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

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

José Betancourt[a,b,*], François Bachoc[a], Thierry Klein[a,b], Déborah 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

This paper investigates the construction of a metamodel for coastal flooding early warning at the peninsula of Gâvres, France. The code under study is an hydrodynamic model which receives time-varying maritime conditions as inputs. We concentrate on Gaussian pocess metamodels to emulate the behavior of the code. To model the inputs we make a projection of them onto a space of lower dimension. This setting gives rise to a model selection methodology which we use to calibrate four characteristics of our functional-input metamodel: (i) the family of basis functions to project the inputs; (ii) the projection dimension; (iii) the distance to measure similarity between functional input points; and (iv) the set of functional predictors to keep active. The proposed methodology seeks to optimize these parameters for metamodel predictability, at an affordable computational cost. A comparison to a dimensionality reduction approach based on the projection error of the input functions only showed that the latter may lead to unnecessarily large projection dimensions. We also assessed the adaptability of our methodology to changes in the number of training and validation points. The methodology proved its robustness by finding the optimal solution for most of the instances, while being computationally efficient.

Keywords: Dimensionality reduction, Gaussian process, Metamodeling, Functional inputs, Computer experiments

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, they

*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 (Déborah Idier), r.pedreros@brgm.fr (Rodrigo Pedreros), j.rohmer@brgm.fr (Jérémy Rohmer)

provide a solution for cases when the real system is a natural process (e.g., volcanic activity) and some input-output conditions are rarely observed. 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 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]. This difficulty is usually resolved by creating quick-to-evaluate mathematical emulators of those numerical codes, based on a limited collection of runs [6, 7, 8]; such emulators are often called *surrogate models* or *metamodels*. In this paper, we illustrate an intermediate step in the development of a surrogate model of a complex hydrodynamic code used in the context of early warning for coastal flooding hazards. This work is said to be an intermediate step as the target hydrodynamic code to emulate is still under calibration. In the meantime, we use a simplified fast-running version of it which allows us to study the dynamics of the system and the specificities of the metamodeling task at hand.

The simplified hydrodynamic code receives four inputs and delivers a single output, all of them functions of time. As usual in practice, each function is supplied to and delivered by the code in the format of a time series represented by a long vector. Even though, we keep referring to them as *functional inputs* (resp. *functional output*) since the notions of order and/or proximity between time points hold for them. The focus of this article is on the modeling of functional inputs. Thus, we keep their full complexity into account, but we reduce the problem by only considering a scalar representation of the output, which corresponds to the cumulative sum of its values over time.

The main difficutly of functional-input regression is the large number of predictors that one may end up dealing with. In our coastal flooding application, for instance, each input variable is a time series with 37 time points. Some applications involve inputs with more then 1000 time points (see e.g., [5]). Such a large number of covariates naturally hampers the tractability and processing speed of the metamodel while making it more prone to overfitting. A common approach to overcome this problem is to make a projection of each functional input onto a space of lower dimension while preserving the main statistical or geometric properties of the variable [9, 10, 5, 4]. The projection is sometimes preceeded by time warping if an input shows a cyclical pattern which is consistent among functions [11]. A suitable basis for the projection space may come from a variety of families, including B-splines, PCA, Legendre polynomials, wavelets, Fourier and many others. The ideal basis family seems to vary from one application to the other. However, most studies set this feature a priori, leaving wide margin for potential improvement of the metamodel.

The approach based on the projection of the inputs also requires the selection of the projection dimension. Seeking for a balance between speed/tractability and prediction quality, the goal should be to set the new dimension considerably lower than the original one, but still sufficiently large to allow for good predictions. Thus, for forecast purposes, dimensionality reduction of the inputs should be primarily leaded by metamodel predictability. However, the new dimension p is often chosen to retain certain amount of information on the input. For

instance, so that most information on its variability is concentrated on the first p components [5] or by minimizing the projection error [12]. Some alternative techniques better incorporate the idea of focusing on metamodel predictability. These include *scalar-on-function regression* [13, 14, 15], methods in the field of *active subspaces* [16], as well as stack models composed of a dimensionality reduction and a metamodeling technique put together and trained using backpropagation [17, 18, 19, 20]. Despite the advantages of these methods in terms of simplicity or predictablity, their application in this paper is prevented by a set of factors. First of all, developments of scalar-on-function regression are mainly related to the linear regression framework, whose scope is exceeded by the complexity of the coastal flooding phenomenon. Secondly, active subspaces techniques often rely on the gradient of the output w.r.t the inputs. This information is rarely available and has to be approximated from data [21], a sometimes difficult task when inputs are structured objects such as time series or spatial fields [22]. Finally, techniques relying on stacked models turn out to be quite restrictive regarding the combination of components of the stack; most of the proposals are limited to one specific combination of dimensionality reduction and metamodeling technique. Rather than restricting oneself to some particular dimensionality reduction method, this paper aims to define a way to explore and select among available alternatives.

Among all metamodel-based solutions (polynomials, splines, neural networks, etc.), we focus on Gaussian processes [23, 24, 25]. 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]. The essence of extending Gaussian process models to receive functional inputs lies in the adaption of the distance used to measure similarity/proximity between pairs of input points. In the case of scalar inputs, the standard is to use a weighted Euclidean distance where each input variable is assigned a weight [25]. These weights are then optimized, typically through the Maximum Likelihood or Cross Validation method [26]. When dealing with functional inputs, the selection of the distance will strongly depend on the way of representing the inputs in the metamodel. For projections, a popular approach is to use the projection coefficients as individual scalar inputs of the model and proceed as described before [5]. In that case, each projection coefficient would be assigned a weight. Another alternative is to aknowledge the fact that certain coefficients correspond to the same input variable. Then, each set of projection coefficients, corresponding to the same input variable, would be assigned a single weight [4]. These two and any other suitable norm are valid choices and once again, the best option will likely depend on the application.

The preceding discussion addresses some of the specificities of functional-input metamodeling; the last paragraph making emphasis on Gaussian processes which are the type of metamodel studied here. In line with that discussion, our main contribution is a methodology to simultaneously tune multiple characteristics of a functional-input metamodel. Here we use it to calibrate: (i) the family of basis functions to project the functional inputs; (ii) the projection dimension; (iii) the distance function to measure similarity between functional input points; and (iv) the set of functional predictors to keep active. As mentioned

earlier, these types of metamodeling choices, herein called *structural parameters* of the metamodel, are often fixed arbitrarily or based on results from other applications. However, as will be shown in this paper through a set of computer experiments, 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 from it [22]. Our proposal is a staged exploration strategy which optimizes the set of structural parameters for metamodel predictability. Although relatively 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 coastal flooding application case that motivates this study. The set of technical details concerning the modeling of functional inputs within Gaussian process metamodels are provided in Section 3. Section 4 describes the exploration approach proposed here to calibrate the structural parameters of the metamodel. This section also presents an analytic case study to illustrate the methodology. In Section 5, we apply the exploration strategy to setup the metamodel for the coastal flooding application. In Section 6, we conduct an experiment to assess the robustness of the proposed methodologies to changes in the training and validation set size. A final section synthesizes the main results of this paper and proposes some future research lines.

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 [27]. 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 a 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 (Td), atmospheric storm surge (Sg), significant wave height (Hs) and wave peak period (Tp). Each input should be provided to the system in a time series format, so that $\boldsymbol{Td} = (Td_t)_{t=1,\dots,L}$, and similarly for the other three inputs. The code outputs a time series of the same length of the inputs, with the value at time $t \in \{1, \dots, L\}$ indicating the overtopped and overflowed water volume during a period equal to the time span between any pair of consecutive instants. From that series, it is naturally possible to compute the cumulative overtopped and overflowed water volume along this transect until time instant

t. We denote that quantity by CV_t. As explained in the introduction of the article, here 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 overtopped and overflowed water volume during the span of an event. It corresponds to the last value of the CV_t series, CV_L. From here on, we denote this quantity by FCV, which stands for *final cumulative volume*.

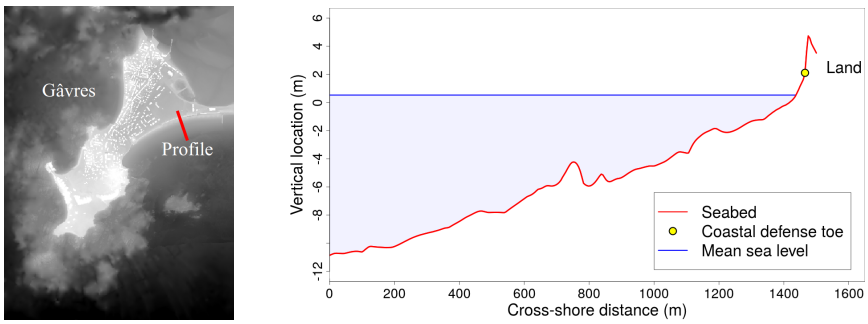


Figure 1: Illustration of the cross-shore transect considered in the RISCOPE application. Vertical location is referenced to the altimetric French system IGN69.

Calculations in the computer code involve the statistical model SWAN [28] and the EurOtop equations [29], both described below.

$$\text{Inputs} \rightarrow \textbf{SWAN} \rightarrow \textbf{EurOtop} \rightarrow \text{Outputs}$$

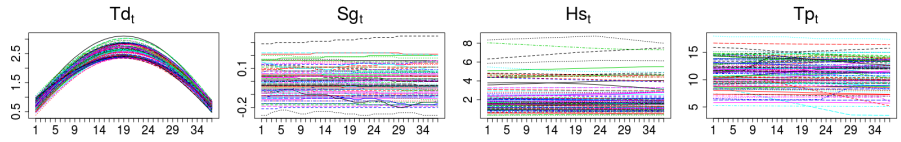
- **SWAN** is a spectral wave model which allows computing the wave conditions at the coastal defence toe, accounting for water level variations induced by tide and surge.
- **EurOtop** refers to the use of the overtopping and overflow discharge formulas provided in the Eurotop (2018) manual ([29], Eq. 5.11 and 5.20). These formulas require as input the wave conditions at the coastal defence toe, 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 discharge, the overtopped and overflowed water volume along the transect 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 intermediate 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 more computationally time consuming models. We remark that in this simplified code, FCV is equal to the sum over time of the overtopped and overflowed water volume; this latter being estimated from scalar quantities. Thus, for this simplified code, a metamodel based on a scalar representation of the inputs may provide relatively good predictions. However, in a future stage

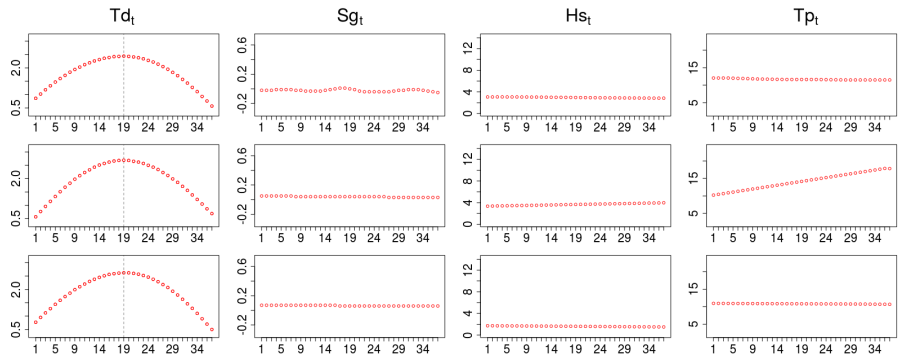
of the RISCOPE project we will address intrinsically functional problems such as the estimation of the water height at different points on land (i.e., in the space between the back of the coastal defense and inland). At that point, we expect the functional metamodels to be able to better reproduce the shape of the output than the scalar ones.

2.2. Dataset

For purposes of training and validation of the metamodel, we rely on a dataset composed of hindcasts of past conditions of *Td*, *Sg*, *Hs* and *Tp*. All registers are located offshore of the study site over the period 1900-2016. The dataset is constituted by the concatenation of hindcasts of different sources (see Appendix A), with bias corrections between the hindcasts through a quantile-quantile correction method (for more details, see [30]). The various hindcasts have different time steps. As the main driver, *Td*, significantly changes in 10 minutes, the other three inputs were also interpolated at a 10 min time step. Then, the long dataset 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 peak 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 *Td* series has a characteristic parabolic shape, which is consistent among events. In fact, its peak (always located at time instant $t = 19$) 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 to represent each input will depend on the sensitivity of the output of the code to changes on it. Even quite small and visually negligible changes on some input might cause important changes in the ouput depending on the interactions that happen within the code.

3. Theoretical background

3.1. Scalar and functional inputs of computer codes

In this paper we study the emulation of an expensive-to-run computer model f_{code} by means of a metamodel. Throughout our discussions, we discriminate between scalar and functional inputs. For the sake of clarity, we stick to the following vocabulary and definitions:

(a) When the code is $\boldsymbol{x} \mapsto f_{\text{code}}(\boldsymbol{x})$, with $\boldsymbol{x} = \left(x^{(1)}, \ldots, x^{(ds)}\right)'$ and $x^{(k)} \in \mathbb{R}$ for $k = 1, \ldots, ds$, we say that the code has $ds \in \mathbb{N}$ scalar inputs, we call $x^{(k)}$ for $k = 1, \ldots, ds$ a scalar input, and we call $\boldsymbol{x}$ a vector of scalar inputs. For simplicity, we may also refer to $\boldsymbol{x}$ as scalar inputs.

(b) When the code is $\boldsymbol{f} \mapsto f_{\text{code}}(\boldsymbol{f})$, with $\boldsymbol{f} = \left(f^{(1)}, \ldots, f^{(df)}\right)'$ and $f^{(k)} : T_k \subset \mathbb{R} \to \mathbb{R}$ for $k = 1, \ldots, df$, we say that the code has $df \in \mathbb{N}$ functional inputs, we call $f^{(k)}$ for $k = 1, \ldots, df$ a functional input, and we call $\boldsymbol{f}$ a vector of functional inputs. For simplicity, we may also refer to $\boldsymbol{f}$ as functional inputs.

(c) When the code is $(\boldsymbol{x}, \boldsymbol{f}) \mapsto f_{\text{code}}(\boldsymbol{x}, \boldsymbol{f})$, we say that the code has ds scalar inputs and df functional inputs, and we use the same vocabulary as before for $\boldsymbol{x}$ and $\boldsymbol{f}$.

3.2. Gaussian process metamodeling of scalar-input codes

Let us first consider the scalar-input setting where f_{code} models the relationship between a vector of scalar inputs $\boldsymbol{x} = \left(x^{(1)}, \ldots, x^{(ds)}\right)' \in \mathbb{R}^{ds}$ and an output variable of interest $y \in \mathbb{R}$, with $y = f_{\text{code}}(\boldsymbol{x})$. As the evaluation of f_{code} is computationally costly, it is proposed to build a light-to-run statistical model to approximate it. To this end, there is available a learning set $D = \{(\boldsymbol{x_1}, y_1), \ldots, (\boldsymbol{x_n}, y_n)\}$. 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. For any pair of input vectors $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathbb{R}^{ds}$, the Gaussian process model can be written as:

$$\xi(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot)), \tag{1}$$

with

$$m(\boldsymbol{x}) = \mathbb{E}[\xi(\boldsymbol{x})] \qquad \text{and} \qquad k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \mathbb{E}[(\xi(\boldsymbol{x}) - m(\boldsymbol{x}))(\xi(\tilde{\boldsymbol{x}}) - m(\tilde{\boldsymbol{x}}))]. \quad (2)$$

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 [25] for more details). Furthermore, the Gaussian process model often has a very high prediction power compared to other approaches [4]. In addition, 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 $n \times ds$ 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 $n_* \times ds$ inputs matrix of prediction points. The Gaussian conditioning theorem (see e.g., [25]) implies that, conditionally to $\boldsymbol{y}$, ξ is a Gaussian process with mean and covariance functions m_n and k_n defined by

$$m_n(\boldsymbol{X_*}) := \mathbb{E}[\xi(\boldsymbol{X_*}) | \boldsymbol{y}] = K(\boldsymbol{X_*}, \boldsymbol{X}) K(\boldsymbol{X}, \boldsymbol{X})^{-1} \boldsymbol{y} \quad (3)$$

and

$$\begin{aligned} k_n(\boldsymbol{X_*}, \boldsymbol{X_*}) &:= \mathrm{Cov}[\xi(\boldsymbol{X_*}), \xi(\boldsymbol{X_*}) | \boldsymbol{y}] \\ &= K(\boldsymbol{X_*}, \boldsymbol{X_*}) - K(\boldsymbol{X_*}, \boldsymbol{X}) K(\boldsymbol{X}, \boldsymbol{X})^{-1} K(\boldsymbol{X}, \boldsymbol{X_*}), \end{aligned} \quad (4)$$

where $K(\boldsymbol{X}, \boldsymbol{X})$ denotes the $n \times n$ matrix of covariances $(k(\boldsymbol{x_i}, \boldsymbol{x_j}))_{1 \leq i,j \leq n}$ among all pairs of training input 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

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

In practice the conditional mean (3) is used as an estimation of the true function f_{code} at the test points $\boldsymbol{X_*}$, while the conditional variance (4) is often interpreted as a measure of the local error of the prediction [8].

Gaussian process models are flexible by incorporating diverse types of covariance functions (a.k.a. kernels), being aware that only functions that yield symmetric positive semidefinite covariance matrices are valid choices [25]. The selection of the covariance function encodes assumptions such as the degree of regularity of the underlying process [31]. A general expression for the covariance between any pair of scalar input points $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathbb{R}^{ds}$ is given by

$$k(\boldsymbol{x} - \tilde{\boldsymbol{x}}; \sigma^2, \boldsymbol{\theta_s}) = \sigma^2 \, R(\boldsymbol{x} - \tilde{\boldsymbol{x}}; \boldsymbol{\theta_s}), \quad (5)$$

where σ^2 is the variance of the stochastic process and R denotes the correlation function which governs the degree of similitude between input points through

the use of the vector of length-scale parameters $\boldsymbol{\theta_s} = \left(\theta_s^{(1)}, \ldots, \theta_s^{(ds)}\right)$. Together, σ^2 and $\boldsymbol{\theta_s}$ are the so-called hyperparameters of the model which have to be estimated.

Examples of standard covariance functions are given for instance in [26] and [32]. Without loss of generality, in this paper we make use of the Matérn 5/2 kernel defined in its anisotropic form for scalar inputs as

$$k(\boldsymbol{\tau}; \sigma^2, \boldsymbol{\theta_s}) = \sigma^2 \left(1 + \sqrt{5}\, \|\boldsymbol{\tau}\|_{L^2,\boldsymbol{\theta_s}} + \frac{5\, \|\boldsymbol{\tau}\|^2_{L^2,\boldsymbol{\theta_s}}}{3}\right) exp\left(-\sqrt{5}\, \|\boldsymbol{\tau}\|_{L^2,\boldsymbol{\theta_s}}\right), \quad (6)$$

where $\boldsymbol{\tau} = \boldsymbol{x} - \tilde{\boldsymbol{x}}$ and $\|\boldsymbol{\tau}\|_{L^2,\boldsymbol{\theta_s}}$ denotes the anisotropic L^2 norm of $\boldsymbol{x} - \tilde{\boldsymbol{x}}$ which can be written as

$$\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_{L^2,\boldsymbol{\theta_s}} = \sqrt{\sum_{k=1}^{ds} \frac{\left\|x^{(k)} - \tilde{x}^{(k)}\right\|^2}{\left(\theta_s^{(k)}\right)^2}}. \quad (7)$$

In the equation above, $\|\cdot\|$ is the Euclidean norm in $\mathbb{R}$, which by definition is just the absolute value of the quantity. Intuitively, if $\boldsymbol{x} = \tilde{\boldsymbol{x}}$, then the correlation is 1, whereas if the distance between both vectors tends to infinity, then the correlation tends to 0.

3.3. Gaussian process metamodeling of functional-input codes

Let us now consider the functional-input setting where f_{code} models the relationship between a vector of functional inputs $\boldsymbol{f} = \left(f^{(1)}, \ldots, f^{(df)}\right)'$, with $f^{(k)} : T_k \subset \mathbb{R} \to \mathbb{R}$ for $k = 1, \ldots, df$, and an output variable of interest $y \in \mathbb{R}$, so that $y = f_{\text{code}}(\boldsymbol{f})$. Similarly to the scalar-input case, we assume that there is available a learning set $D = \{(\boldsymbol{f_1}, y_1), \ldots, (\boldsymbol{f_n}, y_n)\}$. The extension of Gaussian processes to functional inputs reduces to the selection of a suitable distance for functions to be used within the correlation function. That is the topic of this section.

3.3.1. Three distances for functional inputs

Let us consider two functional data points $\boldsymbol{f} = \left(f^{(1)}, \ldots, f^{(df)}\right)'$ and $\tilde{\boldsymbol{f}} = \left(\tilde{f}^{(1)}, \ldots, \tilde{f}^{(df)}\right)'$. The anisotropic L^2 norm of $\boldsymbol{f} - \tilde{\boldsymbol{f}}$ can be written

$$\|\boldsymbol{f} - \tilde{\boldsymbol{f}}\|_{L^2,\boldsymbol{\theta_f}} = \sqrt{\sum_{k=1}^{df} \frac{\left\|f^{(k)} - \tilde{f}^{(k)}\right\|^2}{\left(\theta_f^{(k)}\right)^2}}, \quad (8)$$

where $\boldsymbol{\theta_f} = \left(\theta_f^{(1)}, \ldots, \theta_f^{(df)}\right)$ is the vector of length-scale parameters for the df functional input variables and $\|\cdot\|$ is any norm for functions.

Note that (8) is just the straightforward extension of (7) for functional inputs. However, the norm in each term is no longer as trivial as in the scalar case and different paths can be followed from here. Most times in the literature, the norm is computed using one of the three distances that we discuss now.

The first approach is to use the L^2 norm for functions, under the mild assumption that $f^{(\ell)}$ and $\tilde{f}^{(\ell)}$ for $\ell = 1, \ldots, df$ have finite L^2 norm. In that case, (8) becomes

$$\|\boldsymbol{f} - \tilde{\boldsymbol{f}}\|_{F,\boldsymbol{\theta_f}} := \sqrt{\sum_{k=1}^{df} \frac{\int_{T_k} \left(f^{(k)}(t) - \tilde{f}^{(k)}(t)\right)^2 dt}{\left(\theta_f^{(k)}\right)^2}}, \tag{9}$$

where $T_k \subset \mathbb{R}$ is the domain of the functions $f^{(k)}$ and $\tilde{f}^{(k)}$.

The second approach is to make a projection of $f^{(k)}$ and $\tilde{f}^{(k)}$ for $k = 1, \ldots, df$ onto a subspace of finite, small or moderate dimension, and then use the L^2 norm of the projections in (8) instead of the L^2 norm of the original functions. For illustration, let $\Pi\big(f^{(k)}\big)$ and $\Pi\big(\tilde{f}^{(k)}\big)$ denote the projections of $f^{(k)}$ and $\tilde{f}^{(k)}$ onto the space generated by a basis $\boldsymbol{B^{(k)}} = \left\{B_1^{(k)}, \ldots, B_{p_k}^{(k)}\right\}$. For $k = 1, \ldots, df$, the expression to obtain $\Pi\big(f^{(k)}\big)$ and $\Pi\big(\tilde{f}^{(k)}\big)$ can then be written as

$$\Pi\big(f^{(k)}\big)(t) = \sum_{r=1}^{p_k} \alpha_r^{(k)} B_r^{(k)}(t) \qquad \text{and} \qquad \Pi\big(\tilde{f}^{(k)}\big)(t) = \sum_{r=1}^{p_k} \tilde{\alpha}_r^{(k)} B_r^{(k)}(t), \tag{10}$$

respectively. The projection dimension p_k has to be chosen strategically so that the functions are represented well enough and computations for the meta-model remain tractable. The projection coefficients $\boldsymbol{\alpha^{(k)}} = \left(\alpha_1^{(k)}, \ldots, \alpha_{p_k}^{(k)}\right)$ and $\tilde{\boldsymbol{\alpha}}^{\boldsymbol{(k)}} = \left(\tilde{\alpha}_1^{(k)}, \ldots, \tilde{\alpha}_{p_k}^{(k)}\right)$ are typically set up to minimize the error of the projection with respect to the original input function. Diverse methods such as B-splines, Fourier, PCA, kPCA or PLS can be used to generate the basis functions for the projection of each input variable. The only requirement is that the projection has the structure diplayed in (10).

Once the projection of each curve is made, the norm $\|f^{(k)} - \tilde{f}^{(k)}\|_{L^2}$ can be replaced by its projection based approximation $\left\|\Pi\big(f^{(k)}\big) - \Pi\big(\tilde{f}^{(k)}\big)\right\|_{L^2}$ in (9) to obtain

$$\left\|\Pi(\boldsymbol{f}) - \Pi(\tilde{\boldsymbol{f}})\right\|_{D,\boldsymbol{\theta_f}} := \sqrt{\sum_{k=1}^{df} \frac{\int_{T_k} \left(\sum_{r=1}^{p_k} \left(\alpha_r^{(k)} - \tilde{\alpha}_r^{(k)}\right) B_r^{(k)}(t)\right)^2 dt}{\left(\theta_f^{(k)}\right)^2}}. \tag{11}$$

As noted by [4], an efficient computation of $\left\|\Pi\big(f^{(k)}\big) - \Pi\big(\tilde{f}^{(k)}\big)\right\|_{L^2}$ is possible, by reduction to a norm in $\mathbb{R}^{p_k}$:

$$\left\|\Pi\left(f^{(k)}\right)-\Pi\left(\tilde{f}^{(k)}\right)\right\|_{L^2}^2=\int_{T_k}\left(\sum_{r=1}^{p_k}\left(\alpha_r^{(k)}-\tilde{\alpha}_r^{(k)}\right)B_r^{(k)}(t)\right)^2dt$$
$$:=\int_{T_k}\left(\sum_{r=1}^{p_k}\delta_r^{(k)}B_r^{(k)}(t)\right)^2dt$$
$$=\left(\boldsymbol{\delta^{(k)}}\right)'\boldsymbol{J^{(k)}}\left(\boldsymbol{\delta^{(k)}}\right)$$
$$=\left\|\boldsymbol{\delta^{(k)}}\right\|_{\boldsymbol{J^{(k)}}},\tag{12}$$

where $\boldsymbol{J^{(k)}}$ is the $p_k\times p_k$ Gram matrix $\left(\int_{T_k}B_i^{(k)}(t)B_j^{(k)}(t)dt\right)_{1\le i,j\le p_k}$. The interesting fact about (12) is that $\boldsymbol{J^{(k)}}$ does not depend on the coefficients of the decomposition, but only on the set of basis functions. Thus, it can be stored and reused, saving processing time. Moreover, when the projection basis is an orthonormal family of vectors (e.g., PCA basis), $\boldsymbol{J^{(k)}}$ is simply the identity matrix of dimension $p_k\times p_k$.

The third approach is a variation of the second one, where the distance only considers the coefficients of the decomposition and disregards the information in the basis functions. In addition, this approach works with p_k length-scale parameters for the k-th model input instead of only one as in (9) and (11):

$$\left\|\Pi(\boldsymbol{f})-\Pi(\tilde{\boldsymbol{f}})\right\|_{S,\dot{\boldsymbol{\theta}}_{\boldsymbol{f}}}=\sqrt{\sum_{k=1}^{df}\sum_{r=1}^{p_k}\frac{\left(\alpha_r^{(k)}-\tilde{\alpha}_r^{(k)}\right)^2}{\left(\dot{\theta}_{f,r}^{(k)}\right)^2}}.\tag{13}$$

Note that here we denote the vector of length-scale coefficients by $\dot{\boldsymbol{\theta}}_{\boldsymbol{f}}$, with elements $(\dot{\theta}_{f,r}^{(k)})_{1\le r\le p_k,1\le k\le df}$, to differentiate with the shorter vector $\boldsymbol{\theta_f}$, with elements $(\theta_f^{(k)})_{1\le k\le df}$ used in (9) and (11). Also note that (13) can be interpreted as if each projection coefficient $\alpha_r^{(k)}$ was taken as an individual scalar input of the model, since (13) matches the structure of the anisotropic L^2 norm for scalars shown in (7).

For applications of the three approaches, the reader is referred to [33], [4] and [5], in the corresponding order. For the sake of theory, we expressed (9), (11) and (12) in terms of infinite-dimensional inputs $f^{(k)}:T_k\subset\mathbb{R}\to\mathbb{R}$. However, in practice one typically does not have access to the infinite-dimensional function, but to a vectorial representation of it $\left(f^{(k)}\left(t_1^{(k)}\right)\ldots,f^{(k)}\left(t_{L_k}^{(k)}\right)\right)'$, with $\left\{t_1^{(k)},\ldots,t_{L_k}^{(k)}\right\}\subset T_k$, as is the case in our costal flooding application. In (9), if a vectorial representation of the input is provided, the integral could be computed by numerical approximation or substituted by the Euclidean norm of a vector. A numerical approximation of the integral can be used in (11) and (12) as well.

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 terms of metamodel predictability. However, the two distances based on the projection of the inputs are motivated by potential gains in speed and tractability. The dimension of

our inputs in the coastal flooding application case is large enough (time series of length 37) to take this advantage into consideration. Therefore, in this paper we focus on the two approaches based on the functional decomposition of the inputs, i.e., distances (11) and (13).

3.4. Gaussian process metamodeling with scalar and functional inputs

Our coastal flooding application matches the functional-input setting $\boldsymbol{f} \mapsto f_{\text{code}}(\boldsymbol{f})$, with $\boldsymbol{f} = (Td, Sg, Hs, Tp)'$. However, we want to provide the metamodel with some flexibility so that if we find convenient to remove a functional input from the explanatory variables, we can still keep active a scalar representation of it (e.g., its temporal mean). To do so, we consider the hybrid-input setting where f_{code} models the relationship between a vector of scalar inputs $\boldsymbol{x} = \left(x^{(1)}, \ldots, x^{(ds)}\right)' \in \mathbb{R}^{ds}$, a vector of functional inputs $\boldsymbol{f} = \left(f^{(1)}, \ldots, f^{(df)}\right)'$, with $f^{(k)} : T_k \subset \mathbb{R} \to \mathbb{R}$ for $k = 1, \ldots, df$, and an output variable of interest $y \in \mathbb{R}$, with $y = f_{\text{code}}(\boldsymbol{x}, \boldsymbol{f})$. We model the correlation between scalar points and the correlation between functional points as described in Sections 3.2 and 3.3, respectively. To integrate both types of inputs in the model, we follow the approach in [4] and adopt an anisotropic, tensor-product kernel of the form

$$\text{Cov}(\xi(\boldsymbol{x}, \boldsymbol{f}), \xi(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{f}})) = \sigma^2 \; R(\boldsymbol{x} - \tilde{\boldsymbol{x}}; \boldsymbol{\theta_s}) \; R\Big(\boldsymbol{f} - \tilde{\boldsymbol{f}}; \boldsymbol{\theta_z}\Big), \tag{14}$$

with $\boldsymbol{\theta_z}$ denoting either the vector $\boldsymbol{\theta_f}$ or the vector $\dot{\boldsymbol{\theta}}_{\boldsymbol{f}}$, depending on whether the distance $\|\cdot\|_{D,\boldsymbol{\theta_f}}$ or $\|\cdot\|_{S,\dot{\boldsymbol{\theta}}_{\boldsymbol{f}}}$ is used. To illustrate, if we take our tensor-product kernel from the Matérn 5/2 family (6) and we use the distance $\|\cdot\|_{D,\boldsymbol{\theta_f}}$ for the functional inputs, we obtain:

$$\begin{aligned}
\text{Cov}(\xi(\boldsymbol{x}, \boldsymbol{f}), \xi(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{f}})) = \sigma^2 \; & \left(1 + \sqrt{5}\, \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_{L^2,\boldsymbol{\theta_s}} + \frac{5\, \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2_{L^2,\boldsymbol{\theta_s}}}{3}\right) \\
& exp\left(-\sqrt{5}\, \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_{L^2,\boldsymbol{\theta_s}}\right) \\
& \left(1 + \sqrt{5}\, \|\boldsymbol{f} - \tilde{\boldsymbol{f}}\|_{D,\boldsymbol{\theta_f}} + \frac{5\, \|\boldsymbol{f} - \tilde{\boldsymbol{f}}\|^2_{D,\boldsymbol{\theta_f}}}{3}\right) \\
& exp\left(-\sqrt{5}\, \|\boldsymbol{f} - \tilde{\boldsymbol{f}}\|_{D,\boldsymbol{\theta_f}}\right).
\end{aligned} \tag{15}$$

4. Exploration strategy

The construction of a surrogate model requires making a series of decisions that may have significant impact on its performance. The projection method and projection dimension, the distance function to measure similarity between functional input points, as well as the set of functional predictors to keep active make all part of those decisions. The ideal combination of those parameters varies from one application to the other. In this section, we present an exploration methodology designed to select a suitable combination of these or some other structural parameters. A scheme of the proposed methodology is presented in Figure 3 and its main steps are briefly described below.

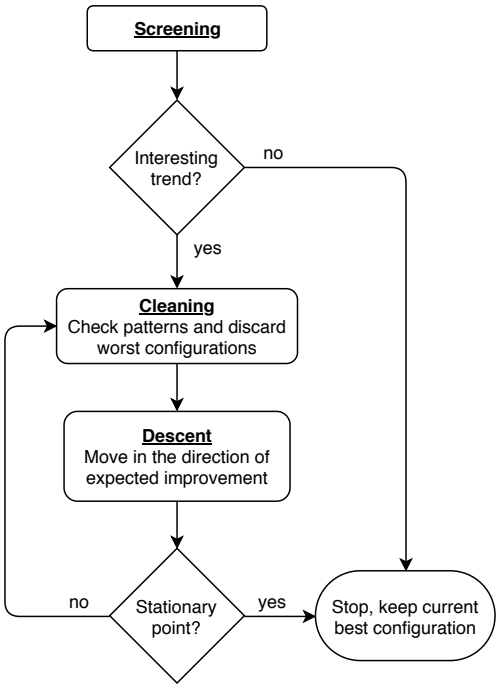


Figure 3: Exploration strategy flowchart.

1. *Screening.* This step is intended to provide an overview of the effect that each parameter has on the performance of the metamodel. The main objectives here are to:

   - Identify patterns, as could be the dominance of certain levels of a given parameter over its other levels. For intance, determine if some projection method clearly outperforms the others.
   - Detect trend in numerical parameters. For example, determine if the performance of the metamodel improves by increasing or decreasing the projection dimension.
   - Determine if the functional representation of each functional input variable adds information to the metamodel or if a scalar representation of it is enough. To do so, a metamodel using only a scalar representation of each functional input is used as a benchmark.

   As one of the main purposes of the exploration methodology is to reduce the dimension of the inputs considerably, we start by exploring configurations with the lowest possible dimension. For instance, configurations using projections of dimension 1, 2 and 3.
2. *Cleaning.* If the screening stage allows to detect a trend to better performance with larger projection dimension, the exploration is extended in that 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 dominated levels of the parameters, identified in screening stage.

3. *Descent.* Once the dominated levels of the parameters have been discarded, greater values of projection dimension are evaluated. To do so, a new factorial design of experiments is built, considering only the non-dominated levels of the other parameters. We call this stage *descent* as its purpose is to explore a new region in the domain of the structural parameters, where the projection error will likely be reduced. Similarly to the response surface methodology [34, 35], this stage is repeated until a stationary point is identified.

4.1. Scope of the methodology

This section briefly discusses some of the concerns that users may have on the scope and adaptability of the proposed methodology. This discussion seeks to provide an insight on the possible uses of the methodology for a variety of modeling scenarios.

- *Learning and validation sample size:* generally speaking, the quality of a regression model has direct correlation with the number of training points. On the other hand, the robustness of the performance statistic used to assess its quality correlates with the number of validation points. Hence, in the frame of this, or any other exploration methodology, fewer training points will reduce the quality of every configuration and fewer validation points will reduce the robustness of the measure used for comparison.

  This cleared up, we could say that the proposed methodology is suitable for both, scenarios of relatively large or considerably short data availability (samples of the code). This aspect is thoroughly discussed in Section 6. For cases of very limited data the performance of each configuration could be assessed by means of cross validation / bootstrap methods [36]. These adopt resampling techniques to estimate the performance of a regression model using training and validation sets of modest size. Efficient formulas exist, e.g., for Gaussian processes [37, 38] and polynomial chaos expansions [11].

- *Large number of features or levels:* in the proposed method, the number of experimental conditions to test grows exponentially with the number of structural parameters. The growth rate, in turn, increases with the number of levels of each parameter. A convenient fact is that all configurations can be trained and validated using the same set of samples of the expensive code. Nonetheless, we have to acknowledge that the processing time to build all metamodel configurations may turn prohibitive for some applications with several inputs and/or levels. A possible way to circumvent this inconvenience, and a potential topic of future research, is to extend the methodology towards a metaheuristic-based algorithm able to deal with wider solution spaces. In this regard, Ant colony programming [39], Artificial bee colony programming [40] and Genetic programming [41] could be suitable choices based on their recurrent usage to solve symbolic regression and automatic programming problems whose nature is quite close to that of the problem discussed here.

- *Functional inputs in larger dimensions:* in both case studies revised here, the functional inputs are time series (functions in dimension one). However, the exploration strategy is generic enough to account for inputs in

larger dimensions such as fields or images (functions in dimension two). To do so, tensorized finite dimensional projection spaces could be considered (see e.g., [42] or [43]). If the functional inputs are functions from $T \subset \mathbb{R}^d \to \mathbb{R}$, then for tensorized projection spaces, the projection dimension is of the form $p^{(1)} \times ... \times p^{(1)}$, where $p^{(1)}, \ldots, p^{(d)}$ can be defined as structural parameters and the already defined steps and rules will hold.

- *Functional output:* the exploration methodology can be used in case of both, scalar and functional outputs. The latter can be handled in at least two ways which are described in the following paragraphs.

  The first way is to transform the problem to a scalar-output version of it. For instance, if the original output is a time series, an individual metamodel can be used to predict the output at each time instant or the time index can be taken as an input of the metamodel, making the output scalar (see e.g., [44]). In both cases, our methodology will proceed as illustrated in the case studies.

  The second way is to project the output onto a space of lower dimension and then fit an individual metamodel to predict each component of the projection (see e.g., [10] or [44]). For each individual metamodel, our methodology will proceed as in the case studies. It is worth mentioning that for this approach, the optimum projection dimension for the output, in terms of projection error, will be the largest possible one. This value, however, will not necessarily be the optimum in terms of metamodel predictability and will likely be a highly expensive choice in terms of computational time. Thereby, a sound approach would be to optimize the output projection dimension w.r.t the prediction error of the metamodel and consider the processing time of the metamodel as a second objetive function or as a constraint (i.e., discard any configuration whose processing time exceeds certain limit).

- *Stochastic code:* metamodeling with stochastic codes often reduces to multiple subproblems consisting on estimating moments or quantiles of the output distribution given an input value (see e.g., [45] and [46], respectively). All these are scalar-output problems that can be addressed similarly to our case studies.

  More advanced techniques predict the probability density function (pdf) [47] or the quantile function [48] of the output given an input value. If the output is scalar, then this case can be perceived and approached as the functional output problems described in the previous item. If the output is functional, for instance a time series, a pdf could be built for each observation at each time step. Then, one could put the time index as an input and the problem will likewise reduce to the case of functional output already discussed.

4.2. Analytic case

In this section we illustrate our exploration methodology by means of a toy case. It corresponds to the second analytic case presented in [4], with a slight different domain for the functional inputs. In [4], a functional-input Gaussian process metamodel is built using B-spline projections of dimension 5 and order

4. Here, we use the exploration strategy presented in previous section to find an attractive metamodel configuration.

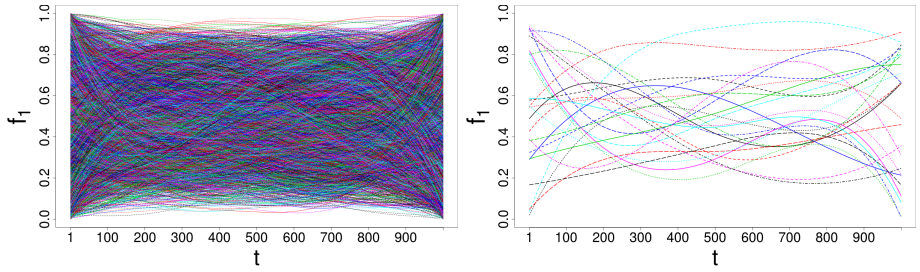
Let $\mathcal{F}$ be the set of continuous functions from $[0, 1000]$ to $\mathbb{R}$. Consider a black box computer code receiving the scalar inputs $\boldsymbol{x} = \left(x^{(1)}, x^{(2)}\right) \in [0,1]^2$ and the continous functional inputs $\boldsymbol{f} = (f^{(1)}, f^{(2)}) \in \mathcal{F}^2$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

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

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 unfeasible to recover the output by means of independent computations on scalar representations of the input over its domain. This 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.

4.2.1. Dataset

We started by creating a dataset with 5000 runs of the code that could be used later to generate multiple independent training and validation sets. The coordinates of the 5000 scalar input points where uniformily sampled over their domain. For the functional part, we followed the approach proposed in [4] by making the design over the coefficients of a functional decomposition. To this end, we modeled each functional input as a B-spline of dimension 5 and order 4. Then, we built a Latin Hypercube design [49] with 5000 points taking the decomposition coefficients as coordinates. We remark that the order and dimension used for the constitution of the dataset 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 on the optimal design of experiments, we do not develop further this aspect and match the 5000 scalar coordinates to the 5000 functional coordinates using a random permutation. For a more elaborated approach to perform this pairing, the reader is referred to [4]. The full dataset and a set of 25 trajectories of the function $f^{(1)}$ are shown in Figures 4a and 4b, respectively.



(a) Full dataset. (b) 25 sample functions.

Figure 4: Illustration of the functional input f_1.

4.2.2. Screening

Once the dataset 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 average over $[0, 1000]$ of each function as its scalar representation. Then, we define the scalar metamodel as:

$$\begin{aligned} \mathcal{M}_{00}: \ [0,1]^2 \times [0,1]^2 \to & \mathbb{R}, \\ (\boldsymbol{x}, \ddot{\boldsymbol{f}}) \mapsto & \mathcal{M}_{00}(\boldsymbol{x}, \ddot{\boldsymbol{f}}). \end{aligned} \tag{16}$$

For the funtional metamodels, let us consider a shifted version of $\boldsymbol{f}$, computed as $\tilde{\boldsymbol{f}} = \boldsymbol{f} - \ddot{\boldsymbol{f}}$. Then, let $\boldsymbol{\Pi} = (\Pi_1, \Pi_2)$ denote the vector of projections of the elements in $\tilde{\boldsymbol{f}}$ onto a space 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 between the scalar metamodel and any functional metamodel will be attributed to the addition of the corresponding projections. As an example, metamodels with (a) only Π_1 active, (b) only Π_2 active, and (c) both, Π_1 and Π_2 active, are defined in (17), (18) and (19), respectively.

$$\begin{aligned} \mathcal{M}_{f0}: \ [0,1]^2 \times [0,1]^2 \times \mathbb{R}^p \to & \mathbb{R}, \\ (\boldsymbol{x}, \ddot{\boldsymbol{f}}, \Pi_1) \mapsto & \mathcal{M}_{f0}(\boldsymbol{x}, \ddot{\boldsymbol{f}}, \Pi_1). \end{aligned} \tag{17}$$

$$\begin{aligned} \mathcal{M}_{0f}: \ [0,1]^2 \times [0,1]^2 \times \mathbb{R}^p \to & \mathbb{R}, \\ (\boldsymbol{x}, \ddot{\boldsymbol{f}}, \Pi_2) \mapsto & \mathcal{M}_{0f}(\boldsymbol{x}, \ddot{\boldsymbol{f}}, \Pi_2). \end{aligned} \tag{18}$$

$$\begin{aligned} \mathcal{M}_{ff}: \ [0,1]^2 \times [0,1]^2 \times (\mathbb{R}^p)^2 \to & \mathbb{R}, \\ (\boldsymbol{x}, \ddot{\boldsymbol{f}}, \boldsymbol{\Pi}) \mapsto & \mathcal{M}_{ff}(\boldsymbol{x}, \ddot{\boldsymbol{f}}, \boldsymbol{\Pi}). \end{aligned} \tag{19}$$

In this notation, 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 Π_2 is active. However, the notation is generic in the sense that each of the metamodels, (17), (18) and (19), might represent configurations involving diverse combinations of projection method, projection dimension and distance measure.

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 structural parameters (see Table 1), except for those cases where both Π_1 and Π_2 are inactive. Those correspond to redundant counts of the scalar metamodel. In the rest of the paper, for concision, we write $\|\cdot\|_{D,\boldsymbol{\theta_f}}$ as $\|\cdot\|_{D,\boldsymbol{\theta}}$ and $\|\cdot\|_{S,\dot{\boldsymbol{\theta}}_{\boldsymbol{f}}}$ as $\|\cdot\|_{S,\boldsymbol{\theta}}$. For the numerical experiments, we concentrate on the B-splines and PCA projection methods, which have consistently appeared as effective ways to model functional data (see e.g., [4, 50, 9] for applications of B-splines and [5, 51] for applications of PCA). Both methods work with a projection of the form (10) and thus, they are suitable choices in our framework. For a full derivation of B-splines and PCA equations, the interested reader may refer to [50] and [52], respectively. Other projection methods such as PLS [53] or kPCA [22] are valid choices as well, and we encourage the inclusion of multiple projection methods in the analysis for comparison. Furthermore, we impose the projection dimension of every functional input to be the same, for simplicity of exposition. That is, we let $p_1 = ... = p_{df} = p$, with the notation of Section 3.

Parameter	**Levels**
State of Π_1	inactive, active
State of Π_2	inactive, active
Projection method	B-splines, PCA
Projection dimension	1, 2, 3
Distance	$\|\cdot\|_{D,\boldsymbol{\theta}}$, $\|\cdot\|_{S,\boldsymbol{\theta}}$

Table 1: Analytic case: parameters and levels for the screening stage.

In all cases, we set the projection coefficients $\alpha_1^{(k)}, \ldots, \alpha_{p_k}^{(k)}$ using an ordinary least squares formulation (see Appendix B). We used the Matérn 5/2 kernel (6) and estimated the hyperparameters of each metamodel by maximizing the joint likelihood of the data [54, 26] in a similar way to the R package DiceKriging [55]. The optimization was done by the R-function *optim*.

We assessed the quality of each configuration by means of the predictive squared correlation coefficient Q^2, which corresponds to the classical coefficient of determinarion R^2 for a test sample, i.e., for prediction residuals [56]. For a test set of n_* output values $y_{*,1}, \ldots, y_{*,n_*}$, with average denoted by $\bar{y}_*$, and corresponding predictions $\hat{y}_{*,1}, \ldots, \hat{y}_{*,n_*}$, the Q^2 is defined as

$$Q^2 = 1 - \frac{\sigma_E^2}{\sigma_T^2}, \tag{20}$$

with

$$\sigma_E^2 = \frac{\sum_{i=1}^{n_*}(y_{*,i} - \hat{y}_{*,i})^2}{n_*} \qquad \text{and} \qquad \sigma_T^2 = \frac{\sum_{i=1}^{n_*}(y_{*,i} - \bar{y}_*)^2}{n_*}.$$

The Q^2 takes values in $[-\infty, 1]$ where 1 indicates perfect fitting to the test data. Thus, it not only allows to make comparisons between configurations, but it also provides information on the absolute quality of each configuration.

To account for the sampling noise, we used a total of 30 independent pairs of training and validation sets for each of the 37 metamodel configurations. Thus, the statistic for comparison between configurations, denoted by $\tilde{Q}^2$, is obtained by computing (20) for each of the 30 samples and then taking the average of the results:

$$\tilde{Q}^2 := \frac{1}{30} \sum_{s=1}^{30} Q_s^2. \tag{21}$$

The 30 pairs were kept fixed for all configurations in order to make the comparison fair. We remark that the correlation between input and output is implicitly taken into account when optimizing a predictability indicator such as the $\tilde{Q}^2$, since the methodology will select a projection dimension sufficiently large to retain the amount of temporal/spatial information of the inputs necessary to accurately reproduce the output.

We let the exploration run until fulfilling one of the following convergence-oriented stopping conditions:

(i) Stop if the slope of a linear least squares fitting to the best $\tilde{Q}^2$ values of the screening is lower than a reference value m^*;

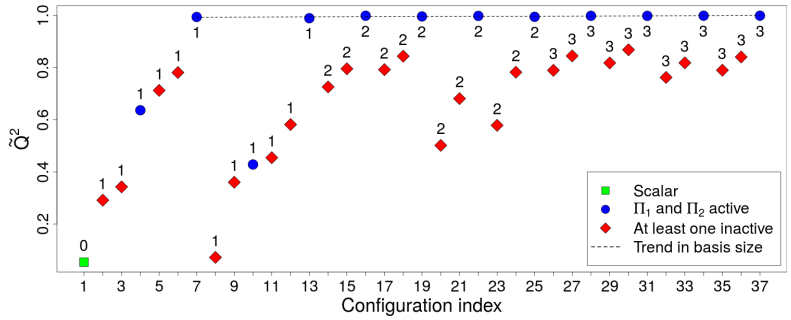
(ii) Stop if a stationary point or plateau is reached. Each time the experiment is expanded to a greater level $\dot{p}$ of projection dimension, compute a new linear least squares fitting to the best $\tilde{Q}^2$ values among those corresponding to configurations based on projections of dimension $\dot{p} - 1$ and $\dot{p}$. If the slope of such a fitting is lower than a reference value m^*, count a flat step. Stop if z consecutive flat steps are registered.

The first rule seeks to prevent the extension of the experiment unless evidence of potential improvement of the $\tilde{Q}^2$ was found during screening. On the other hand, the second rule is oriented to stop if a prospect local optimum is detected or the strategy reached certain degree of convergence. Note that the slope of the linear least squares fitting is updated each time the projection dimension is increased, and the fitting only takes into account the last two projection dimensions. This seeks to obtain clear information on how the $\tilde{Q}^2$ is behaving locally. If other projection dimensions were considered in the fitting, one might end up mistakenly thinking that the $\tilde{Q}^2$ is still improving, when it is not. Also note that we do not stop the search the first time we notice a flat step, but after z consecutive counts. This is to prevent premature stops due to saddle points. Similar stopping rules are often used in general for optimization, e.g., for gradient based methods [57] and heuristics [58].

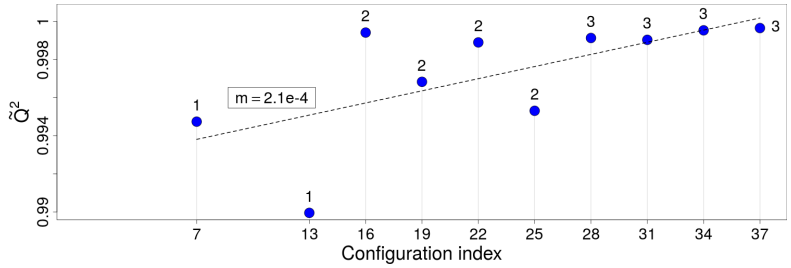
Figure 5 illustrates the performance in terms of $\tilde{Q}^2$ of the 37 metamodel configurations, using 800 training points and 1500 validation points. Those results were obtained using $m^* = 10^{-4}$ and $z = 3$ for the stopping conditions, which based on a set of preliminary tests seem to provide a good balance between degree of exploration and convergence rate. 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.7.



(a) Analytic case: all configurations of screening.



(b) Analytic case: zoom to best configurations of screening. The value of m in the box indicates the slope of the linear least squares fitting of the points.

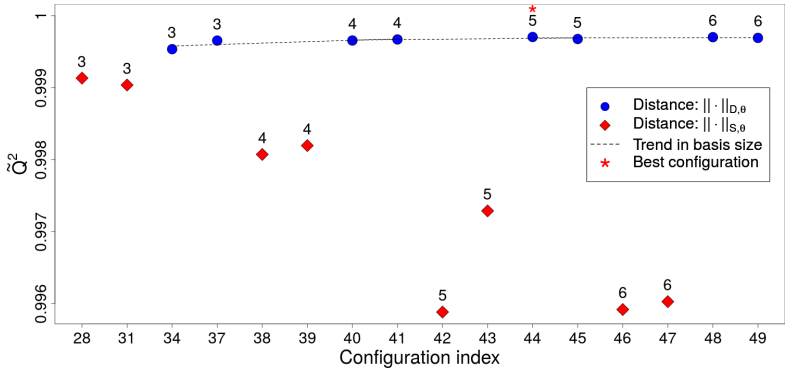
Figure 5: Analytic case: results of the screening experiment. Points are labeled by projection dimension p; the label 0 corresponds to the scalar metamodel.

As a first noticeable result, the scalar metamodel was the worst performing configuration, closely followed by a metamodel with $p = 1$ (configuration 8). 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, it is visible that the $\tilde{Q}^2$ tends to grow as the number of basis functions increases. In fact, the best performing metamodel of the screening stage (configuration 37) was found for $p = 3$, the largest value tested so far. Since the slope of the linear trend was larger than the critical value $m = 10^{-4}$, we proceed to cleaning and descent.

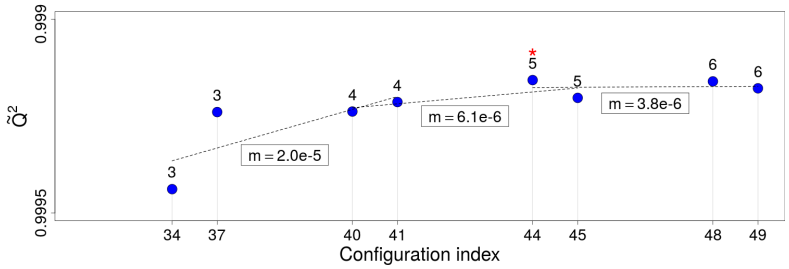
4.2.3. Cleaning and descent

Appart from a positive trend of the $\tilde{Q}^2$ for increments in p, the screening stage revealed that configurations with the functional representation of both

inputs being active dominate any other configuration. Therefore, only this type of metamodel is retained and we start expanding the experiment by increasing p by steps of a unity. At each step, we inspect again for patterns or changes in trend, and cleaning is performed if possible. The $\tilde{Q}^2$ of the new configurations is plotted in Figure 6. A detailed list of the experimental conditions and corresponding results of this stage is provided in Appendix C, Table C.8. From $p = 3$ to $p = 4$, and also from $p = 4$ to $p = 5$, we registered a flat step. At $p = 5$, we removed the configurations using the scalar distance $\|\cdot\|_{S,\boldsymbol{\theta}}$, as those were clearly dominated by configurations using the decomposition-based distance $\|\cdot\|_{D,\boldsymbol{\theta}}$. Then, $p = 5$ to $p = 6$ we registered a third flat step, which fulfilled our secont stopping condition. Thus, at $p = 6$ we stopped.



(a) Analytic case: all configurations of cleaning and descent.



(b) Analytic case: zoom to best configurations of cleaning and descent. The value of m in the box indicates the slope of the linear least squares fitting of the points.

Figure 6: Analytic case: results of cleaning and descent. Points are labeled by projection dimension p. Configurations from the screening stage with $p = 3$ are also plotted here for comparison against configurations with larger p.

The metamodel with an active B-splines representation of size $p = 5$ for both functional inputs, using the decomposition-based distance $\|\cdot\|_{D,\boldsymbol{\theta}}$ (condition 44) is the most attractive configuration found. As we do not know the shape of the $\tilde{Q}^2$ surface, we cannot guarantee that such a configuration provides the global optimum. However, we know that its $\tilde{Q}^2$ is 18.8 times as large as the $\tilde{Q}^2$ of the worst configuration assessed (conditon 1). Considering that, and based on the patterns found during the exploration, condition 44 is likely one of the best metamodel configurations in terms of $\tilde{Q}^2$ for this case study.

The fitting of the ordered true output for the best performing sample of the

best configuration is illustrated in Figure 7. For this sample, the proportion of output values lying within the confidence intervals at 99%, 95% and 90% was 89%, 77% and 68%, respectively.

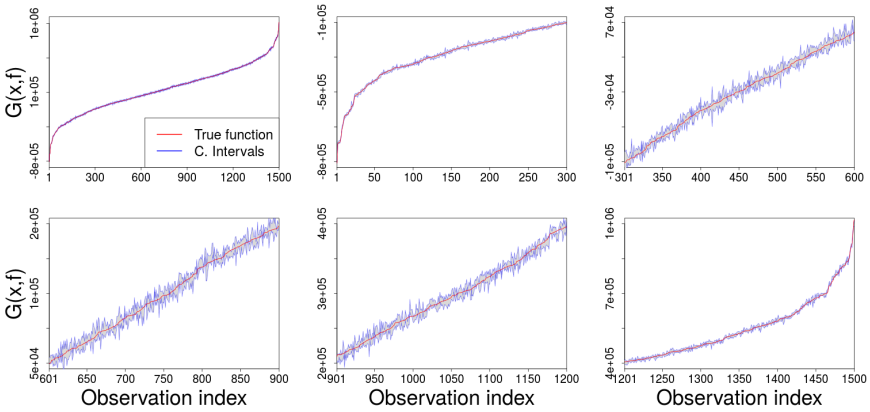
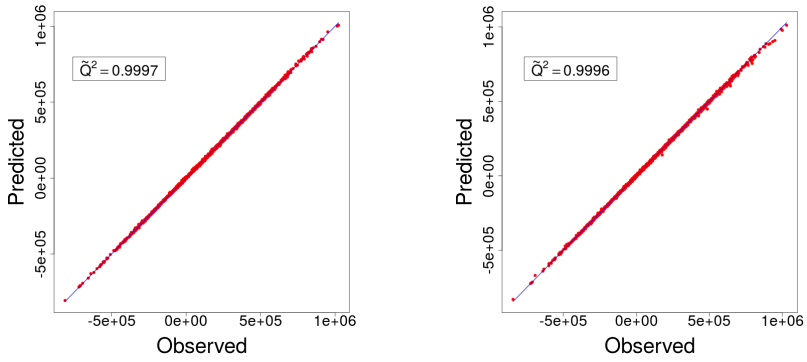


Figure 7: Analytic case: fitting of the best performing sample of the best configuration. Left-top subplot illustrates the whole set of 1500 output values in increasing order; this subplot is then subdivided to generate the remaining 5 subplots by splitting the abscissa into 5 sections and zooming the resulting figures.

On the other hand, the calibration plot for the best and worst samples of the best configuration are presented in Figures 8a and 8b, respectively. Based on these results, we may conclude that this metamodel provides good predictions with no evident fitting problems (e.g., skewness, heavy tails).



(a) Analytic case: best sample.

(b) Analytic case: worst sample.

Figure 8: Analytic case: calibration plot with 1500 data points for the best and worst samples of the best performing metamodel (configuration 44).

5. Coastal flooding case study

In this section, we address the application case introduced in Section 2. Similarly to the analytic case, we implement our exploration methodology to tune the structural parameters of the metamodel. Note that the computer code under study is actually a concatenation of two blocks: (i) the spectral wave

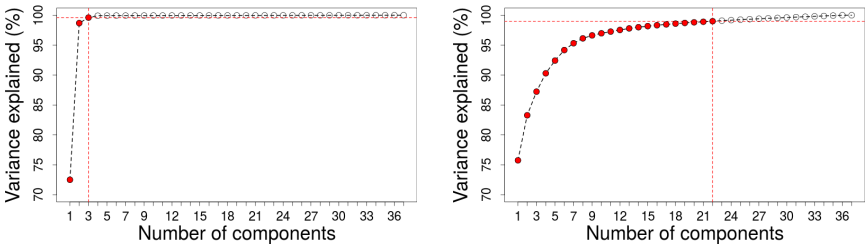
model SWAN; and (ii) the EurOtop overtopping and overflow discharge formulas (see the system description in Section 2.1). A possible way to handle problems involving multiple nested blocks is to use multiple metamodels, one for each block of the system. In a recent work this method was compared to the classical approach based on a single metamodel, using Gaussian processes in both cases [59, 60]. A set of numerical experiments showed a superior prediction capacity when using multiple nested metamodels instead of a single one, however no theoretical guarantees were provided. In the present paper we preferred to keep the simpler yet powerful single metamodel approach and focus on the modeling of functional inputs and optimization of structural parameters. We outlook the comparisson of the two approaches for our application case as an interesting topic of future research.

5.1. Screening

Following the exploration methodology described in Section 4, we start by the screening stage oriented to identify patterns, detect trend and determine if the functional representation of the inputs adds value to the metamodel. We denote by $\mathbf{f} = (\mathbf{Td}, \mathbf{Sg}, \mathbf{Hs}, \mathbf{Tp})$ the vector of functional inputs in a time series format (see Section 2) and correspondingly, we denote by $\ddot{\mathbf{f}} = (\ddot{\mathbf{Td}}, \ddot{\mathbf{Sg}}, \ddot{\mathbf{Hs}}, \ddot{\mathbf{Tp}})$ the vector of shifted scalar representations of the elements in $\boldsymbol{f}$. From the physical perspective, the $\boldsymbol{Td}$ peak (its value at time 19) is the point in the series with the most influence on the output. Thus, we use that quantity as its scalar representation. On the other hand, for $\boldsymbol{Sg}$, $\boldsymbol{Hs}$ and $\boldsymbol{Tp}$ we use the average of the series over 37 time points, given their smooth and almost constant behavior in the historical dataset (see Figure 2). Using a similar notation to that used for the analytic case, we define the scalar benchmark metamodel as:

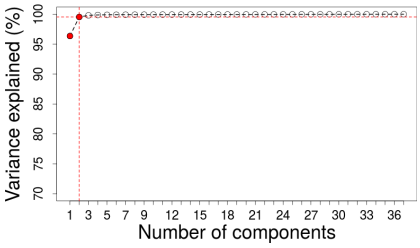
$$\begin{aligned} \mathcal{M}_{000} :\ & \mathbb{R}^4 \to \mathbb{R}, \\ & \ddot{\boldsymbol{f}} \mapsto \mathcal{M}_{000}(\ddot{\boldsymbol{f}}). \end{aligned} \tag{22}$$

As before, the funtional metamodels require the definition of a shifted version of $\boldsymbol{f}$ 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 which had only two. As mentioned earlier, in the proposed exploration method the number of experimental conditions grows exponentially with the number of functional inputs, and so does the processing time. In Section 4.1 we proposed an extension to deal with a larger number of structural parameters and levels. Such an extension would certainly be of service here. However, its development requires a considerable amount of additional work which is out of the scope of this paper. Thus, in this section we adopt the simpler approach of performing a classic principal component analysis to determine if any shifted functional input could be discarded from exploration (see Figure 9). Note that the plot is built for the shifted inputs as those are the ones that will potentially be used as functional inputs of the metamodel (see the setup for the analytic case in Section 4.2).

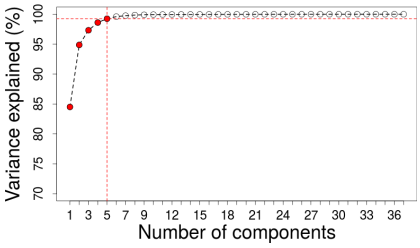


(a) PCA shifted *Td*.

(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 lines indicate the number of principal components required to explain at least the 99% of the data variability.

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. Considering this and the behavior of the time series in Figure 2, we decided to discard $\tilde{\boldsymbol{Hs}}$ as a functional input of the metamodel. However, we keep its scalar representation $\ddot{\boldsymbol{Hs}}$ active, as it does not affect the number of experimental conditions to run and may help to improve predictions. Although $\tilde{\boldsymbol{Td}}$ is also well described by just a few components, the tide is known to be a primary forcing factor of coastal flooding. Both, statistical and physical reasoning are relevant for this filtering process. Thus, we decided to keep $\tilde{\boldsymbol{Td}}$ in the experiment.

Now we let $\boldsymbol{\Pi} = (\Pi_1, \Pi_2, \Pi_3)$ denote the vector of projections of dimension p for $\tilde{\boldsymbol{Td}}$, $\tilde{\boldsymbol{Sg}}$ and $\tilde{\boldsymbol{Tp}}$. For every functional metamodel, we keep all the elements in $\ddot{\boldsymbol{f}}$ active and we add at least one element of $\boldsymbol{\Pi}$ as a functional input. Functional metamodels are defined the same way as for the analytic case. For instance, metamodels with (a) only Π_1 active, (b) Π_2 and Π_3 active, and (c) all three projections active, are defined in (23), (24) and (25), respectively.

$$\begin{aligned} \mathcal{M}_{f00}: \ & [0,1]^4 \times \mathbb{R}^p \to \mathbb{R}, \\ & (\ddot{\boldsymbol{f}}, \Pi_1) \mapsto \mathcal{M}_{f00}(\ddot{\boldsymbol{f}}, \Pi_1). \end{aligned} \tag{23}$$

$$\begin{aligned} \mathcal{M}_{0ff}: \ & [0,1]^4 \times \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}, \\ & (\ddot{\boldsymbol{f}}, \Pi_2, \Pi_3) \mapsto \mathcal{M}_{0ff}(\ddot{\boldsymbol{f}}, \Pi_2, \Pi_3). \end{aligned} \tag{24}$$

$$\begin{aligned} \mathcal{M}_{fff}: \ & [0,1]^4 \times (\mathbb{R}^p)^3 \to \mathbb{R}, \\ & (\ddot{\boldsymbol{f}}, \boldsymbol{\Pi}) \mapsto \mathcal{M}_{fff}(\ddot{\boldsymbol{f}}, \boldsymbol{\Pi}). \end{aligned} \tag{25}$$

The interpretation of this notation is analogous to that of the metamodels in the analytic case. The subscript of $\mathcal{M}$ indicates which functional decompositions are active. The notation remains generic in the sense that (23), (24) and (25), might all represent configurations involving diverse combinations of projection method, projection dimension and distance measure.

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 in Table 2, except for those where Π_1, Π_2 and Π_3 are simultaneously inactive, which are equivalent to the scalar metamodel. A detailed list of the 85 configurations and corresponding results is provided in Appendix D, Table D.9. As for the analytic case, we have considered the B-spline and PCA projection methods and set the projection dimension p to be the same for all active functional inputs.

Parameter	**Levels**
State of Π_1	inactive, active
State of Π_2	inactive, active
State of Π_3	inactive, active
Projection method	B-splines, PCA
Projection dimension	1, 2, 3
Distance	$\|\cdot\|_{D,\boldsymbol{\theta}}$, $\|\cdot\|_{S,\boldsymbol{\theta}}$

Table 2: Coastal flooding case: parameters and levels for the screening stage.

In the analytical example, we used an ordinary least squares formulation to set the projection coefficients, given that all points in the input series where considered equally important. For the RISCOPE application, the midpoint of the series is of particular relevance, as it corresponds to the moment of high tide. Therefore, in this case we used a weighted least squares formulation instead (see Appendix B). A constrained or weighted constrained formulation could also be suitable choices here. Those are used for further analysis in Section 5.2.

For the weighted least squares formulation we denote the vector of weights by $\boldsymbol{w} = (w_1, \ldots, w_T)$, with $T = 37$ for the application case. We set the value of each element in $\boldsymbol{w}$, based on the model

$$w_t = \begin{cases} 1 & : \text{ if } t = t_* \\ \lambda & : \text{ if } 0 < |t - t_*| \leq \delta \\ \lambda \exp\left(-\dfrac{(|t - t_*| - \delta)^2}{2\sigma^2}\right) & : \text{ if } |t - t_*| > \delta, \end{cases} \tag{26}$$

with $\sigma^2 = -(\omega^2)/(2\ln(\gamma))$ controlling the decay rate of the function.

Models like (26) are often used to represent the relevance of results for queries on search engines [61]. It retains some interesting properties from the Gaussian pdf, such as the non-negativity and the existence of at least one maximum located at the origin t_*. A particular shape can be given to the curve by setting the value of its parameters as follows. First, δ should be choosen from $[0, t_*]$. Then, ω should be set in $[0, t_* - \delta]$. Finally, λ and γ should be set, each in $[0, 1]$. This setting along with the first case of (26) ensure that the greatest possible score in $\boldsymbol{w}$ is 1. The weighting curve produced by such a model is illustrated in Figure 10 with the parameterization used for the coastal flooding case.

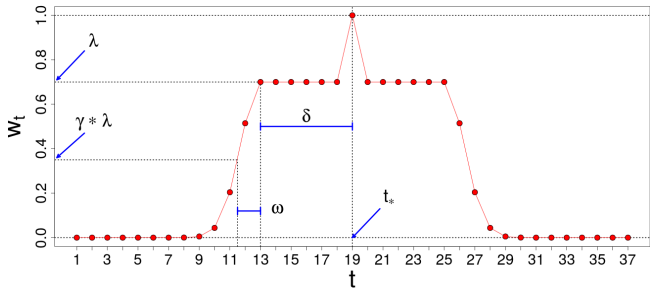
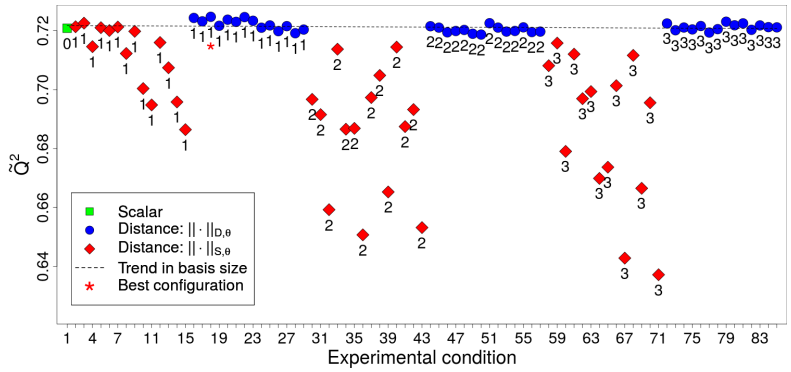


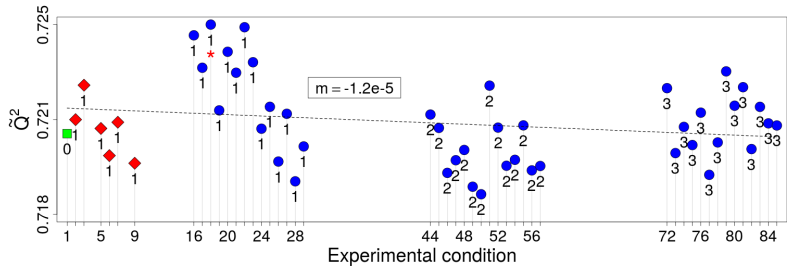
Figure 10: Weighting function for the coastal flooding case. The parameters λ, γ, ω and δ, controling the shape of the curve, were set to 0.7, 0.5, 1.5 and 6, respectively.

Figure 11 shows the $\tilde{Q}^2$ of the 85 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 $\tilde{Q}^2$ for each was computed. We used the same stopping conditions and parameters $(m*, z)$ as for the analytic case. That is $m* = 10^{-4}$ and $z = 3$. For convenience in the analysis, Figure 11 classifies metamodel configurations into three groups based on their performance:

(i) the scalar metamodel $\mathcal{M}_{000}$,
(ii) all metamodels using a decomposition-based distance $\|\cdot\|_{D,\boldsymbol{\theta}}$,
(iii) all metamodels using a scalar distance $\|\cdot\|_{S,\boldsymbol{\theta}}$.



(a) Coastal flooding case: all configurations of screening.



(b) Coastal flooding case: zoom to best configurations of screening. The value of m in the box indicates the slope of the linear least squares fitting of the points.

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

The scalar metamodel and the functional ones using the decomposition-based distance $\|\cdot\|_{D,\boldsymbol{\theta}}$ performed similarly and outperformed almost every configuration using the scalar distance $\|\cdot\|_{S,\boldsymbol{\theta}}$, except for a few ones with $p = 1$. Regarding the $\tilde{Q}^2$ trend, here it seems that lower p values work better. Since the slope of the linear fitting of the best configurations was smaller than the critical value $m = 10^{-4}$ during screening, we stop based on our second stopping condition. Thus, we stop the search and keep the current best configuration. Strictly speaking, such a configuration corresponds to experimental condition number 18; a metamodel with an active B-spline representation of size $p = 1$ for $\tilde{\boldsymbol{T}}\boldsymbol{d}$ and $\tilde{\boldsymbol{S}}\boldsymbol{g}$, using the decomposition-based distance $\|\cdot\|_{D,\boldsymbol{\theta}}$. However, in practice any of the dominant configurations included in Figure 11b could be a good choice, since the $\tilde{Q}^2$ of all that group of configurations was quite similar and processing times were all reasonable (see Appendix D, Table D.9).

It is inquiring to see that the $\tilde{Q}^2$ values reported in Figure 11 are quite moderate, even for the best configurations. Operationally, the problem is that almost in each of the 30 samples of each configuration, there is at least one of the 1500 validation points, whose prediction is significantly bad. To illustrate, in Figure 12 we report the squared error of the 1500 validation points for each of the 30 samples of metamodel configuration 18 — the best configuration of the coastal flooding case. In almost every sample a few points behave as outliers, increasing the sum of squared errors and thus, decreasing the $\tilde{Q}^2$.

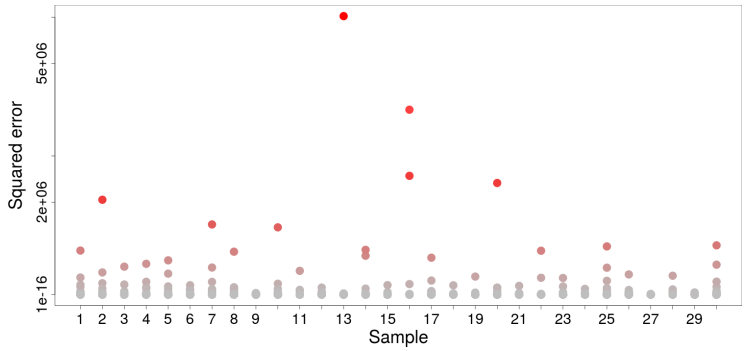
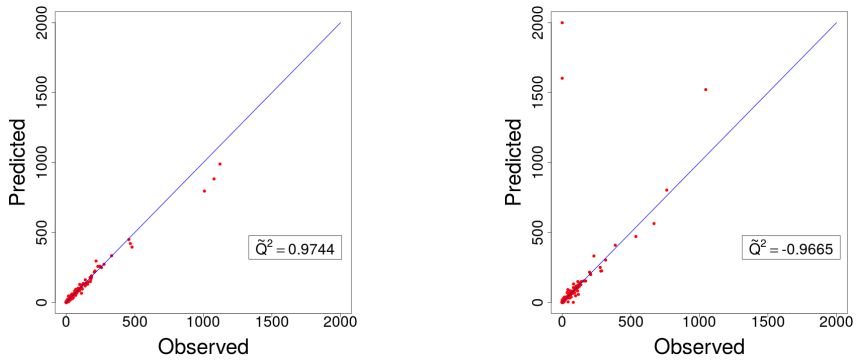


Figure 12: Coastal flooding case: squared errors for each of the 1500 validation points in each of the 30 samples of configuration 18.

We think that the problem comes from a strong imbalance in the dataset between mild events (leading to minor or no flooding) and strong events (leading to major flooding). In fact, after simulating each of the 20557 available hindcast events (see Section 2), we found 90% of the output values below $4\,\mathrm{m}^3$ although the largest output value found was $3455\,\mathrm{m}^3$. This proportion of mild events has total physical sense as most part of the time Gâvres is not flooded. In that sense, strong events — like Johanna storm, which hit Gâvres in 2008 — are rather statistically uncommon. However, this natural bias impacts the efficiency of metamodel training, as the majority of learning data will match mild events (see Figure 13). A possible way to deal with this issue is to use sequential design techniques [62, 63] to dynamically add events to the learning set, seeking to diminish the bias in the data. Further analysis on this issue is out of the scope of this paper, but will be addressed in the upcoming steps of the project.



(a) Coastal flooding: best sample.

(b) Coastal flooding: worst sample.

Figure 13: Coastal flooding case: calibration plot with 1500 data points for the best and worst samples of the best performing metamodel (configuration 18).

One may say that one or two bad predictions over 1500 is not exactly a bad performance. Here the issue is that the $\tilde{Q}^2$ is overly affected by only these few very large errors. Therefore, in this case, a more robust and appropiate way to assess the absolute quality of each metamodel is to compute the $\ddot{Q}^2$, which we define as the $\tilde{Q}^2$ in (21), but using $\ddot{\sigma}^2_E = \text{median}\left(\{(y_{*,i} - \hat{y}_{*,i})^2\}_{i=1,\dots,n_*}\right)$ instead of σ^2_E and $\ddot{\sigma}^2_T = \text{median}\left(\{(y_{*,i} - \bar{y}_*)^2\}_{i=1,\dots,n_*}\right)$ instead of σ^2_T. Here $\text{median}\left(\{u_1, \dots, u_a\}\right)$ is the empirical median of $u_1, \dots, u_a \in \mathbb{R}$. Configurations 18 and 71, the best and worst metamodels of the screening, reported $\tilde{Q}^2$ values of 0.7247 and 0.6371, respectively. In contrast, if we compute their $\ddot{Q}^2$, we obtain 0.9999857 and 0.9997, in the same order. Hence, the metamodel predictions are accurate for the large majority of the elements in the testbase.

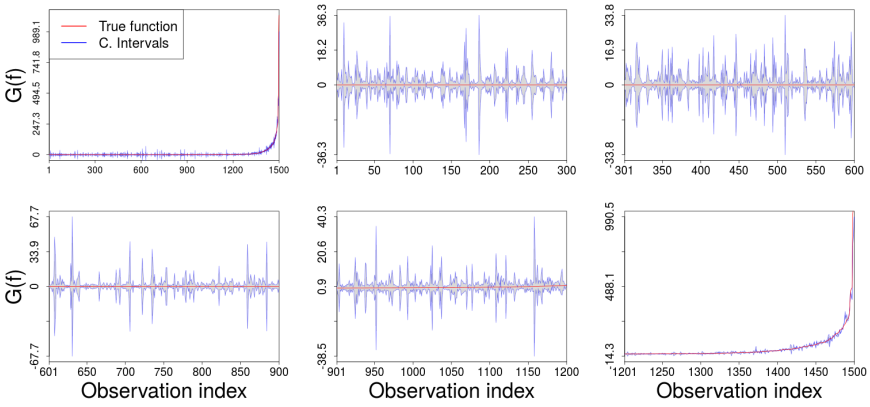


Figure 14: Coastal flooding case: fitting of the best performing sample of the best configuration. Left-top subplot illustrates the whole set of 1500 output values in increasing order; this subplot is then subdivided to generate the remaining 5 subplots by splitting the abscissa into 5 sections and zooming the resulting figures.

The fitting of the ordered true output for the best performing sample of the best configuration is illustrated in Figure 14. For this sample, the proportion of output values lying within confidence intervals at 99%, 95% and 90% was 92%, 91% and 89%, respectively. In this case, the plot shows more variability in the

confidence intervals than for the analytic case. However, this is just a visual matter as the scale of each subplot is custom to the range of the output values at each segment. Based on the results, we may conclude that the selected metamodel provides good predictions for the majority of data and could be improved by means of strategic sampling/sequential design techniques.

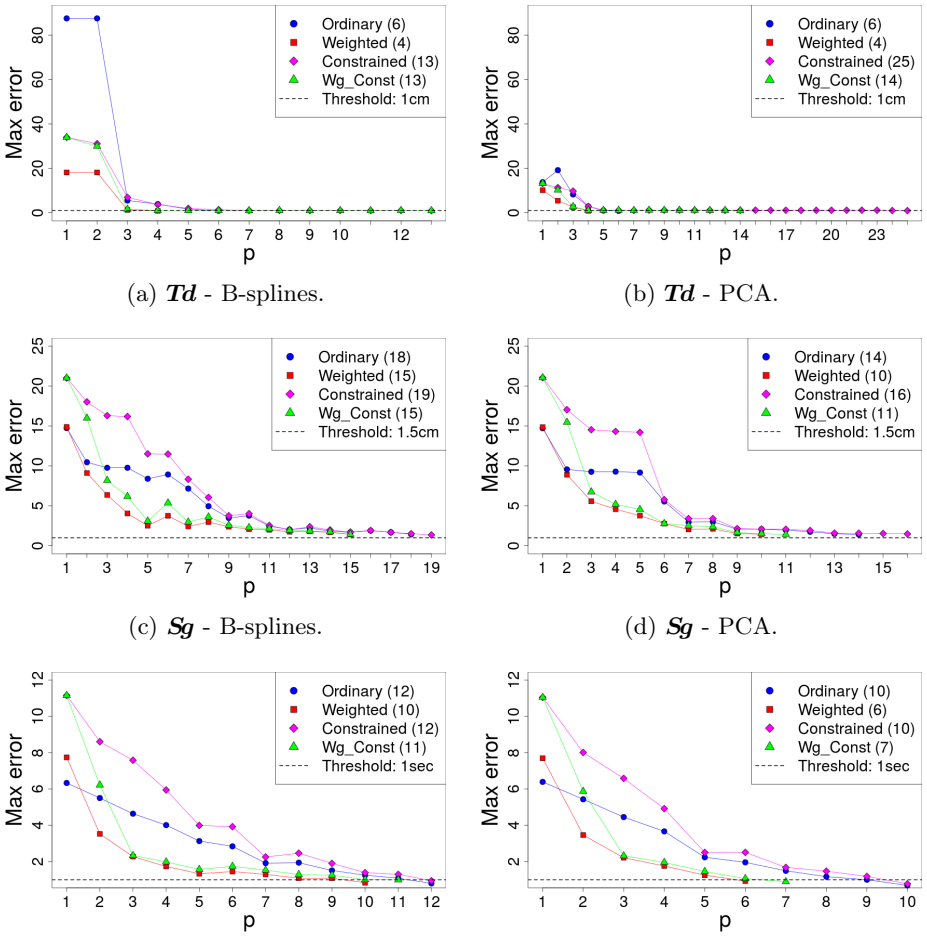
Interestingly, whilst the inputs of the code are time series of dimension 37, it was possible to achieve quite good metamodel predictability just by using a scalar representation of them. This result may be explained by the fact that, as commented in Section 2.1, the simplified code considered here can be seen as the sum of independent scalar-to-scalar problems. Thus, if we represent the functional inputs by a scalar related 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 was intrinsically functional as in the analytic case studied in Section 4.2. There, the best configuration found was a metamodel using a projection of dimension 5.

5.2. Dimension selection based on projection error

Earlier in the paper, we have commented the common practice in the literature to set up the projection dimension p, which is using the accuracy of the projection itself as criterion. The problem with this approach, as mentioned earlier, 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 p based on the projection error, and then we assess the performance of the corresponding metamodels based on their $\tilde{Q}^2$. Finally, we compare results with those of the metamodels assessed in the frame of our exploration methodology.

5.2.1. Selecting the dimension for each input based on projection error

Here we follow an approach which consists in the definition of an error tolerance for each input, and the posterior search of the lowest dimension for which the tolerance is reached. Based on knowledge of the coastal flooding phenomenon, we set a maximum error of 1 cm, 1.5 cm, and 1 s, for the projections of ***Td***, ***Sg*** and ***Tp***, respectively. This tolerance should be achieved within the critical time window $t = \{13, \ldots, 25\}$, which corresponds to the moment of maximum tide $\pm$ 1 hour. Hence, the procedure will point out to find the lowest projection dimension, for which every curve of the hindcast dataset satisfies the stated tolerance. We also use this experiment to compare the four least squares formulations presented in Appendix B to set the coefficients of the projection, those being: the ordinary, weighted, constrained and weighted-constrained formulation. The last three, could be interesting choices here as the points of the series lying in the critical time window have greater importance than the others; in particular at point t_{19}. Results are condensed in Figure 15.



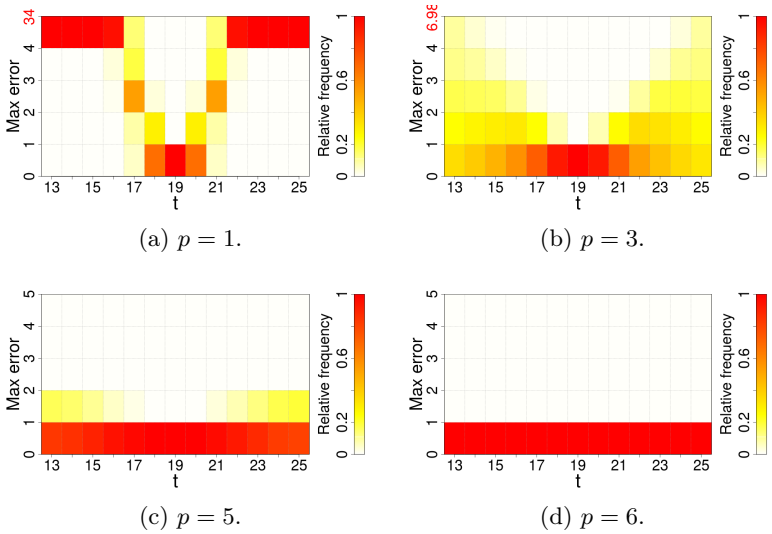
(a) ***Td*** - B-splines. (b) ***Td*** - PCA.

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

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

Figure 15: Maximum projection error within the critical time window as a function of the projection dimension p. ***Td*** and ***Sg*** error in centimeters and ***Tp*** error in seconds. In the legends, the quantity in parenthesis indicates the value of p needed to meet the tolerance.

As suggested in Section 4, by focusing on the error of the projection one may end up with an unnecessarily large projection dimension. For instance, the projection error for ***Td*** using any of the formulations was already quite low at $p = 6$, however, the constrained and weighted-constrained formulations required at least $p = 13$ to reach the tolerance. What makes it even worst is that usually, there are only a couple of curves in the dataset that require such a high projection dimension. For instance, in Figure 16 we show how for the B-splines projection method and the constrained formulation, almost all the curves of ***Td*** had already reached the tolerance at $p = 6$. However, seven additional dimensions where required to be compliant for all the curves. 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 in Figure 15e. At $p = 5$ the error was considerably low, however, it required $p = 10$ to meet the tolerance.



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

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

Figure 16: Distribution of errors (in centimeters) for points in the critical window using the constrained least quared optimization formulation and B-splines for the projection of ***Td***.

5.2.2. Efficiency of configurations based on projection error

The weighted formulation was the best performing one in the experiment above. It required the lowest dimension in all cases and its maximum error remained almost always below that of all the other formulations. Therefore, in this section we assess the performance of the metamodel using the projection dimension suggested by that formulation. Similarly to the previous experiments, here we evaluate all the possible combinations of the following structural parameters: state of each projection (inactive or active), projection method (B-splines or PCA) and distance ($\|\cdot\|_{S,\boldsymbol{\theta}}$ or $\|\cdot\|_{D,\boldsymbol{\theta}}$). In this case the projection dimension p is not taken as a factor of the experiment, as its values are taken from results of the selection based on projection error. Those values are listed in Table 3 for each combination of input and projection method. In addition, in this experiment we do not consider the case where all functional decompositions are inactive, as it corresponds to the scalar metamodel, which we already evaluated as part of the selection based on metamodel predictability in Section 5.1.

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

Table 3: Selected dimension based on projection error.

A total of 28 experimental conditions were evaluated this time. A detailed list of them and their corresponding results is provided in Appendix D, Table D.10. The $\tilde{Q}^2$ of each of these experimental conditions is reported in Figure 17, along with that of the best configuration found with the approach based on metamodel predictability. To recall, the latter corresponds to configuration 18, which has an active B-spline representation of size $p = 1$ only for ***Td***, using the decomposition-based distance $\|\cdot\|_{D,\boldsymbol{\theta}}$.

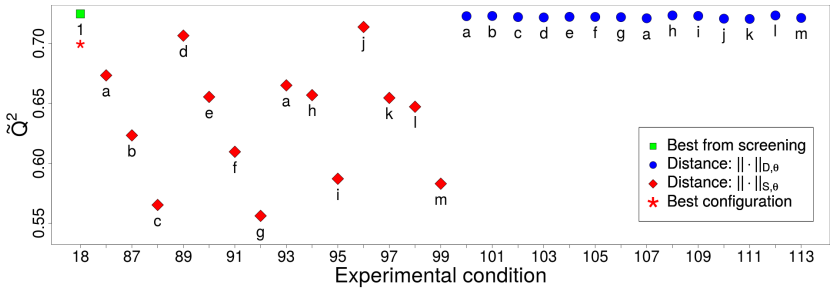


Figure 17: Coastal flooding case: performance of metamodels with the projection dimension based on projection error. Points are labeled by projection dimension; as it varies from one input to the other in the new experiment, those points are labeled using an alphabetic convention where each letter is 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 distance $\|\cdot\|_{D,\boldsymbol{\theta}}$ were better than those with the scalar one. The former has consistently been showing better performance throughout the paper. Its advantage over the scalar distance is presumably the fact that it keeps the number of length-scale parameters controlled, which in turn maintains the learning problem handy and so its resolution time. Conversely, the scalar distance implies in some cases many more hyperparameters. To illustrate, the average training time of the fourteen configurations reported in Figure 17 for the scalar distance was 384.7 s, which corresponds to 10 times the average training time of configuration 18. In contrast, the same quantity for the fourteen configurations using the decomposition-based distance was 56.7 s, or 1.5 times the average training time of configuration 18.

None of the new metamodels outperformed the best configuration found by means of our exploration methodology. Metamodels selected with the approach based on the projection error are in general more comptationally demanding. Our exploration strategy eludes this problem by only increasing the projection dimension if there is evidence of potential improvement in accuracy.

6. Robustness to changes in the amount of training/validation data

We close the experimental segment of the article with an analysis on the behavior of our exploration methodology when different amounts of training and validation data are used. In Sections 4.2 and 5.1, the exploration strategy was used to calibrate the structural parameters of a metamodel for the analytic case study and the coastal flooding application, respectively. In both cases, we used training sets of size 800 and validation sets of size 1500. Those numbers were selected during a preliminary verification of functionality of our codes, taking into consideration the numerical stability of the metamodel (e.g., when computing inverse correlation matrices), its predictability and the stability of performance statistics for it. However, the number of training and validation points are undoubtedly influential factors in regression and also in model selection. If different numbers of training and validation sets are chosen, the performance statistics of any metamodel previously assessed will most likely change, so will the optimal choice of a metamodel configuration. Then, a critical question is

if the exploration methodology under use is robust to such changes. In other words, if it is able to efficiently identify good metamodel configurations when we change the amount of information available to train and validate. In this section we conduct an experiment to test this attribute of our exploration methodology.

6.1. Experiment setting

The experiment is based on the two case studies previosly addressed in the paper. For each of them, we search the optimal metamodel configuration, given different amounts of training and validation data. To do so, we use an exhaustive search (ES) approach, where we evaluate all possible combinations of the structural parameters. Then, we use the data generated by ES ($\tilde{Q}^2$ of each configuration) and emulate the exploration process using our search methodology, which we will refer to in this section as SS, standing for *strategic search*. Finally, we assess the performance of SS by comparisson against ES.

To keep the experiment tractable, for each case study we consider a solution space including all levels of the structural parameters already evaluated, except for the projection dimension. For this latter, we only explore a range of levels large enough to cover all metamodel configurations assessed in Sections 4.2 and 5. Thus, for the new experiment we make the ES method explore all configurations with projections of dimension up to 8 for the analytic case and up to 4 for the coastal flooding case. Conversely, we make the SS method run until fulfilling one of the convergence-oriented stopping conditions used in Sections 4.2 and 5 (with $m* = 10^-4$ and $z = 3$), or until reaching a corner of the solution space.

6.2. Performance statistics

In this paper we evaluate our exploration strategy in terms of solution quality and runtime. To do so, we define the following two indicators:

- *Optimality gap.* Relative difference between the $\tilde{Q}^2$ of the optimal solution found by the ES method and that of the solution found by our SS method:

$$\Delta\tilde{Q}^2 := \frac{\tilde{Q}^2_{\text{ES}} - \tilde{Q}^2_{\text{SS}}}{\tilde{Q}^2_{\text{ES}}} \times 100\%, \tag{27}$$

  with $\tilde{Q}^2_{\text{ES}}$ and $\tilde{Q}^2_{\text{SS}}$ denoting the $\tilde{Q}^2$ of the best solution for the ES and the SS method, respectively, computed with (21).

- *Time saving.* Relative difference between the runtime of the ES method and that of our SS method:

$$\Delta\text{Time} := \frac{T_{\text{ES}} - T_{\text{SS}}}{T_{\text{ES}}} \times 100\%, \tag{28}$$

  where T_{ES} denotes the sum of training and validation times of all the configurations evaluated by the ES method, and similarly for T_{SS}.

The $\tilde{Q}^2$ values, training times and validation times recovered by ES are recycled by SS in order to have a fair comparisson among exploration methods. To preserve the legitimacy of the results, the optimal configuration is kept unknown until SS has been run and formal stopping conditions are used. Similarly as before, 30 pairs of training and validation points are used to account for noise.

6.3. Analysis

Statistics for the analytic and the coastal flooding case are presented in Tables 4 and 5, respectively. The results suggest the following findings: (1) the number of training points has greater impact on the optimal combination of structural parameters than the number of validation points. Configurations tend to vary more between rows than between columns; (2) greater number of training points leads to selection of configurations with larger projection dimension when the problem is intrinsically functional, as in the analytic case; (3) the decomposition-based distance $\|\cdot\|_{D,\boldsymbol{\theta}}$ could be in general a better choice than the scalar distance $\|\cdot\|_{S,\boldsymbol{\theta}}$. The former was the optimal choice in almost all cases except for two instances for the coastal flooding application.

Tr \ Val	500 (ES)	500 (SS)	1000 (ES)	1000 (SS)	1500 (ES)	1500 (SS)	2000 (ES)	2000 (SS)
100	B − 3	B − 3	B − 3	B − 3	B − 3	B − 3	B − 3	B − 3
	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$
	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$
	$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%	
	ΔTime: 76.7%		ΔTime: 75.4%		ΔTime: 73.7%		ΔTime: 73.1%	
	T_{ES}: 34.2 m	T_{SS}: 8.0 m	T_{ES}: 47.1 m	T_{SS}: 11.6 m	T_{ES}: 62.6 m	T_{SS}: 16.5 m	T_{ES}: 81.1 m	T_{SS}: 21.8 m
400	P − 4	P − 4	P − 8	P − 6	P − 4	P − 4	P − 8	P − 4
	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$
	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$
	$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 1.1e−7%		$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 6.4e−8%	
	ΔTime: 66.2%		ΔTime: 65.7%		ΔTime: 65.1%		ΔTime: 64.4%	
	T_{ES}: 8.4 h	T_{SS}: 2.8 h	T_{ES}: 8.7 h	T_{SS}: 3.0 h	T_{ES}: 9.1 h	T_{SS}: 3.2 h	T_{ES}: 9.8 h	T_{SS}: 3.5 h
700	P − 5	P − 5	B − 6	B − 6	B − 6	B − 6	B − 6	B − 6
	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$
	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$
	$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%	
	ΔTime: 68.0%		ΔTime: 67.8%		ΔTime: 67.5%		ΔTime: 67.1%	
	T_{ES}: 36.9 h	T_{SS}: 11.8 h	T_{ES}: 37.4 h	T_{SS}: 12.1 h	T_{ES}: 38.3 h	T_{SS}: 12.4 h	T_{ES}: 39.3 h	T_{SS}: 13.0 h
1000	B − 6	B − 6	B − 6	B − 6	B − 6	B − 6	B − 6	B − 6
	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$
	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$	$\|\cdot\|_{D,\boldsymbol{\theta}}$
	$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%	
	ΔTime: 69.1%		ΔTime: 69.0%		ΔTime: 68.8%		ΔTime: 68.5%	
	T_{ES}: 4.3 d	T_{SS}: 1.3 d	T_{ES}: 4.3 d	T_{SS}: 1.3 d	T_{ES}: 4.4 d	T_{SS}: 1.4 d	T_{ES}: 4.4 d	T_{SS}: 1.4 d

Table 4: Analytic case: robustness to changes in the amount of training and validation data. Each intersection contains the configuration selected by (i) the ES method (darker colored cell ■) and (ii) the SS method (lighter colored cell □), as well as the performance statistics computed with (27) and (28). The convention used to denote a configuration is to divide its components in three lines. First line contains the projection method (P: PCA, B:B-splines) and dimension $(1, \ldots, 8)$ separated by a script. Second line indicates the active functional inputs ($\mathcal{M}_{00}$, $\mathcal{M}_{0f}$, $\mathcal{M}_{f0}$, $\mathcal{M}_{ff}$). Finally, the third line indicates the type of distance ($\|\cdot\|_{S,\boldsymbol{\theta}}$, $\|\cdot\|_{D,\boldsymbol{\theta}}$). For example, the configuration selected by ES for 400 training and 500 validation points is a metamodel with a B-splines projection of dimension 3 for both functional inputs, using the distance $\|\cdot\|_{D,\boldsymbol{\theta}}$. T_{ES} and T_{SS} are provided in minutes (m), hours (h) and days (d).

Regarding the performance of the exploration methodology proposed in this paper, results show that: (1) the methodology is robust to changes in both, the

amount of training and validation data. Its configuration choice was optimal in the vast majority of cases. In the remaining ones, the optimality gap $\Delta\tilde{Q}^2$ was always negligible (worst case in the order of 1e−2%); (2) our exploration strategy provides an efficient way to solve the problem of structural parameter calibration. It caused time savings of at least 64.4% for the analytic case and 28.7% for the coastal flooding case. We remark that the times reported in Tables 4 and 5 are the sums of training and validation times of the 30 samples of every configuration evaluated by each exploration method (ES and SS). At each combination of number of training and validation points, ES evaluated 97 configurations in the analytic case and 113 in the coastal flooding case. For the instance with 1000 training points and 2000 validation points, SS evaluated 47 configurations in the analytic case and 85 in the coastal flooding case. This gives average metamodel construction times (training and validation) for the analytic case of 2.19 min for ES and 1.42 min for SS. For the coastal flooding case, the average construction times are 2.08 min for ES and 1.71 min for SS.

Tr \ Val	500		1000		1500		2000	
100	NA $\mathcal{M}_{000}$ NA	NA $\mathcal{M}_{000}$ NA	NA $\mathcal{M}_{000}$ NA	NA $\mathcal{M}_{000}$ NA	NA $\mathcal{M}_{000}$ NA	NA $\mathcal{M}_{000}$ NA	NA $\mathcal{M}_{000}$ NA	NA $\mathcal{M}_{000}$ NA
	$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%	
	ΔTime: 30.7%		ΔTime: 30.7%		ΔTime: 30.3%		ΔTime: 28.7%	
	T_{ES}: 31.8 m	T_{SS}: 22.1 m	T_{ES}: 43.7 m	T_{SS}: 30.3 m	T_{ES}: 61.4 m	T_{SS}: 42.8 m	T_{ES}: 79.9 m	T_{SS}: 56.9 m
400	B − 4 $\mathcal{M}_{00f}$ $\|\cdot\|_{D,\boldsymbol{\theta}}$	P − 3 $\mathcal{M}_{00f}$ $\|\cdot\|_{D,\boldsymbol{\theta}}$	B − 4 $\mathcal{M}_{00f}$ $\|\cdot\|_{D,\boldsymbol{\theta}}$	P − 3 $\mathcal{M}_{00f}$ $\|\cdot\|_{D,\boldsymbol{\theta}}$	B − 4 $\mathcal{M}_{00f}$ $\|\cdot\|_{D,\boldsymbol{\theta}}$	P − 3 $\mathcal{M}_{00f}$ $\|\cdot\|_{D,\boldsymbol{\theta}}$	B − 4 $\mathcal{M}_{00f}$ $\|\cdot\|_{D,\boldsymbol{\theta}}$	P − 3 $\mathcal{M}_{00f}$ $\|\cdot\|_{D,\boldsymbol{\theta}}$
	$\Delta\tilde{Q}^2$: 4.0e−2%		$\Delta\tilde{Q}^2$: 9.7e−5%		$\Delta\tilde{Q}^2$: 9.7e−5%		$\Delta\tilde{Q}^2$: 6.5e−3%	
	ΔTime: 36.3%		ΔTime: 35.3%		ΔTime: 35.4%		ΔTime: 34.8%	
	T_{ES}: 8.9 h	T_{SS}: 5.7 h	T_{ES}: 9.9 h	T_{SS}: 6.4 h	T_{ES}: 9.8 h	T_{SS}: 6.3 h	T_{ES}: 10.6 h	T_{SS}: 6.9 h
700	P − 1 $\mathcal{M}_{0f0}$ $\|\cdot\|_{D,\boldsymbol{\theta}}$	P − 1 $\mathcal{M}_{0f0}$ $\|\cdot\|_{D,\boldsymbol{\theta}}$	P − 1 $\mathcal{M}_{f0f}$ $\|\cdot\|_{D,\boldsymbol{\theta}}$	P − 1 $\mathcal{M}_{f0f}$ $\|\cdot\|_{D,\boldsymbol{\theta}}$	P − 1 $\mathcal{M}_{f0f}$ $\|\cdot\|_{D,\boldsymbol{\theta}}$	P − 1 $\mathcal{M}_{f0f}$ $\|\cdot\|_{D,\boldsymbol{\theta}}$	P − 1 $\mathcal{M}_{f00}$ $\|\cdot\|_{D,\boldsymbol{\theta}}$	P − 1 $\mathcal{M}_{f00}$ $\|\cdot\|_{D,\boldsymbol{\theta}}$
	$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%	
	ΔTime: 34.6%		ΔTime: 34.5%		ΔTime: 34.2%		ΔTime: 33.9%	
	T_{ES}: 39.7 h	T_{SS}: 26.0 h	T_{ES}: 40.4 h	T_{SS}: 26.5 h	T_{ES}: 41.4 h	T_{SS}: 27.2 h	T_{ES}: 42.6 h	T_{SS}: 28.1 h
1000	P − 1 $\mathcal{M}_{00f}$ $\|\cdot\|_{D,\boldsymbol{\theta}}$	P − 1 $\mathcal{M}_{00f}$ $\|\cdot\|_{D,\boldsymbol{\theta}}$	B − 2 $\mathcal{M}_{0f0}$ $\|\cdot\|_{S,\boldsymbol{\theta}}$	B − 2 $\mathcal{M}_{0f0}$ $\|\cdot\|_{S,\boldsymbol{\theta}}$	P − 1 $\mathcal{M}_{f0f}$ $\|\cdot\|_{D,\boldsymbol{\theta}}$	P − 1 $\mathcal{M}_{f0f}$ $\|\cdot\|_{D,\boldsymbol{\theta}}$	B − 3 $\mathcal{M}_{00f}$ $\|\cdot\|_{S,\boldsymbol{\theta}}$	B − 3 $\mathcal{M}_{00f}$ $\|\cdot\|_{S,\boldsymbol{\theta}}$
	$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%	
	ΔTime: 38.7%		ΔTime: 38.6%		ΔTime: 38.4%		ΔTime: 38.4%	
	T_{ES}: 4.7 d	T_{SS}: 2.9 d	T_{ES}: 4.8 d	T_{SS}: 2.9 d	T_{ES}: 4.8 d	T_{SS}: 3.0 d	T_{ES}: 4.9 d	T_{SS}: 3.0 d

Table 5: Coastal flooding case: robustness to changes in the amount of training and validation data. Each intersection contains the configuration selected by (i) the ES method (darker colored cell ■) and (ii) the SS method (lighter colored cell □), as well as the performance statistics computed with (27) and (28). The convention is analogous to that of Table 4, except that the projection dimension takes values from $(1, \ldots, 4)$. T_{ES} and T_{SS} are provided in minutes (m), hours (h) and days (d).

7. Conclusions

In this article we propose a methodology to simultaneously tune multiple characteristics of a functional-input metamodel. Its construction is motivated by a coastal flooding application where a surrogate of a functional-input hydrodynamic code is to be built for early warning. The nature of the inputs gives rise to a number of questions about the proper way to represent them in the metamodel: which inputs should be kept as predictors, what is a good method to reduce their dimension, which dimension is ideal, and given our choice to work with Gaussian process metamodels which are kernel based methods, also the question of which is a convenient distance to meassure similarities between functional input points. The proposed methodology is intended to find dominant combinations of these types of features of the metamodel, which we call structural parameters. One of its main features is the possibility to calibrate the projection of the inputs (method and dimension) based on metamodel predictability, rather than projection error which is the common approach.

The proposed methodology works in a staged fashion. First it explores some low levels of projection dimension with all possible combinations of the remaining structural parameters. From there, the exploration evolves by detecting trends and patterns indicating the direction of improvement on the performance of the metamodel. Dominated levels of the structural parameters are discarded along the way to speed up the exploration. The exploration ends when a potential local optimum is detected or the performance statistic reaches certain degree of convergence.

While relatively simple, the proposed methodology proved its effectiveness through a theoretical case study and our coastal flooding application. In both cases it allowed to find metamodel configurations of outstanding performance able to accurately predict the output of the numerical model. The ideal projection method and projection dimension proved to vary from one application to the other, and even for different instances of the same application. For instance, for intrinsic functional problems where the output of the code cannot be reconstructed by iterative scalar-input runs, greater number of training points seem to lead to the selection of larger projection dimensions. Regarding the distance to meassure similarity among functional input points, the decomposition-based distance $\|\cdot\|_{D,\boldsymbol{\theta}}$ consistently reported better results than the scalar distance $\|\cdot\|_{S,\boldsymbol{\theta}}$ throughout the experiments. Apparently, decomposition-based distance is a useful alternative to integrate functional inputs in a kernel-based model while keeping the complexity of the learning process manageable.

Interestingly, our application case made evident that even when the inputs of the code are functional, it could be possible to obtain good predictions just by using a scalar representation of them. Whether this is the case will depend on the way the numerical model exploits the inputs to produce the outputs. Therefore, our main premise throughout the article has been that dimensionality reduction of the inputs should be mainly guided by metamodel performance. Our comparison with an approach based on a tolerance of projection error illustrated how this type of approach may lead to an unnecessarily large projection dimension. Depending on other metamodeling choices, such as the type of distance used to measure similarities between functional input points, large projection dimensions may imply a significant increase in processing time, not justified by any improvement in prediction accuracy.

The proposed methodology showed its efficiency through an experiment where it was compared to an exhaustive search approach. Our method was able to find an attractive solution while saving up to 76.7% and 38.7% of the time spent by the exhaustive search in the analytic case and coastal flooding case, respectively. The solution found by our methodology was optimal in most cases. A critical factor on its efficiency is that it applies the principles of exploration and exploitation present in many classical meta-heuristics such as Genetic Algorithms [64] and Ant Colony Optimization [65]. The first principle points out to start the exploration with a screening of a wide variety of metamodel configurations. Then, the second principle leads to concentrate around the best solutions so far and seek for local improvements. Given the positive results obtained in this study, an interesting research avenue would be the extension of the proposed methodology towards an heuristic-based optimization algorithm. Other studies in the field of computer experiments have pointed out this possibility as well [11].

Another potential direction of research is to develop one of the functional-input regression methods cited in the introduction. The extension of scalar-on-function techniques to nonlinear settings could be achieved, for instance, by defining penalized likelihood and cross validation formulations [13]. This would pave the road to the selection of the relevant components of the inputs during the optimization of the hyperparameters for powerful non-linear metamodels such as Gaussian processes.

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 [66] or try to adapt the work of [67] 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 bathymetric data used to set up the cross-shore numerical model. Finally, we would like to thank the anonymous reviewers whose comments significantly enchanced the quality of this work.

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

[7] Santner TJ, Williams BJ, Notz W, Williams BJ. The design and analysis of computer experiments. vol. 1. Springer; 2003.

[8] Lataniotis C, Marelli S, Sudret B. The Gaussian process modelling module in UQLab. Soft Computing in Civil Engineering. 2018;2(3):91–116.

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

[10] 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.

[11] Mai CV, Sudret B. Surrogate models for oscillatory systems using sparse polynomial chaos expansions and stochastic time warping. SIAM/ASA Journal on Uncertainty Quantification. 2017;5(1):540–571.

[12] Vincent P, Larochelle H, Bengio Y, Manzagol PA. Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on Machine learning. ACM; 2008. p. 1096–1103.

[13] Reiss PT, Goldsmith J, Shang HL, Ogden RT. Methods for scalar-on-function regression. International Statistical Review. 2017;85(2):228–249.

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

[15] 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.

[16] Constantine PG. Active subspaces: Emerging ideas for dimension reduction in parameter studies. vol. 2. SIAM; 2015.

[17] Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. science. 2006;313(5786):504–507.

[18] Damianou A, Lawrence N. Deep gaussian processes. In: Artificial Intelligence and Statistics; 2013. p. 207–215.

[19] Huang W, Zhao D, Sun F, Liu H, Chang E. Scalable gaussian process regression using deep neural networks. In: Twenty-Fourth International Joint Conference on Artificial Intelligence; 2015. p. 3576 – 3582.

[20] Calandra R, Peters J, Rasmussen CE, Deisenroth MP. Manifold Gaussian processes for regression. In: 2016 International Joint Conference on Neural Networks (IJCNN). IEEE; 2016. p. 3338–3345.

[21] Fornasier M, Schnass K, Vybiral J. Learning functions of few arbitrary linear parameters in high dimensions. Foundations of Computational Mathematics. 2012;12(2):229–262.

[22] Lataniotis C, Marelli S, Sudret B. Extending classical surrogate modelling to ultrahigh dimensional problems through supervised dimensionality reduction: a data-driven approach; 2019. Available from: `https://arxiv.org/abs/1812.06309`.

[23] Sacks J, Welch WJ, Mitchell TJ, Wynn HP. Design and analysis of computer experiments. Statistical science. 1989;4:409–423.

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

[25] Rasmussen CE. Gaussian processes in machine learning. In: Summer School on Machine Learning. Springer; 2003. p. 63–71.

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

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

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

[29] Van der Meer J, Allsop N, Bruce T, De Rouck J, Kortenhaus A, Pullen T, et al.. EurOtop: Manual on wave overtopping of sea defences and related sturctures: an overtopping manual largely based on European research, but for worlwide application; 2016.

[30] Idier D, Rohmer J, Pedreros R, Le Roy S, Lambert J, Louisor J, et al. Coastal flood: a composite method for past events characterisation providing insights in past, present and future hazards. In: AGU Fall Meeting 2019. AGU; 2019. .

[31] 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.

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

[33] 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.

[34] Pulido HG, De la Vara Salazar R, González PG, Martínez CT, Pérez MdCT. Análisis y diseño de experimentos. New York: McGraw-Hill; 2012.

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

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

[37] Ripley BD. Spatial statistics. vol. 575. John Wiley & Sons; 2005.

[38] Bachoc F. Cross validation and maximum likelihood estimations of hyper-parameters of Gaussian processes with model misspecification. Computational Statistics & Data Analysis. 2013;66:55–69.

[39] Green J, Whalley JL, Johnson CG. Automatic programming with ant colony optimization. In: Proceedings of the 2004 UK Workshop on Computational Intelligence. Loughborough University; 2004. p. 70–77.

[40] Karaboga D, Ozturk C, Karaboga N, Gorkemli B. Artificial bee colony programming for symbolic regression. Information Sciences. 2012;209:1–15.

[41] Uy NQ, Hoai NX, O'Neill M, McKay RI, Galván-López E. Semantically-based crossover in genetic programming: application to real-valued symbolic regression. Genetic Programming and Evolvable Machines. 2011;12(2):91–119.

[42] Maatouk H, Bay X. A new rejection sampling method for truncated multivariate Gaussian random variables restricted to convex sets. In: Monte carlo and quasi-monte carlo methods. Springer; 2016. p. 521–530.

[43] López-Lopera AF, Bachoc F, Durrande N, Roustant O. Finite-dimensional Gaussian approximation with linear inequality constraints. SIAM/ASA Journal on Uncertainty Quantification. 2018;6(3):1224–1255.

[44] Rougier J. Efficient emulators for multivariate deterministic functions. Journal of Computational and Graphical Statistics. 2008;17(4):827–843.

[45] Marrel A, Iooss B, Da Veiga S, Ribatet M. Global sensitivity analysis of stochastic computer models with joint metamodels. Statistics and Computing. 2012;22(3):833–847.

[46] Picheny V, Ginsbourger D, Richet Y, Caplin G. Quantile-based optimization of noisy computer experiments with tunable precision. Technometrics. 2013;55(1):2–13.

[47] Moutoussamy V, Nanty S, Pauwels B. Emulators for stochastic simulation codes. ESAIM: Proceedings and Surveys. 2015;48:116–155.

[48] Browne T, Iooss B, Gratiet LL, Lonchampt J, Remy E. Stochastic simulators based optimization by Gaussian process metamodels – application to maintenance investments planning issues. Quality and Reliability Engineering International. 2016;32(6):2067–2080.

[49] 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.

[50] De Boor C. A Practical Guide to Splines. In: Applied mathematical sciences. vol. 27. Heidelberg: Springer; 1978. p. 15–16.

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

[52] Jolliffe I. Principal component analysis. Springer; 2011.

[53] Papaioannou I, Ehre M, Straub D. PLS-based adaptation for efficient PCE representation in high dimensions. Journal of Computational Physics. 2019;387:186–204.

[54] 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.

[55] Roustant O, Ginsbourger D, Deville Y. Dicekriging, Diceoptim: Two R packages for the analysis of computer experiments by kriging-based metamodelling and optimization. Journal of Statistical Software. 2012;51(1):54p.

[56] Nilsson J, de Jong S, Smilde AK. Multiway calibration in 3D QSAR. Journal of Chemometrics: A Journal of the Chemometrics Society. 1997;11(6):511–524.

[57] Shen Z, Toh KC, Yun S. An accelerated proximal gradient algorithm for frame-based image restoration via the balanced approach. SIAM Journal on Imaging Sciences. 2011;4(2):573–596.

[58] Taillard E. Some efficient heuristic methods for the flow shop sequencing problem. European journal of Operational research. 1990;47(1):65–74.

[59] Marque-Pucheu S. Gaussian process regression of two nested computer codes. Université Paris-Diderot-Paris VII; 2018.

[60] Marque-Pucheu S, Perrin G, Garnier J. Efficient sequential experimental design for surrogate modeling of nested codes. ESAIM: Probability and Statistics. 2019;23:245–270.

[61] Dixit B. Elasticsearch Essentials. Packt Publishing Ltd; 2016.

[62] Hendrickx W, Gorissen D, Dhaene T. Grid enabled sequential design and adaptive metamodeling. In: Proceedings of the 2006 Winter Simulation Conference. IEEE; 2006. p. 872–881.

[63] Hendrickx W, Dhaene T. Sequential design and rational metamodelling. In: Proceedings of the 2005 Winter Simulation Conference. IEEE; 2005. p. 290 – 298.

[64] Davis L. Handbook of genetic algorithms. CUMINCAD; 1991.

[65] Dorigo M, Birattari M. Ant colony optimization. Springer; 2010.

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

[67] Bigot J, Gouet R, Klein T, López A, et al. Geodesic PCA in the Wasserstein space by convex PCA. In: Annales de l'Institut Henri Poincaré, Probabilités et Statistiques. vol. 53. Institut Henri Poincaré; 2017. p. 1–26.

[68] Carrere L, Lyard F, Cancet M, Guillot A, Picot N, et al. FES 2014, a new tidal model—Validation results and perspectives for improvements. In: Proceedings of the ESA living planet symposium; 2016. p. 9–13.

[69] 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`.

[70] 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.

[71] 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.

[72] 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.

[73] 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. 2012;25:2020–2039.

[74] 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;3:40–52.

[75] Bertsekas DP. Nonlinear programming. Journal of the Operational Research Society. 1997;48(3):334–334.

Appendix A. Dataset constitution

Input	Period	Initial time step	Source			
			Name	**Provider**	**Reference**	**Website**
Tide	1900-2016	No constrained. Set at 10min.	FES2014	LEGOS	[68]	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	[69]	https://reanalyses.org/atmosphere/overview-current-atmospheric-reanalyses\#TWENTv2c
	1979-2005	1h	CFSR (sea surface pressure)	NOAA	[70]	https://climatedataguide.ucar.edu/climate-data/climate-forecast-system-reanalysis-cfsr
	2006-2016	15min	MARC	Ifremer and LOPS	[71]	http://marc.ifremer.fr/
Hs, Tp	1900-1957	6h	Sonel (waves)	Liens	[72]	http://www.sonel.org/-Waves-.html?lang=en
	1958-09/2002	1h	BoBWA	BRGM	[73]	http://bobwa.brgm.fr/
	10/2002-2007	1h	Homere	Ifremer and LOPS	[74]	http://marc.ifremer.fr/en/produits/rejeu_d_etats_de_mer_homere
	2008-2016	1h	Iowaga/Norgasug	Ifremer and LOPS	[74]	https://wwz.ifremer.fr/iowaga/Products

Table A.6: Sources of data for the coastal flooding application case.

Appendix B. Setting the projection coefficients

Here we discuss the calibration of the projection coefficients. For the sake of presentation, let us consider a single functional input variable provided in a time series format over the set $\boldsymbol{t} = \{t_1, \ldots, t_L\}$, with $L \in \mathbb{N}$. Let $\boldsymbol{F}$ be a matrix of dimension $L \times n$ containing $n \in \mathbb{N}$ observations of that input variable. By adapting the expression for projections provided in (10) to discretized functions, and generalizing for the simultaneous projection of multiple curves, we obtain

$$\boldsymbol{F} \approx \Pi(\boldsymbol{F}) = \boldsymbol{B}\boldsymbol{\alpha},$$

where $\boldsymbol{B}$ is a $L \times p$ matrix containing p basis functions discretized into the L points in $\boldsymbol{t}$ and $\boldsymbol{\alpha}$ is a $p \times n$ matrix containing the p projection coefficients required to represent the n input curves. Assuming that the matrix $\boldsymbol{B}$ is produced by means of a standard method such as PLS, B-splines or PCA, the problem reduces to setting the values of the matrix $\boldsymbol{\alpha}$. This task is often completed using standard least squares optimization formulations. Closed form expressions for four variations of the problem are provided below. In the formulations we use $\boldsymbol{A}_{i,\bullet}$ to denote the i-th row of a matrix $\boldsymbol{A}$ and similarly $\boldsymbol{A}_{\bullet,j}$ to denote its j-th column. In addition, the orientation of the elements holds so that $\boldsymbol{A}_{i,\bullet}$ is a row vector whilst $\boldsymbol{A}_{\bullet,j}$ is a column vector

- Weighted Least Squares (WLS)

  Sometimes, certain points in the domain of the inputs are more important than others or the information about the input is more reliable there. This can be taken into account in the selection of the projection coefficients by introducing a diagonal weight matrix $\boldsymbol{W}$ of dimension $L \times L$ indicating the importance of each point in $\boldsymbol{t}$. Then, the projection coefficients can be found by minimizing the weighted sum of squared residuals. For $j = 1, \ldots, n$, the optimization problem can be written:

  $$\min_{\boldsymbol{\alpha}_{\bullet,j} \in \mathbb{R}^p} (\boldsymbol{F}_{\bullet,j} - \boldsymbol{B}\boldsymbol{\alpha}_{\bullet,j})' \boldsymbol{W} (\boldsymbol{F}_{\bullet,j} - \boldsymbol{B}\boldsymbol{\alpha}_{\bullet,j}). \tag{B.1}$$

  The integrated solution by derivatives for the n problems yields:

  $$\hat{\boldsymbol{\alpha}} = (\boldsymbol{B}'\boldsymbol{W}\boldsymbol{B})^{-1}\boldsymbol{B}'\boldsymbol{W}\boldsymbol{F}. \tag{B.2}$$

- Ordinary Least Squares (OLS)

  If all points in $\boldsymbol{t}$ are equaly important and the information at all points is equally reliable, the matrix $\boldsymbol{W}$ can be replaced by the identity matrix of dimension $L \times L$ in (B.1) and (B.2) or simply removed from the equations.

- Weighted-Constrained Least Squares (WCLS)

  If in addition to a set of important points in the domain of the inputs, there is some point $t_{i*} \in \boldsymbol{t}$ of outstanding relevance, a weighted-constrained formulation could be used. It allows to enforce the projection to interpolate exactly the true function at t_{i*}, while keeping relatively good precision on the remaining critical points. In this case, the coefficients of the projection can be found by solving (B.1), subject to the constraint

$$\boldsymbol{B}_{i*,\bullet}\boldsymbol{\alpha}_{\bullet,j} - \boldsymbol{F}_{i*,j} = 0. \tag{B.3}$$

To solve this problem we use the well known method of Lagrange multipliers [75] which allows to include equality constraints as part of the objective function in order to solve the problem by derivatives. For $j = 1, \ldots, n$, the Lagrange function to minimize can be writen as

$$\begin{aligned}\mathcal{L}(\boldsymbol{\alpha}_{\bullet,j}, \lambda_j) = (\boldsymbol{F}_{\bullet,j} - \boldsymbol{B}\boldsymbol{\alpha}_{\bullet,j})^{'}\boldsymbol{W}(\boldsymbol{F}_{\bullet,j} - \boldsymbol{B}\boldsymbol{\alpha}_{\bullet,j}) \\ +\lambda_j(\boldsymbol{B}_{i*,\bullet}\boldsymbol{\alpha}_{\bullet,j} - \boldsymbol{F}_{i*,j}),\end{aligned} \tag{B.4}$$

with $\lambda_j \in \mathbb{R}$ denoting the Lagrange multiplier for the optimization problem j. If we collect the values of the n Lagrange multipliers into a row vector $\boldsymbol{\lambda}$, the integrated solution by derivatives for the n optimization problems yields

$$\hat{\boldsymbol{\alpha}} = (\boldsymbol{B}'\boldsymbol{W}\boldsymbol{B})^{-1}\left(\boldsymbol{B}'\boldsymbol{W}\boldsymbol{F} - \frac{1}{2}\boldsymbol{B}'_{i*,\bullet}\hat{\boldsymbol{\lambda}}\right), \tag{B.5}$$

with

$$\hat{\boldsymbol{\lambda}} = \frac{2\left[\boldsymbol{B}_{i*,\bullet}(\boldsymbol{B}'\boldsymbol{W}\boldsymbol{B})^{-1}\boldsymbol{B}'\boldsymbol{W}\boldsymbol{F} - \boldsymbol{F}_{i*,\bullet}\right]}{\boldsymbol{B}_{i*,\bullet}(\boldsymbol{B}'\boldsymbol{W}\boldsymbol{B})^{-1}\boldsymbol{B}'_{i*,\bullet}}. \tag{B.6}$$

- Constrained Least Squares (CLS)

  The WCLS formulation can be easily modified for cases where only the point t_{i*} is of particular relevance. It suffices to replace the matrix $\boldsymbol{W}$ in (B.5) and (B.6) by the identity matrix of dimension $L \times L$.

We remark that the closed form solutions provided in (B.2), (B.5) and (B.6) 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).

Appendix C. Experimental conditions and results for analytic case

Conf.	Functional input		Projection method	Covariance function	Projection dimension	Results		
	f1	f2				$\bar{Q}^2$	CPU time (sec)	
							Train	Pred
1	0	0	-	-	-	0.0533	14.2	2.0
2	1	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	0.2914	23.2	2.0
3	0	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	0.3425	15.6	2.0
4	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	0.6367	17.6	2.0
5	1	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	0.1729	18.6	2.0
6	0	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	0.7814	19.0	2.0
7	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	0.9947	53.5	2.1
8	1	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	0.0717	18.7	2.6
9	0	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	0.3603	18.1	2.6
10	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	0.4283	21.8	2.6
11	1	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	0.4544	25.9	2.6
12	0	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	0.5819	23.3	2.6
13	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	0.9899	86.1	2.7
14	1	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	0.7262	17.3	2.2
15	0	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	0.7959	16.9	2.2
16	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	0.9994	32.4	2.3
17	1	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	0.7925	37.7	2.3
18	0	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	0.8442	38.8	2.3
19	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	0.9968	53.9	2.4
20	1	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	0.5017	20.9	2.6
21	0	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	0.6813	20.9	2.6
22	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	0.9989	49.5	2.9
23	1	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	0.5788	33.4	2.6
24	0	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	0.7825	36.2	2.6
25	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	0.9953	89.5	2.9
26	1	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	0.7901	56.6	2.4
27	0	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	0.8452	59.5	2.4
28	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	0.9991	27.6	2.4
29	1	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	0.8183	95.7	2.3
30	0	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	0.8687	91.7	2.1
31	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	0.9990	35.3	2.2
32	1	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	0.7626	33.0	2.9
33	0	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	0.8187	34.6	2.8
34	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	0.9995	45.9	3.1
35	1	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	0.7904	43.5	2.8
36	0	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	0.8412	44.2	2.8
37	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	0.9997	42.4	3.1

Table C.7: 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. For the functional input, 1 denotes active and 0 denotes inactive.

Conf.	Functional input		Projection method	Covariance function	Projection dimension	Results		
	f1	f2				$\bar{Q}^2$	CPU time (sec)	
							Train	Pred
38	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	4	0.9981	43.2	2.2
39	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	4	0.9982	54.9	2.3
40	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	4	0.9997	36.9	3.3
41	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	4	0.9997	49.0	3.3
42	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	5	0.9959	52.8	2.5
43	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	5	0.9973	211.9	2.8
44	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	5	0.9997	34.2	3.5
45	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	5	0.9997	48.3	3.5
46	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	6	0.9959	62.1	2.6
47	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	6	0.9960	365.3	2.8
48	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	6	0.9997	34.0	3.7
49	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	6	0.9997	43.0	3.7

Table C.8: Analytic case: experimental conditions and results from cleaning and descent stages. The training and prediction times are computed as described in Table C.7. 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	Projection dimension	Results		
								CPU time (sec)	
	Td	Sg	Tp				$\bar{Q}^2$	Train	Pred
1	0	0	0	-	-	-	0.7209	27.7	2.0
2	1	0	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	0.7214	32.0	2.0
3	0	1	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	0.7226	31.2	2.1
4	1	0	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	0.7146	36.3	2.1
5	0	0	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	0.7211	29.0	2.0
6	1	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	0.7201	34.8	2.1
7	0	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	0.7213	33.9	2.1
8	1	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	0.7123	42.6	2.1
9	1	0	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	0.7198	29.1	2.1
10	0	1	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	0.7004	29.1	2.1
11	1	1	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	0.6948	30.6	2.1
12	0	0	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	0.7160	36.7	2.1
13	1	0	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	0.7074z	40.2	2.1
14	0	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	0.6958z	39.3	2.1
15	1	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	1	0.6864z	45.3	2.1
16	1	0	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	0.7244	35.8	2.7
17	0	1	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	0.7232	34.3	2.6
18	1	1	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	0.7247	38.5	2.8
19	0	0	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	0.7217	48.8	2.7
20	1	0	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	0.7238	61.9	2.8
21	0	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	0.7230	57.0	2.7
22	1	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	0.7247	62.5	2.8
23	1	0	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	0.7234	33.5	2.6
24	0	1	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	0.7210	33.4	2.6
25	1	1	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	0.7218	37.7	2.7
26	0	0	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	0.7199	52.0	2.6
27	1	0	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	0.7216	62.4	2.7
28	0	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	0.7192	59.1	2.7
29	1	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	1	0.7204	64.9	2.8
30	1	0	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	0.6967	32.0	2.3
31	0	1	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	0.6915	33.2	2.3
32	1	0	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	0.6592	42.3	2.4
33	0	0	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	0.7137	37.8	2.3
34	1	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	0.6866	48.1	2.4
35	0	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	0.6869	48.3	2.4
36	1	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	0.6507	67.0	2.6
37	1	0	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	0.6973	31.4	2.3
38	0	1	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	0.7048	34.0	2.3
39	1	1	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	0.6653	44.4	2.4
40	0	0	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	0.7145	42.4	2.3
41	1	0	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	0.6875	60.5	2.5
42	0	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	0.6932	61.3	2.5
43	1	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	2	0.6532	95.1	2.6
44	1	0	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	0.7215	33.9	2.7
45	0	1	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	0.7211	35.7	2.7
46	1	1	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	0.7195	38.4	2.8
47	0	0	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	0.7199	55.0	2.8
48	1	0	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	0.7203	60.8	2.9
49	0	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	0.7190	61.2	2.9
50	1	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	0.7187	65.8	3.0
51	1	0	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	0.7226	33.2	2.7
52	0	1	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	0.7211	37.2	2.7
53	1	1	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	0.7197	35.5	2.8
54	0	0	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	0.7199	52.0	2.7
55	1	0	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	0.7212	58.7	2.9
56	0	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	0.7196	59.5	2.9
57	1	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	2	0.7197	63.0	3.0
58	1	0	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	0.7081	36.3	2.3
59	0	1	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	0.7158	35.9	3.4
60	1	0	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	0.6790	50.7	2.5
61	0	0	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	0.7121	53.8	2.4
62	1	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	0.6969	84.2	2.6
63	0	1	1	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	0.6994	81.1	2.6
64	1	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	0.6699	130.6	2.7
65	1	0	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	0.6737	38.0	2.4
66	0	1	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	0.7014	37.0	2.4
67	1	1	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	0.6428	55.8	2.6
68	0	0	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	0.7117	56.5	2.4
69	1	0	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	0.6665	122.9	2.6
70	0	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	0.6956	94.4	2.6
71	1	1	1	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	3	0.6371	182.7	2.7
72	1	0	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	0.7225	33.9	2.7
73	0	1	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	0.7202	36.1	2.8
74	1	1	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	0.7211	37.5	2.9
75	0	0	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	0.7205	53.3	2.8

Continued on next page

Table D.9 – *Continued from previous page*

76	1	0	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	0.7216	67.7	3.0
77	0	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	0.7194	60.7	3.0
78	1	1	1	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	0.7206	65.8	3.2
79	1	0	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	0.7231	31.5	2.7
80	0	1	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	0.7219	36.9	2.7
81	1	1	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	0.7225	35.7	2.9
82	0	0	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	0.7203	52.4	2.8
83	1	0	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	0.7218	58.0	3.0
84	0	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	0.7212	60.0	3.0
85	1	1	1	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	3	0.7212	63.3	3.2

Table D.9: Coastal flooding case: experimental conditions and results from screenig stage. The training and prediction times are computed as described in Table C.7. For the functional input, 1 denotes active and 0 denotes inactive.

Run	Projection dimension			Projection method	Covariance function	Results		
	Td	Sg	Tp			$\bar{Q}^2$	CPU time (sec)	
							Train	Pred
86	4	0	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	0.6734	40.6	2.2
87	0	15	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	0.6233	285.5	2.6
88	4	15	0	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	0.5653	400.9	2.8
89	0	0	10	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	0.7066	261.4	2.4
90	4	0	10	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	0.6554	420.4	2.6
91	0	15	10	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	0.6096	1055.3	3.0
92	4	15	10	B-splines	$\|\cdot\|_{S,\boldsymbol{\theta}}$	0.5562	1280.8	3.2
93	4	0	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	0.6651	44.9	2.1
94	0	10	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	0.6569	71.4	2.4
95	4	10	0	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	0.5871	120.5	2.6
96	0	0	6	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	0.7138	101.1	2.3
97	4	0	6	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	0.6546	265.6	2.4
98	0	10	6	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	0.6472	363.4	2.6
99	4	10	6	PCA	$\|\cdot\|_{S,\boldsymbol{\theta}}$	0.5830	674.1	3.1
100	4	0	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	0.7228	35.8	2.8
101	0	15	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	0.7230	50.0	3.6
102	4	15	0	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	0.7221	54.1	3.7
103	0	0	10	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	0.7218	61.4	3.3
104	4	0	10	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	0.7223	65.7	3.5
105	0	15	10	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	0.7222	85.3	4.1
106	4	15	10	B-splines	$\|\cdot\|_{D,\boldsymbol{\theta}}$	0.7220	98.2	4.3
107	4	0	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	0.7211	35.6	2.8
108	0	10	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	0.7235	32.8	3.2
109	4	10	0	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	0.7230	34.8	3.4
110	0	0	6	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	0.7207	53.5	2.6
111	4	0	6	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	0.7205	60.3	2.9
112	0	10	6	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	0.7234	59.1	3.2
113	4	10	6	PCA	$\|\cdot\|_{D,\boldsymbol{\theta}}$	0.7214	67.5	3.4

Table D.10: Coastal flooding case: evaluation of projection dimension selected based on projection error. The training and prediction times are computed as described in Table C.7.