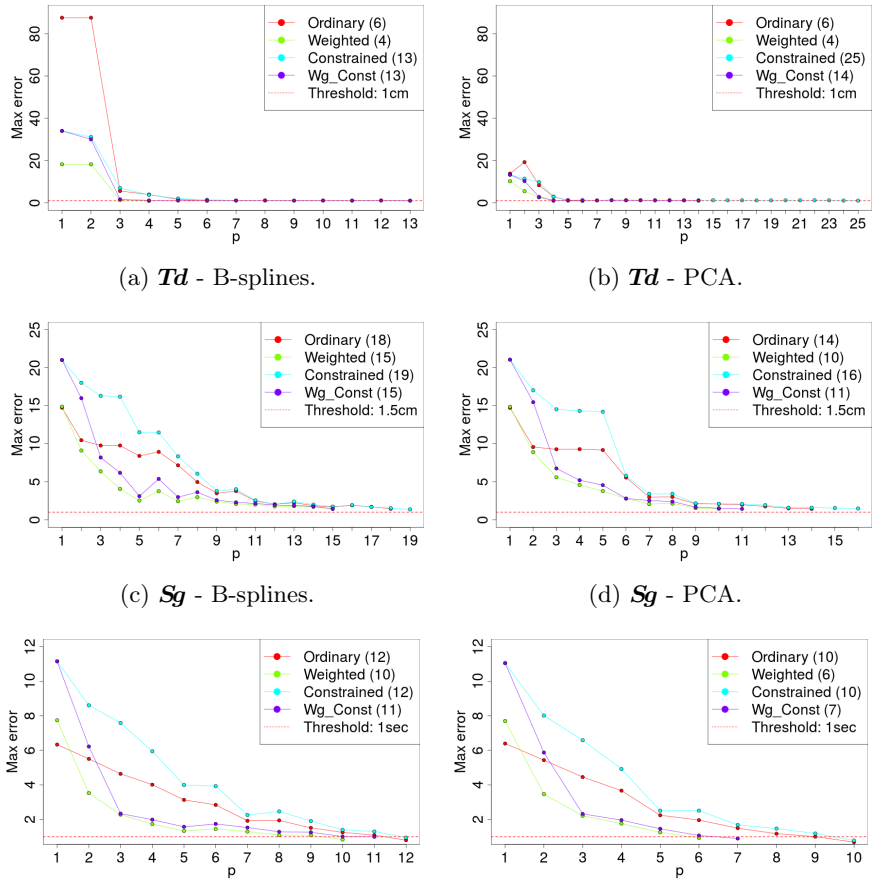
may be expected. As those extreme output values correspond to scenarios of great flooding, addressing such a difficulty should be a main priority in the development of metamodels for this application. That subject is out of the scope of the present study, but is currently under investigation.

5.2. Selecting the projection size

In Section 4, we briefly discussed the common practice in the literature to set up the projection dimension p, which is to select a number of basis for which the error of the projection istelf is relatively small. We also mentioned the weakness of this approach, which is that the projection dimension offering a good fit of the input does not necessarily lead to a better performance of the metamodel. In this section, we take advantage of the RISCOPE case study to illustrate such an inconsistency. To do so, we first set the p based on the projection error, and then we assess the performance of the corresponding metamodels.

5.2.1. Selection based on projection error

This first approach requires the definition of a maximum admisible error in the projection, that could be used as a stopping criterion when increasing the number of basis functions. Based on the advice of experts in coastal flooding, we set a maximum error within the critical time window $t = [13, 25]$, of 1 centimeter, 1.5 centimeters, and 1 second for the projections of ***Td***, ***Sg*** and ***Tp***, respectively. Such a time window corresponds to the moment of maximum tide $\pm$ 1 hour. The objective is to find the lowest number of basis functions, for which every curve of the data set satisfies the stated error tolerance. We also use this experiment to assess the results obtained by using the four least squares formulations presented in Section Appendix B, being the ordinary, weighted, constrained and weighted-constrained. The last three formulations could be interesting choices here as the points lying in the critical time window have outstanding relavance, specially the point t_{19} which matches the maximum ***Td***.



(a) ***Td*** - B-splines.

(b) ***Td*** - PCA.

(c) ***Sg*** - B-splines.

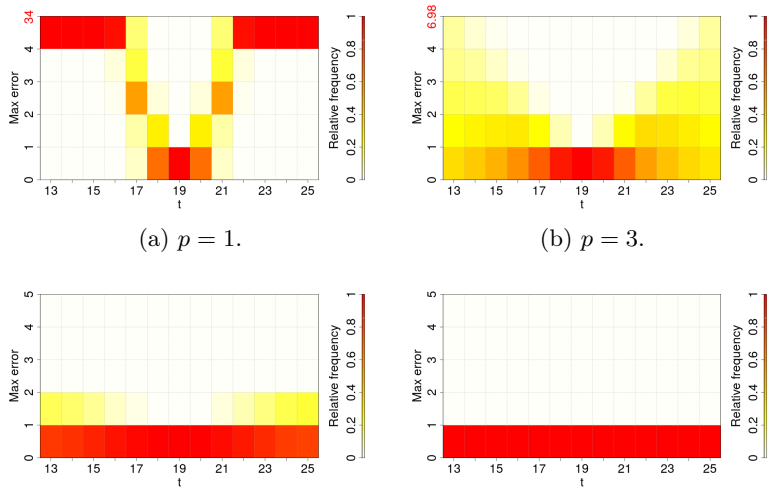
(d) ***Sg*** - PCA.

(e) ***Tp*** - B-splines.

(f) ***Tp*** - PCA.

Figure 13: Maximum projection error versus projection dimension p. ***Td*** and ***Sg*** error in centimeters and ***Tp*** error in seconds.

Figure 13 illustrates the maximum error within the critical time window, among the 20557 time series included in the data set, as a function of the basis size p for both the B-spline and PCA methods. Note that using a tolerance as the criterion to set the number of basis functions may end up in an unnecessarily large number of basis functions. For instance, the error of the ***Td*** projection with all formulations was already quite low at 6 basis functions, however, the constrained and weighted-constrained formulations required at least 13 basis functions to reach the tolerance. What makes it even worst is that usually, there are only a couple of curves in the data set that require such a high projection dimension. For instance, for B-splines projection of ***Td*** using the constrained formulation, almost all the curves reached the tolerance at $p = 6$ (see Figure 14). It means that the additional 7 basis functions required in that case where only used to get a small portion of the data set to fulfil the criterion.



(a) $p = 1$. (b) $p = 3$.

(c) $p = 5$. (d) $p = 6$.

Figure 14: Distribution of errors (in centimeters) for points in the critical window using constrained least quared optimization and B-splines projection for $\boldsymbol{Td}$ fitting.

Although the demanding constraint of perfect fitting at t_{19} has part on such behavior, the problem is also present in the ordinary and weighted formulations, which do not implement the constraint. See for instance the curve of the weighted formulation for $\boldsymbol{Tp}$ using the B-splines method in Figure 13. At $p = 5$ the error was considerably low, however, it required 10 basis functions to reach the tolerance.

5.2.2. Selection based on metamodel performance

The weighted formulation was the best performing one in the selection based on projection error. It required the lowest number of basis in all cases and its maximum error remained almost always below the error of all the other formulations. Therefore, in this section we assess the performance of the metamodel using the number of basis functions suggested by such formulation. Similarly to the previous experiments, here we also evaluate all the possible combinations of the cases where the functional decomposition of each input is inactive or active, both projection methods (B-splines and PCA) and the type of covariance function which can be scalar $\|\boldsymbol{f_1} - \boldsymbol{f_2}\|_{S,\boldsymbol{\theta}}$ or decomposition-based $\|\boldsymbol{f_1} - \boldsymbol{f_2}\|_{D,\boldsymbol{\theta}}$. However, in this case we do not evaluate the case where all functional decompositions are inactive, as it corresponds to the already evaluated scalar metamodel. In addition, the projection dimension p is no longer a factor of the experiment since its value is preset in the values listed in Table 1.

	Td	***Sg***	***Tp***
B-splines	4	15	10
PCA	4	10	6

Table 1: Selected dimension p based on projection error.

A total of 28 experimental conditions were evaluated. See Appendix D, Table D.6 for a detailed list of the experimental conditions and corresponding results of this experiment. Results are illustrated in Figure 15, which in turn includes results from the scalar metamodel and the configurations using the decomposition-based covariance function, as those were the best performing metamodels of the screening experiment (see Section 5.1).

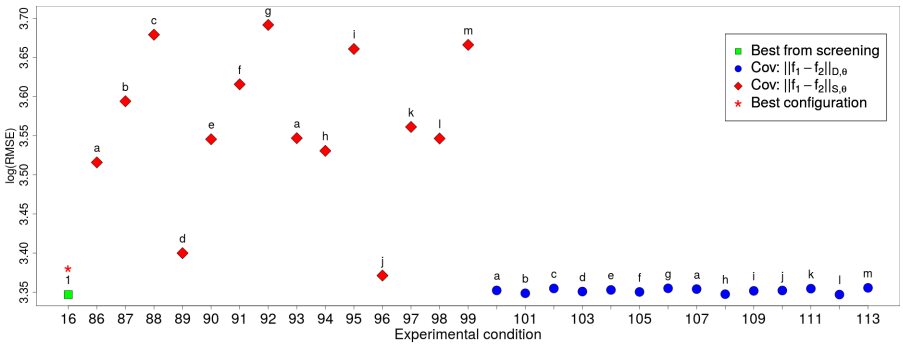


Figure 15: Coastal flooding case: performance of metamodels with the projection dimension p selected based on projection error. Points are labeled by basis size p; as the number of basis functions varies from one input to the other in the new experiment, those points are labeled using an alphabetic convenention where each letter is matched to a triple of integers denoting the projection dimension for each input in the order $\boldsymbol{Td}, \boldsymbol{Sg}, \boldsymbol{Tp}$: $a : (4, 0, 0)$, $b : (0, 15, 0)$, $c : (4, 15, 0)$, $d : (0, 0, 10)$, $e : (4, 0, 10)$, $f : (0, 15, 10)$, $g : (4, 15, 10)$, $h : (0, 10, 0)$, $i : (4, 10, 0)$, $j : (0, 0, 6)$, $k : (4, 0, 6)$, $l : (0, 10, 6)$, $m : (4, 10, 6)$.

Once again, results with the decomposition-based covariance function were better than those with the scalar one. However, none of the new configurations outperformed the best configuration found by means of our exploration methodology. Although the error of the best performing configurations in the new experiment is comparable to that of the scalar metamodel or the configurations with 1, 2 or 3 basis functions and decomposition-based covariance function, the new configurations are much more computationally demanding as they require larger data structures to be stored and processed. For instance, the average training time (over the 30 independent runs) of configuration 92 was 9.8 times the largest training time registered during the screening stage, corresponding to configuration 64. This makes those metamodels slower and more prone to numerical issues, which makes the large number of basis functions suggested by the selection based on error projection, not only unnecessary, but also problematic.

6. Conclusions

In this article we propose a methodology to explore diverse Gaussian process metamodeling approaches, while determining a good dimension to represent the functional inputs. The methodology starts with a screening stage, where several functional metamodels are compared with a scalar benchmark. This stage is intended to identify patterns showing dominant metamodel configurations, detect tendency in metamodel performance regarding numerical parameters as the projection dimension, and determine if the functional representation of the

inputs provides useful information to the metamodel or not. It is followed by a series of iterative experiments composing the cleaning and descent stages where the best known configurations are tested for new projection dimensions until a stationary point is reached. While relatively simple, the proposed methodology allows to find metamodel configurations of proven outstanding performance.

The exploration methodology was successfully applied on a theoretical example and also on a real life application to coastal flooding early warning. In both application cases, the best metamodel configuration found was able to accurately approximate the code. Results from the two applications corroborated that the best metamodeling settings (projection method, type of covariance function, projection dimension and combination of functional representation of inputs) depends on the particular application. Therefore, we conclude that those settings should be optimized each time a new metamodel is to be built in order to ensure good performance. The exploration methodology proposed here showed to be an effective tool for such optimization task.

One of the main advantages of the methodology is that it applies the principle of exploration-exploitation present in some classic heuristics such as Genetic Algorithm and Ant Colony Optimization. This principle allows to achieve diversification in the initial set of metamodel configurations evaluated and then leads the search to concentrate on the local improvement of the most attractive solutions found at each iteration. Considering that, the study of alternative exploration methods such as algorithms supported by the exploration-exploitation principle, could be an interesting topic of future research. In the same spirit, the construction of penalized likelihood and cross validation formulations for nonlinear models seems an interesting extension of the scalar-on-function techniques existing for linear regression (see e.g., [17]). The advantage of such a formulation would be the possibility of intrinsically setting the projection dimension for any metamodel configuration, which would considerably reduce the number of experimental conditions to test.

Through the coastal flooding application we showed how the traditional approach of setting the projection dimension based on the error of the projection itself may result in an unnecessarily large number of basis functions. In contrast, our methodology starts from the simpler metamodel configurations requiring a smaller basis, and explores more complex alternatives only if potential of improvement in that direction is found. Interestingly, we found that even when the inputs of the code were time series of dimension 37, it was possible to achieve quite good metamodel predictability just by using a scalar representation of them. On the other hand, denser metamodel configurations found by the traditional approach offered no improvement in performance, but significantly larger training time (up to 9.8 times larger than the worst training time using our methodology). This result may be explained by the fact that the simplified code considered here can be seen as the sum of independent scalar-to-scalar problems. Thus, if we represent the functional inputs by a scalar associated to a key time instant in the evolution of the output, we may expect to have reasonable predictions. We would not expect this kind of result if the code is intrinsically functional as in the analytic case studied in Section 4.1, where the best configuration found was a metamodel using a projection of the inputs in a basis of dimension 5.

As the two case studies presented here consider a discretized representation of the functional inputs, it seems interesting to assess alternative distances adapted

to time series [26] or try to adapt the work of [39] where a Geodesic PCA for density functions is introduced.

Acknowledgements

This research was undertaken within the RISCOPE project (ANR, project No.16CE04-0011, https://perso.math.univ-toulouse.fr/riscope/). The authors gratefully acknowledge the data providers (Ifremer, LOPS, NOAA, Liens ; see Table Appendix A). We also thank Xavier Bertin for dedicated running of wave model for the Sonel-wave dataset. Sylvestre Le Roy and Camille André are also acknowledged for providing the bahymetric data used to set up the cross-shore numerical model.

References

[1] Rohmer J, Idier D. A meta-modelling strategy to identify the critical offshore conditions for coastal flooding. Natural Hazards and Earth System Sciences. 2012;12(9):2943–2955.

[2] Jia G, Taflanidis AA. Kriging metamodeling for approximation of high-dimensional wave and surge responses in real-time storm/hurricane risk assessment. Computer Methods in Applied Mechanics and Engineering. 2013;261:24–38.

[3] Rueda A, Gouldby B, Méndez F, Tomás A, Losada I, Lara J, et al. The use of wave propagation and reduced complexity inundation models and metamodels for coastal flood risk assessment. Journal of Flood Risk Management. 2016;9(4):390–401.

[4] Muehlenstaedt T, Fruth J, Roustant O. Computer experiments with functional inputs and scalar outputs by a norm-based approach. Statistics and Computing. 2017;27(4):1083–1097.

[5] Nanty S, Helbert C, Marrel A, Pérot N, Prieur C. Sampling, metamodeling, and sensitivity analysis of numerical simulators with functional stochastic inputs. SIAM/ASA Journal on Uncertainty Quantification. 2016;4(1):636–659.

[6] Simpson T, Mistree F, Korte J, Mauery T. Comparison of response surface and kriging models for multidisciplinary design optimization. In: 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization; 1998. p. 4755.

[7] Forrester A, Sobester A, Keane A. Engineering design via surrogate modelling: a practical guide. John Wiley & Sons; 2008.

[8] Santner TJ, Williams BJ, Notz WI. The design and analysis of computer experiments. Springer Science & Business Media; 2013.

[9] Lataniotis C, Marelli S, Sudret B. The Gaussian process modelling module in UQLab. Journal of Soft Computing in Civil Engineering. 2018;.

[10] Sacks J, Welch WJ, Mitchell TJ, Wynn HP. Design and analysis of computer experiments. Statistical science. 1989;p. 409–423.

[11] Oakley J, O'hagan A. Bayesian inference for the uncertainty distribution of computer model outputs. Biometrika. 2002;89(4):769–784.

[12] Rasmussen CE, Williams CK. Gaussian processes for machine learning. 2006. The MIT Press, Cambridge, MA, USA. 2006;38:715–719.

[13] Ramsay JO, Silverman BW. Applied functional data analysis: methods and case studies. Springer; 2007.

[14] Marrel A, Iooss B, Jullien M, Laurent B, Volkova E. Global sensitivity analysis for models with spatially dependent outputs. Environmetrics. 2011;22(3):383–397.

[15] Nanty S, Helbert C, Marrel A, Pérot N, Prieur C. Uncertainty quantification for functional dependent random variables. Computational Statistics. 2017;32(2):559–583.

[16] De Boor C, De Boor C, Mathématicien EU, De Boor C, De Boor C. A practical guide to splines. vol. 27. Springer-Verlag New York; 1978.

[17] Reiss PT, Goldsmith J, Shang HL, Ogden RT. Methods for Scalar-on-Function Regression. International Statistical Review. 2017;85(2):228–249.

[18] Antoniadis A, Helbert C, Prieur C, Viry L. Spatio-temporal metamodeling for West African monsoon. Environmetrics. 2012;23(1):24–36.

[19] Rohmer J. Boosting Kernel-Based Dimension Reduction for Jointly Propagating Spatial Variability and Parameter Uncertainty in Long-Running Flow Simulators. Mathematical Geosciences. 2015;47(2):227–246.

[20] ANR RISCOPE Project;. Accessed: 2018-12-04. `https://perso.math.univ-toulouse.fr/riscope/`.

[21] Booij N, Holthuijsen L, Ris R. The" SWAN" wave model for shallow water. In: Coastal Engineering 1996; 1997. p. 668–676.

[22] EurOtop I. Manual on Wave Overtopping of Sea Defences and Related Structures. An Overtopping Manual Largely Based on European Research, But for Worldwide Application; 2016. `http://www.overtopping-manual.com/`.

[23] Moustapha M, Sudret B, Bourinet JM, Guillaume B. Quantile-based optimization under uncertainties using adaptive Kriging surrogate models. Structural and multidisciplinary optimization. 2016;54(6):1403–1421.

[24] Stein ML. Interpolation of spatial data: some theory for kriging. Springer Science & Business Media; 2012.

[25] Abrahamsen P. A review of Gaussian random fields and correlation functions. Norsk Regnesentral/Norwegian Computing Center Oslo; 1997.

[26] Mori U, Mendiburu A, Lozano JA. Distance measures for time series in R: The TSdist package. R Journal. 2016;8(2):451–459.

[27] Ginsbourger D, Rosspopoff B, Pirot G, Durrande N, Renard P. Distance-based kriging relying on proxy simulations for inverse conditioning. Advances in water resources. 2013;52:275–291.

[28] De Boor C. A practical guide to splines, revised Edition, Vol. 27 of Applied Mathematical Sciences. Mechanical Sciences, year. 2001;.

[29] Jolliffe I. Principal component analysis. In: International encyclopedia of statistical science. Springer; 2011. p. 1094–1096.

[30] Morris MD. Gaussian surrogates for computer models with time-varying inputs and outputs. Technometrics. 2012;54(1):42–50.

[31] Friedman J, Hastie T, Tibshirani R. The elements of statistical learning. vol. 1. Springer series in statistics New York, NY, USA:; 2001.

[32] Pulido HG, De La Vara Salazar R, González PG, Martínez CT, Pérez MdCT. Análisis y diseño de experimentos. McGraw-Hill; 2012.

[33] Montgomery DC. Design and analysis of experiments. John wiley & sons; 2017.

[34] McKay MD, Beckman RJ, Conover WJ. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics. 1979;21(2):239–245.

[35] Marrel A, Iooss B, Van Dorpe F, Volkova E. An efficient methodology for modeling complex computer codes with Gaussian processes. Computational Statistics & Data Analysis. 2008;52(10):4731–4744.

[36] Roustant O, Ginsbourger D, Deville Y. DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization. 2012;.

[37] Hamilton JD. Time series analysis. vol. 2. Princeton university press Princeton, NJ; 1994.

[38] Dixit B. Elasticsearch Essentials. 2nd ed. Birmingham: Packt Publishing Ltd; 2016.

[39] Bigot J, Gouet R, Klein T, López A. Geodesic PCA in the Wasserstein space by convex PCA. Ann Inst Henri Poincaré Probab Stat. 2017;53(1):1–26. Available from: `https://doi.org/10.1214/15-AIHP706`.

[40] Carrere L, Lyard F, Cancet M, Guillot A, Picot N. FES 2014, a new tidal model-Validation results and perspectives for improvements. In: Proceedings of the ESA Living Planet Symposium; 2016. .

[41] Compo G, Whitaker J, Sardeshmukh P, Matsui N, Allan R, Yin X, et al.. NOAA/CIRES Twentieth Century Global Reanalysis Version 2c. Research Data Archive at the National Center for Atmospheric Research; 2015, updated yearly. Accessed: 28 feb 2017. `https://doi.org/10.5065/D6N877TW`.

[42] Dee D, Balmaseda M, Balsamo G, Engelen R, Simmons A, Thépaut JN. Toward a consistent reanalysis of the climate system. Bulletin of the American Meteorological Society. 2014;95(8):1235–1248.

[43] Muller H, Pineau-Guillou L, Idier D, Ardhuin F. Atmospheric storm surge modeling methodology along the French (Atlantic and English Channel) coast. Ocean Dynamics. 2014;64(11):1671–1692.

[44] Bertin X, Prouteau E, Letetrel C. A significant increase in wave height in the North Atlantic Ocean over the 20th century. Global and Planetary Change. 2013;106:77–83.

[45] Charles E, Idier D, Thiébot J, Le Cozannet G, Pedreros R, Ardhuin F, et al. Wave climate variability and trends in the Bay of Biscay from 1958 to 2001. Journal of Climate doi. 2012;10.

[46] Boudière E, Maisondieu C, Ardhuin F, Accensi M, Pineau-Guillou L, Lepesqueur J. A suitable metocean hindcast database for the design of Marine energy converters. International Journal of Marine Energy. 2013;3:e40–e52.

[47] Bertsekas DP. Nonlinear programming. Athena scientific Belmont; 1999.

Input	Period	Initial time step	Source			
			Name	**Provider**	**Reference**	**Website**
Tide	1900-2016	No constrained. Set at 10min.	FES2014	LEGOS	[40]	https://www.aviso.altimetry.fr/en/data/products/auxiliary-products/global-tide-fes/description-fes2014.html
Surge	1900-1978	6h	20CR (sea surface pressure)	NOAA	[41]	https://reanalyses.org/atmosphere/overview-current-atmospheric-reanalyses\#TWENTv2c
	1979-2005	1h	CFSR (sea surface pressure)	NOAA	[42]	https://climatedataguide.ucar.edu/climate-data/climate-forecast-system-reanalysis-cfsr
	2006-2016	15min	MARC	Ifremer and LOPS	[43]	http://marc.ifremer.fr/
Hs, Tp	1900-1957	6h	Sonel (waves)	Liens	[44]	http://www.sonel.org/-Waves-.html?lang=en
	1958-09/2002	1h	BoBWA	BRGM	[45]	http://bobwa.brgm.fr/
	10/2002-2007	1h	Homere	Ifremer and LOPS	[46]	http://marc.ifremer.fr/en/produits/rejeu_d_etats_de_mer_homere
	2008-2016	1h	Iowaga/Norgasug	Ifremer and LOPS	[46]	https://wwz.ifremer.fr/iowaga/Products

Table A.2: Sources of data for the coastal flooding application case.

Appendix B. Setting the coefficients for the functional basis

Here we discuss the calibration of the coefficients $\alpha_1, \ldots, \alpha_p$ of the decomposition. This task can be performed by means of standard least squares optimization. Closed form solutions for four variations of the optimization problem are provided below. For the sake of exposition, let us assume first that a single realization of the functional input wants to be projected.

- Weighted Least Squares (WLS)

  Let $\boldsymbol{f} = (f(t_1), \ldots, f(t_T))^\top$ be a column vector with the true values of a functional input $f \in \mathbb{R}$, observed at time instants $\boldsymbol{t} = \{t_1, \ldots, t_T\}$. Consider the functional decomposition $\Pi_p(f)$ of dimension p defined in (11). Then, let $\boldsymbol{B}$ denote the $T \times p$ matrix containing the values of the p basis functions evaluated at $\boldsymbol{t}$, and let $\boldsymbol{\alpha}$ denote the p-dimensional vector of coefficients for the decomposition. Consider a case where some parts of the domain of f are more relevant than others. This can be considered in the selection of the expansion coefficients by assigning a weight w_t to each one of the errors of the projection. To this end, let us consider a diagonal weight matrix $\boldsymbol{W}$ of dimension $T \times T$. Let now

$$\begin{aligned} \mathrm{SSR}_w : \quad & \mathbb{R}^p \to \mathbb{R}, \\ & \boldsymbol{\alpha} \to (\boldsymbol{f} - \boldsymbol{B}\boldsymbol{\alpha})' \boldsymbol{W} (\boldsymbol{f} - \boldsymbol{B}\boldsymbol{\alpha}), \end{aligned} \tag{B.1}$$

  denote the weighted sum of squared residuals of the fit of $\boldsymbol{f}$ by means of the functional decomposition $\boldsymbol{B}\boldsymbol{\alpha}$. The weighted formulation can be then writen as:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^p} \mathrm{SSR}_w. \tag{B.2}$$

  The solution by derivatives yields:

$$\hat{\boldsymbol{\alpha}} = (\boldsymbol{B}'\boldsymbol{W}\boldsymbol{B})^{-1}\boldsymbol{B}'\boldsymbol{W}\boldsymbol{f}. \tag{B.3}$$

  This formulation is also useful for instance, when observations of the function at different points of its domain are not equally reliable.

- Ordinary Least Squares (OLS)

  In the particular case where $\boldsymbol{W}$ is the identity matrix of dimension $T \times T$ (i.e., all points in the domain of f are equaly important), (B.3) becomes

$$\hat{\boldsymbol{\alpha}} = (\boldsymbol{B}'\boldsymbol{B})^{-1}\boldsymbol{B}'\boldsymbol{f}. \tag{B.4}$$

  Which corresponds to the classic normal equations for ordinary least squares in linear regression problems.

- Weighted-constrained Least Squares (WCLS)

  If there are multiple critical points in the domain of f and there is also a particular point $t* \in \boldsymbol{t}$ of outstanding importance, the weighted-constrained formulation can be applied. It allows to enforce the projection to interpolate exactly the true function at the main critical point $t*$, while assigning relatively large weights to the remaining critical points. Note that additional constraints could be added in order to warrant a good fitting at multiple critical points, however, the more constraints we add, the more prone we are to leave no feasible solution.

  Let us denote $\boldsymbol{b'_{t*}}$ the $t*$-th row of the matrix $\boldsymbol{B}$, corresponding to the values of the p basis functions at the critical point $t*$. Similarly, let $f_{t*} = f(t*)$ denote the value of the functional input at the critical point $t*$. We can write the weighted constrained optimization problem as:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^p} \ \mathrm{SSR}_w; \tag{B.5}$$

$$s.t. \qquad \boldsymbol{b'_{t*}}\boldsymbol{\alpha} - f_{t*} = 0. \tag{B.6}$$

  Here we use the well known method of Lagrange multipliers [47] which allows to include equality constraints as part of the objective function in order to solve the problem by derivatives. The Lagrange function for this problem can be writen as

$$\mathcal{L}(\boldsymbol{\alpha}, \lambda) = \mathrm{SSR}_w + \lambda(\boldsymbol{b'_{t*}}\boldsymbol{\alpha} - f_{t*}), \tag{B.7}$$

  with $\lambda \in \mathbb{R}$, playing the role of Lagrange multiplier.

  The solution in this case is given by

$$\hat{\boldsymbol{\alpha}} = (\boldsymbol{B}'\boldsymbol{W}\boldsymbol{B})^{-1}\left(\boldsymbol{B}'\boldsymbol{W}\boldsymbol{f} - \frac{1}{2}\boldsymbol{b_{t*}}\hat{\lambda}\right), \tag{B.8}$$

  with

$$\hat{\lambda} = \frac{2\left[\boldsymbol{b'_{t*}}(\boldsymbol{B}'\boldsymbol{W}\boldsymbol{B})^{-1}\boldsymbol{B}\boldsymbol{W}\boldsymbol{f} - f_{t*}\right]}{\boldsymbol{b'_{t*}}(\boldsymbol{B}'\boldsymbol{W}\boldsymbol{B})^{-1}\boldsymbol{b_{t*}}}. \tag{B.9}$$

- Constrained Least Squares (CLS)

  Similarly to the ordinary least squares, the constrained formulation is a particular case of the weighted-constrained one where $\boldsymbol{W}$ is the identity matrix of dimension $T \times T$. In that case, (B.8) becomes

$$\hat{\boldsymbol{\alpha}} = (\boldsymbol{B}'\boldsymbol{B})^{-1}\left(\boldsymbol{B}'\boldsymbol{f} - \frac{1}{2}\boldsymbol{b_{t*}}\hat{\lambda}\right), \tag{B.10}$$

and (B.9) becomes

$$\hat{\lambda} = \frac{2\left[\boldsymbol{b}'_{\boldsymbol{t}*}(\boldsymbol{B}'\boldsymbol{B})^{-1}\boldsymbol{B}\boldsymbol{f} - f_{t*}\right]}{\boldsymbol{b}'_{\boldsymbol{t}*}(\boldsymbol{B}'\boldsymbol{B})^{-1}\boldsymbol{b}_{\boldsymbol{t}*}}. \tag{B.11}$$

The least squares formulations provided above arise from the basic case where a single realization of the functional input needs to be projected. However, we remark that closed form solutions provided by Equations (B.3), (B.4), (B.8), (B.9), (B.10) and (B.11), also work as vectorized expressions for multiple simultaneous projections (i.e., they do not require loops in code if matrix oriented coding environments like R or Matlab are used). To that end, the column vector $\boldsymbol{f}$ should be replaced by a $T \times n$ matrix $\boldsymbol{F}$ with n observations of the functional input. Correspondingly, the output of the optimization would be a $p \times n$ matrix containing n p-dimensional vectors of expansion coefficients α_k for the decomposition of the n observations of the function.

Appendix C. Experimental conditions and results for analytic case

Conf.	Functional input		Projection method	Covariance function	Basis size	Results		
	f1	f2				log(RMSE)	CPU time (sec) Train	CPU time (sec) Pred
1	0	0	-	-	-	12.6	14.2	2.0
2	1	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	12.4	23.2	2.0
3	0	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	12.3	15.6	2.0
4	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	12.1	17.6	2.0
5	1	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	12.0	18.6	2.0
6	0	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	11.9	19.0	2.0
7	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	10.0	53.5	2.1
8	1	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	12.6	18.7	2.6
9	0	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	12.4	18.1	2.6
10	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	12.3	21.8	2.6
11	1	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	12.3	25.9	2.6
12	0	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	12.2	23.3	2.6
13	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	10.3	86.1	2.7
14	1	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	12.0	17.3	2.2
15	0	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	11.8	16.9	2.2
16	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	8.9	32.4	2.3
17	1	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	11.8	37.7	2.3
18	0	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	11.7	38.8	2.3
19	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	9.7	53.9	2.4
20	1	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	12.2	20.9	2.6
21	0	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	12.0	20.9	2.6
22	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	9.2	49.5	2.9
23	1	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	12.1	33.4	2.6
24	0	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	11.8	36.2	2.6
25	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	9.9	89.5	2.9
26	1	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	11.8	56.6	2.4
27	0	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	11.7	59.5	2.4
28	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	9.1	27.6	2.4
29	1	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	11.8	95.7	2.3
30	0	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	11.6	91.7	2.1
31	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	9.1	35.3	2.2
32	1	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	11.9	33.0	2.9
33	0	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	11.8	34.6	2.8
34	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	8.8	45.9	3.1
35	1	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	11.8	43.5	2.8
36	0	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	11.7	44.2	2.8
37	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	8.6	42.4	3.1

Table C.3: Analytic case: experimental conditions and results from screening stage. For training and prediction time, the value displayed is the average over 30 runs using independent training and validation sets of size $n = 800$ and $n_* = 1500$, respectively. In the case of the error, the value displayed is the natural logarithm of the average RMSE over the 30 runs. For the functional input, 1 denotes active and 0 denotes inactive.

Conf.	Functional input		Projection method	Covariance function	Basis size	Results		
	f1	f2				log(RMSE)	CPU time (sec)	
							Train	Pred
38	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	4	9.5	43.2	2.2
39	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	4	9.5	54.9	2.3
40	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	4	8.6	36.9	3.3
41	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	4	8.6	49.0	3.3
42	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	5	9.9	52.8	2.5
43	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	5	9.7	211.9	2.8
44	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	5	8.6	34.2	3.5
45	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	5	8.6	48.3	3.5
46	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	6	9.9	62.1	2.6
47	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	6	9.9	365.3	2.8
48	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	6	8.6	34.0	3.7
49	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	6	8.6	43.0	3.7
50	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	7	8.6	36.0	3.8
51	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	7	8.6	41.9	3.8
52	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	8	8.6	35.6	3.9
53	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	8	8.6	43.1	4.0

Table C.4: Analytic case: experimental conditions and results from cleaning and descent stages. The values of log(RMSE), training time and prediction time are computed as described in Table C.3. For the functional input, 1 denotes active and 0 denotes inactive.

Appendix D. Experimental conditions and results for coastal flooding case

Conf.	Functional input			Projection method	Covariance function	Basis size	Results		
	Td	Sg	Tp				log(RMSE)	CPU time (sec)	
								Train	Pred
1	0	0	0	-	-	-	3.35	27.7	2.0
2	1	0	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	3.37	32.0	2.0
3	0	1	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	3.38	31.2	2.1
4	1	0	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	3.41	36.3	2.1
5	0	0	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	3.35	29.0	2.0
6	1	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	3.38	34.8	2.1
7	0	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	3.38	33.9	2.1
8	1	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	3.41	42.6	2.1
9	1	0	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	3.38	29.1	2.1
10	0	1	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	3.40	29.1	2.1
11	1	1	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	3.43	30.6	2.1
12	0	0	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	3.37	36.7	2.1
13	1	0	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	3.40	40.2	2.1
14	0	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	3.41	39.3	2.1
15	1	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	3.45	45.3	2.1
16	1	0	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	3.35	35.8	2.7
17	0	1	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	3.35	34.3	2.6
18	1	1	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	3.35	38.5	2.8
19	0	0	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	3.35	48.8	2.7
20	1	0	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	3.35	61.9	2.8
21	0	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	3.35	57.0	2.7
22	1	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	3.35	62.5	2.8
23	1	0	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	3.35	33.5	2.6
24	0	1	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	3.35	33.4	2.6
25	1	1	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	3.35	37.7	2.7
26	0	0	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	3.35	52.0	2.6
27	1	0	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	3.35	62.4	2.7
28	0	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	3.36	59.1	2.7
29	1	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	3.36	64.9	2.8
30	1	0	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	3.46	32.0	2.3
31	0	1	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	3.42	33.2	2.3
32	1	0	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	3.53	42.3	2.4
33	0	0	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	3.38	37.8	2.3
34	1	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	3.48	48.1	2.4
35	0	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	3.44	48.3	2.4
36	1	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	3.54	67.0	2.6
37	1	0	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	3.46	31.4	2.3
38	0	1	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	3.42	34.0	2.3
39	1	1	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	3.53	44.4	2.4
40	0	0	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	3.37	42.4	2.3
41	1	0	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	3.48	60.5	2.5
42	0	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	3.44	61.3	2.5
43	1	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	3.54	95.1	2.6
44	1	0	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	3.36	33.9	2.7
45	0	1	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	3.35	35.7	2.7

Continued on next page

Table D.5 – *Continued from previous page*

46	1	1	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	3.36	38.4	2.8
47	0	0	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	3.35	55.0	2.8
48	1	0	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	3.36	60.8	2.9
49	0	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	3.36	61.2	2.9
50	1	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	3.36	65.8	3.0
51	1	0	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	3.35	33.2	2.7
52	0	1	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	3.35	37.2	2.7
53	1	1	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	3.36	35.5	2.8
54	0	0	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	3.35	52.0	2.7
55	1	0	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	3.36	58.7	2.9
56	0	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	3.36	59.5	2.9
57	1	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	3.36	63.0	3.0
58	1	0	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	3.44	36.3	2.3
59	0	1	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	3.40	35.9	3.4
60	1	0	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	3.50	50.7	2.5
61	0	0	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	3.38	53.8	2.4
62	1	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	3.46	84.2	2.6
63	0	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	3.44	81.1	2.6
64	1	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	3.52	130.6	2.7
65	1	0	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	3.51	38.0	2.4
66	0	1	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	3.42	37.0	2.4
67	1	1	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	3.57	55.8	2.6
68	0	0	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	3.38	56.5	2.4
69	1	0	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	3.53	122.9	2.6
70	0	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	3.44	94.4	2.6
71	1	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	3.58	182.7	2.7
72	1	0	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	3.35	33.9	2.7
73	0	1	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	3.36	36.1	2.8
74	1	1	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	3.36	37.5	2.9
75	0	0	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	3.35	53.3	2.8
76	1	0	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	3.35	67.7	3.0
77	0	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	3.36	60.7	3.0
78	1	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	3.36	65.8	3.2
79	1	0	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	3.35	31.5	2.7
80	0	1	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	3.35	36.9	2.7
81	1	1	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	3.35	35.7	2.9
82	0	0	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	3.35	52.4	2.8
83	1	0	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	3.35	58.0	3.0
84	0	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	3.35	60.0	3.0
85	1	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	3.36	63.3	3.2

Table D.5: Coastal flooding case: experimental conditions and results from screenig stage. The values of log(RMSE), training time and prediction time are computed as described in Table C.3. For the functional input, 1 denotes active and 0 denotes inactive.

Run	Basis size			Projection method	Covariance function	Results		
	Td	Sg	Tp			log(RMSE)	CPU time (sec) Train	CPU time (sec) Pred
86	4	0	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3.52	40.6	2.2
87	0	15	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3.59	285.5	2.6
88	4	15	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3.68	400.9	2.8
89	0	0	10	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3.40	261.4	2.4
90	4	0	10	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3.55	420.4	2.6
91	0	15	10	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3.62	1055.3	3.0
92	4	15	10	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3.69	1280.8	3.2
93	4	0	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3.55	44.9	2.1
94	0	10	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3.53	71.4	2.4
95	4	10	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3.66	120.5	2.6
96	0	0	6	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3.37	101.1	2.3
97	4	0	6	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3.56	265.6	2.4
98	0	10	6	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3.55	363.4	2.6
99	4	10	6	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3.67	674.1	3.1
100	4	0	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3.35	35.0	2.8
101	0	15	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3.35	50.0	3.6
102	4	15	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3.35	54.1	3.7
103	0	0	10	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3.35	61.4	3.3
104	4	0	10	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3.35	65.7	3.5
105	0	15	10	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3.35	85.3	4.1
106	4	15	10	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3.36	98.2	4.3
107	4	0	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3.35	35.6	2.8
108	0	10	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3.35	32.8	3.2
109	4	10	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3.35	34.8	3.4
110	0	0	6	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3.35	53.5	2.6
111	4	0	6	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3.35	60.3	2.9
112	0	10	6	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3.35	59.1	3.2
113	4	10	6	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3.36	67.5	3.4

Table D.6: Coastal flooding case: evaluation of projection dimension selected based on projection error. The values of log(RMSE), training time and prediction time are computed as described in Table C.3.