



Functional-input metamodeling: an application to coastal flood early warning

José Daniel Betancourt

► To cite this version:

José Daniel Betancourt. Functional-input metamodeling: an application to coastal flood early warning. Modeling and Simulation. Université Paul Sabatier - Toulouse III, 2020. English. NNT: 2020TOU30097 . tel-03117879

HAL Id: tel-03117879

<https://theses.hal.science/tel-03117879>

Submitted on 21 Jan 2021

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

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



THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le 8 Juin 2020 par :

JOSÉ BETANCOURT

**Functional-input metamodeling: an application to coastal
flood early warning**

JURY

PERRIN GUILLAUME	Ciat à l'Énergie Atomique	Rapporteur
SONJA KUHNT	FH Dortmund University	Rapporteuse
BERNARD BERCU	Université Bordeaux 1	Examineur
FABRICE GAMBOA	Université Toulouse 3	Examineur
CÉLINE HELBERT	École Centrale de Lyon	Examinatrice
JÉRÉMY ROHMER	BRGM Orléans	Examineur
THIERRY KLEIN	Université Toulouse 3 - ENAC	Directeur de thèse
FRANÇOIS BACHOC	Université Toulouse 3	Co-directeur de thèse

École doctorale et spécialité :

MITT : Domaine Mathématiques : Mathématiques appliquées

Unité de Recherche :

Institut de Mathématiques de Toulouse (UMR 5219)

Directeur(s) de Thèse :

Thierry Klein et François Bachoc

Rapporteurs :

Perrin Guillaume et Sonja Kuhnt

Dedicated to

God

Who gives me every single breath and heartbeat.

My beloved family

*My grandparents Vicente and Gloria, my parents Julia and Luis, my uncles Juan and Jorge,
my auntie Zaida, her amazing child Sara and Josue, and my precious wife Juliette.
I profoundly love each of you.*

Acknowledgments



Photo taken at Gavre, France, during a dinner of the RISCOPE project.

The more I learn, the more I observe, the more I see the majesty of God in his amazing creation. Which model made by us, humans, could ever reach such a stunning degree of complexity? Which mankind or artificial mind is so powerful to form a universe that stands as the aspiration of any model the science might conceive? For God is all the glory of my triumph.

Huge thanks to Professor Fabrice Gamboa and my advisors, Professors Thierry Klein and François Bachoc. You three have behaved as amazing leaders. Thanks for dedicating so much time to my thesis and even paying attention when I needed help with some personal issue. Also special thanks for being warm and sitting down at the table with me several times. I really enjoyed those moments and will be happy to do it again in the future... just don't choose the restaurant "Poivre rouge" next to the Comfort hotel at Orléans ;).

Working at the RISCOPE project has been a wonderful experience for me. I have learnt a lot, enjoyed the trips and dinners and met new places. I'm grateful for the fundings I received from the project to conduct my PhD studies. Beyond that, I am enormously grateful to Déborah Idier and Jérémy Rohmer, from BRGM Orléans, who were so attentive that I followed the good path in order to complete my work in the project with success. Thank you both for reading those long emails, for paying attention to my ideas, for giving valuable opinions that ended up in great results within the thesis.

Special recognitions for my family, who have always encouraged me and gave me advice throughout this project. While staying here in France doing the PhD I have had the time to think on many situations where you were patient, tolerant and loving while I was making mistakes by your side. Thank you for taking care of me and supporting me so much.

My wife, Juliette... you have been just essential. Thanks for helping me in many different ways; giving me support when I felt down, studying with me until we were able to understand, staying awake till the daybreak working hard to complete this project... What an amazing privilege it is to work with you, my brilliant partner. Your technical and personal support was crucial on every step of this thesis.

Thanks to everybody who has opened their arms to Juliette and me during this phase of our lives. Daniel and Barbara, who received us at their home when we first arrived; Bryan, the smart and friendly French professor that we happily met; and last but definitely not least, my two officemates Camille and Trang, who made pleasant to go to the IMT and stay most of the time nowhere else but at office with you.

Hope the best for the special people mentioned above and every other person that somehow helped to make this possible.

Résumé

Les inondations en général affectent plus de personnes que tout autre catastrophe. Au cours de la dernière décennie du 20ème siècle, plus de 1.5 milliard de personnes ont été affectées. Afin d'atténuer l'impact de ce type de catastrophe, un effort scientifique significatif a été consacré à la constitution de codes de simulation numériques pour la gestion des risques. Les codes disponibles permettent désormais de modéliser correctement les événements d'inondation côtière à une résolution assez élevée. Malheureusement, leur utilisation est fortement limitée pour l'alerte précoce, avec une simulation de quelques heures de dynamique maritime prenant plusieurs heures à plusieurs jours de temps de calcul. Cette thèse fait partie du projet ANR RISCOPE, qui vise à remédier cette limitation en construisant des métamodèles pour substituer les codes hydrodynamiques coûteux en temps de calcul.

En tant qu'exigence particulière de cette application, le métamodèle doit être capable de traiter des entrées fonctionnelles correspondant à des conditions maritimes variant dans le temps. À cette fin, nous nous sommes concentrés sur les métamodèles de processus Gaussiens, développés à l'origine pour des entrées scalaires, mais maintenant disponibles aussi pour des entrées fonctionnelles. La nature des entrées a donné lieu à un certain nombre de questions sur la bonne façon de les représenter dans le métamodèle: (i) quelles entrées fonctionnelles méritent d'être conservées en tant que prédicteurs, (ii) quelle méthode de réduction de dimension (e.g., B-splines, PCA, PLS) est idéale, (iii) quelle est une dimension de projection appropriée, et (iv) quelle est une distance adéquate pour mesurer les similitudes entre les points d'entrée fonctionnels dans la fonction de covariance. Certaines de ces caractéristiques - appelées ici paramètres structurels - du modèle et d'autres telles que la famille de covariance (e.g., Gaussien, Matérn 5/2) sont souvent arbitrairement choisies a priori. Comme nous l'avons montré à travers des expériences, ces décisions peuvent avoir un fort impact sur la capacité de prédiction du métamodèle. Ainsi, sans perdre de vue notre but de contribuer à l'amélioration de l'alerte précoce des inondations côtières, nous avons entrepris la construction d'une méthodologie efficace pour définir les paramètres structurels du modèle.

Comme première solution, nous avons proposé une approche d'exploration basée sur la Méthodologie de Surface de Réponse. Elle a été utilisée efficacement pour configurer le métamodèle requis pour une fonction de test analytique, ainsi que pour une version simplifiée du code étudié dans RISCOPE. Bien que relativement simple, la méthodologie proposée a pu trouver des configurations de métamodèles de capacité de prédiction élevée avec des économies allant jusqu'à 76.7% et 38.7% du temps de calcul utilisé par une approche d'exploration exhaustive dans les deux cas étudiés. La solution trouvée par notre méthodologie était optimale dans la plupart des cas. Nous avons développé plus tard un deuxième prototype basé sur l'Optimisation par Colonies de Fourmis. Cette nouvelle approche est supérieure en termes de temps de solution et de flexibilité sur les configurations du modèle qu'elle permet d'explorer. Cette méthode explore intelligemment l'espace de solution et converge progressivement vers la configuration optimale. La collection d'outils statistiques utilisés dans cette thèse a motivé le développement d'un package R appelé funGp. Celui-ci est maintenant disponible dans GitHub et sera soumis prochainement au CRAN.

Dans un travail indépendant, nous avons étudié l'estimation des paramètres de covariance d'un processus Gaussien transformé par Maximum de Vraisemblance (MV) et Validation Croisée. Nous avons montré la consistance et la normalité asymptotique des deux estimateurs. Dans le cas du MV, ces résultats peuvent être interprétés comme une preuve de robustesse du MV Gaussien dans le cas de processus non Gaussiens.

Abstract

Currently, floods in general affect more people than any other hazard. In just the last decade of the 20th century, more than 1.5 billion were affected. In the seek to mitigate the impact of this type of hazard, strong scientific effort has been devoted to the constitution of computer codes that could be used as risk management tools. Available computer models now allow properly modelling coastal flooding events at a fairly high resolution. Unfortunately, their use is strongly prohibitive for early warning, with a simulation of few hours of maritime dynamics taking several hours to days of processing time, even on multi-processor clusters. This thesis is part of the ANR RISCOPE project, which aims at addressing this limitation by means of surrogate modeling of the hydrodynamic computer codes.

As a particular requirement of this application, the metamodel should be able to deal with functional inputs corresponding to time varying maritime conditions. To this end, we focused on Gaussian process metamodels, originally developed for scalar inputs, but now available also for functional inputs. The nature of the inputs gave rise to a number of questions about the proper way to represent them in the metamodel: (i) which functional inputs are worth keeping as predictors, (ii) which dimension reduction method (e.g., B-splines, PCA, PLS) is ideal, (iii) which is a suitable projection dimension, and given our choice to work with Gaussian process metamodels, also the question of (iv) which is a convenient distance to measure similarities between functional input points within the kernel function. Some of these characteristics - hereon called structural parameters - of the model and some others such as the family of kernel (e.g., Gaussian, Matérn 5/2) are often arbitrarily chosen a priori. Sometimes, those are selected based on other studies. As one may intuit and has been shown by us through experiments, those decisions could have a strong impact on the prediction capability of the resulting model. Thus, without losing sight of our final goal of contributing to the improvement of coastal flooding early warning, we undertook the construction of an efficient methodology to set up the structural parameters of the model.

As a first solution, we proposed an exploration approach based on the Response Surface Methodology. It was effectively used to tune the metamodel for an analytic toy function, as well as for a simplified version of the code studied in RISCOPE. While relatively simple, the proposed methodology was able to find metamodel configurations of high prediction capability with savings of up to 76.7% and 38.7% of the time spent by an exhaustive search approach in the analytic case and coastal flooding case, respectively. The solution found by our methodology was optimal in most cases. We developed later a second prototype based on Ant Colony Optimization (ACO). This new approach is more powerful in terms of solution time and flexibility in the features of the model allowed to be explored. The ACO based method smartly samples the solution space and progressively converges towards the optimal configuration. The collection of statistical tools used for metamodeling in this thesis motivated the development of the funGp R package, which is now available in GitHub and about to be submitted to CRAN.

In an independent work, we studied the estimation of the covariance parameters of a Transformed Gaussian Process by Maximum Likelihood (ML) and Cross Validation. We showed that both estimators are consistent and asymptotically normal. In the case of ML, these results can be interpreted as a proof of robustness of Gaussian ML in the case of non-Gaussian processes.

Contents

List of Tables	x
List of Figures	xi
1 Introduction	1
1.1 Research context	1
1.2 Contributions of the thesis	2
1.3 Outline of the manuscript	3
2 Gaussian process metamodeling for coastal flood hazard assessment	7
2.1 Introduction	9
2.2 Motivating case: coastal flooding prediction at Gâvres, France	12
2.2.1 Hydrodynamic code	12
2.2.2 Dataset	13
2.3 Theoretical background	14
2.3.1 Scalar and functional inputs of computer codes	14
2.3.2 Gaussian process metamodeling of scalar-input codes	15
2.3.3 Gaussian process metamodeling of functional-input codes	16
2.3.3.1 Three distances for functional inputs	17
2.3.4 Gaussian process metamodeling with scalar and functional inputs	19
2.4 Exploration strategy	19
2.4.1 Scope of the methodology	21
2.4.2 Analytic case	22
2.4.2.1 Dataset	23
2.4.2.2 Screening	24
2.4.2.3 Cleaning and descent	27
2.5 Coastal flooding case study	30
2.5.1 Screening	30
2.5.2 Dimension selection based on projection error	36
2.5.2.1 Selecting the dimension for each input	36
2.5.2.2 Efficiency of configurations based on projection error	38
2.6 Robustness to changes in the amount of training/validation data	40
2.6.1 Experiment setting	40
2.6.2 Performance statistics	40
2.6.3 Analysis	41
2.7 Conclusions	44
Appendix 2.A Dataset constitution	46
Appendix 2.B Setting the projection coefficients	47

Appendix 2.C	Conditions and results for analytic case	49
Appendix 2.D	Conditions and results for coastal flooding case	50
3	Ant Colony algorithm for structural parameter calibration	53
3.1	Introduction	57
3.2	The ant colony system	57
3.2.1	Biological inspiration	57
3.2.2	The optimization algorithm	58
3.2.3	Adaption to model selection	59
3.3	Analytic test cases	64
3.3.1	Black-box functions	64
3.3.2	Data generation and heuristic setup	65
3.3.3	Results	66
3.4	Conclusions	68
3.5	Aknowledgments	68
4	User manual for the R package funGp	69
	In-code notation	73
4.1	Base functionalities	74
4.1.1	Create a funGp model	74
4.1.2	Predict using a funGp model	76
4.1.3	Simulate from a funGp model	79
4.1.4	Update a funGp model	81
4.2	Model customizations	83
4.2.1	Kernel family	84
4.2.2	Projection basis	84
4.2.3	Projection dimension	85
4.2.4	Distance for functions	85
4.3	Heuristic model selection	87
4.3.1	Concept	87
4.3.2	Using the model factory in funGp	88
4.4	Parallelization in funGp	97
4.4.1	Parallelized hyperparameters optimization	98
4.4.2	Parallelized model selection	99
	Closing discussion	99
	Acknowledgements	100
5	Structural parameter optimization in the coastal flooding case	101
5.1	Introduction	103
5.2	The new hydrodynamic code	103
5.3	Structural parameter optimization	104
5.4	Conclusions	106
6	ML and CV Asymptotics for trans-Gaussian processes	107
6.1	Introduction	109
6.2	General properties of transformed Gaussian processes	111
6.3	Two main technical results	113
6.3.1	Transformed Gaussian process framework	113
6.3.2	Bounds on the elements of inverse covariance matrices	113

6.3.3	CLT for quadratic forms of transformed Gaussian processes	114
6.4	Estimation of a single variance parameter	115
6.5	General covariance	116
6.5.1	Framework	116
6.5.2	Maximum Likelihood	117
6.5.3	Cross Validation	120
6.5.4	Joint asymptotic normality	122
6.6	Discussions of extensions	123
6.6.1	Nugget effect	123
6.6.2	Unknown mean	124
6.7	Illustration	124
6.8	Conclusion	128
Appendix 6.A	Proofs	129
6.A.1	Technical results	129
6.A.2	Proofs of the main results	135
	Proof of Lemma 1	135
	Proof of Lemma 2	136
	Proof of Theorem 1	136
	Proof of Theorem 2	138
	Proof of Theorem 3	140
	Proof of Theorem 4	141
	Proof of Theorem 5	142
	Proof of Theorem 6	142
	Proof of Lemma 4	143
	Proof of Theorem 7	145
	Proof of Proposition 1	146
	Proof of Corollary 3	146
7	Global conclusions and perspectives	147
7.1	Synthesis	147
7.2	Scientific production	147
7.3	Research perspectives	148

List of Tables

2.1	Analytic case: parameters and levels for the screening stage.	25
2.2	Coastal flooding case: parameters and levels for the screening stage.	32
2.3	Selected dimension based on projection error.	38
2.4	Analytic case: robustness to changes in the amount of data	42
2.5	Coastal flooding case: robustness to changes in the amount of data	43
2.6	Sources of data for the coastal flooding application case.	46
2.7	Analytic case: results of screening	49
2.8	Analytic case: results of cleaning and descent	50
2.9	Coastal flooding case: results of screening	52
2.10	Coastal flooding case: results of cleaning and descent	52
5.1	Ants vs. 3 benchmark configurations	106

List of Figures

2.1	Cross-shore transect used by the simplified hydrodynamic code	13
2.2	Functional inputs used by the simplified hydrodynamic code	14
2.3	Flowchart of our first methodology for structural parameter calibration	20
2.4	Analytic case: functional input f_1	23
2.5	Analytic case: results of screening	27
2.6	Analytic case: results of cleaning and descent	28
2.7	Analytic case: fitting of the selected configuration	29
2.8	Analytic case: calibration plot of the selected configuration	29
2.9	Coastal flooding case: PCA on the shifted inputs	31
2.10	Coastal flooding case: weights for optimization of projection coefficients	32
2.11	Coastal flooding case: results of screening	33
2.12	Coastal flooding case: influence of outliers in model performance	34
2.13	Coastal flooding case: calibration plot of the selected configuration	35
2.14	Coastal flooding case: fitting of the selected configuration	35
2.15	Coastal flooding case: dimension based on projection error	37
2.16	Coastal flooding case: large dimension based on projection error	38
2.17	Coastal flooding case: quality of metamodels based on projection error	39
3.1	Ants learning through stigmergy	58
3.2	Decision network used by the ant colony algorithm	60
3.3	Example of regularized pheromones	63
3.4	Example of normalized pheromones	64
3.5	Normalized pheromones for analytic test cases	65
3.6	Calibration plot for selected models.	66
3.7	Relative quality of selected models.	67
3.8	Evolution of the heuristic.	67
4.1	Decision network used by the ant colony algorithm	88
4.2	Calibration plot optimized vs arbitrary structural configuration	89
5.1	Main roads and surveillance points at Gâvres	104
5.2	Ants vs. 3 benchmak configurations	105
6.1	Histograms of variance maximum likelihood estimates	125
6.2	Scatter plots of variance and range maximum likelihood estimates	126
6.3	Mean squared error, squared bias and variance as a function of n	127
6.4	Wasserstein distance as a function of n	128

Chapter 1

Introduction

1.1 Research context

In the early 20s, Ronald Fisher developed the base concepts of modern design of experiments [1]. At the time, the only way to get data about a system of interest was to conduct a physical experiment. The outstanding evolution of computer technology has took those concepts to a whole new level, where in-field measurements can now be replaced by reliable and detailed computer simulations. This possibility has enabled the study of a variety of complex systems that otherwise would be too expensive or too difficult to observe. We refer, for instance, to the optimal design of nuclear reactors [2] and space vehicles [3], or the modeling and prediction of natural hazards [4]. Despite their advantages in terms of costs and control over the experiment, 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) [5]. This obstacle is typically circumvented by creating quick-to-evaluate mathematical emulators of those numerical codes, based on a limited collection of runs (see e.g., [6]); such emulators are often called *surrogate models* or *metamodels*. This thesis makes part of the ANR RISCOPE project (<https://perso.math.univ-toulouse.fr/riscope/> ) devoted to the application of metamodeling techniques for the improvement of coastal flooding early warning.

One of the particularities of the codes studied here, is that they receive multiple time series as inputs. Those correspond to key maritime conditions such as the tide, atmospheric storm surge, and significant wave height. The proper modeling of functional inputs has been of main interest in the project since the beginning, and the manuscript makes strong emphasis on this subject. In this regard, we have studied the possibility of implementing dimension reduction techniques in order to simplify the input data structures and obtain lighter models. Our work is focused on Gaussian process metamodels, featured by the degree of flexibility they allow in the input-output relationship, their interpretability, and remarkable tractability. The handling of functional inputs and our choice to work with Gaussian process models give rise to a number of questions about the ideal model setup. Specifically, we aimed to decide:

- 1) the scalar and functional inputs to keep as predictors;
- 2) the dimension reduction method to use for each input;
- 3) the projection dimension for each input;
- 4) the kernel function of the model;
- 5) the distance used to measure similarity between functional input coordinates within the kernel function.

We call those features ***structural parameters*** of the model. Some of these characteristics of the model are often arbitrarily fixed. Through a set of computer experiments, we show that the structural parameters may have a strong impact on the prediction capability of the resulting model. We further conclude that the ideal model setup will vary from one application to the other, and thus it should be optimized each time a model is built, in order to get the best results of it. This thesis introduces an algorithm to conduct such an optimization task. We validate the algorithm in the RISCOPE case. All the obtained metamodels were of outstanding prediction quality.

1.2 Contributions of the thesis

This thesis focuses on the construction of Gaussian process metamodels for the substitution of complex hydrodynamic codes in the frame of coastal flood early warning. Our main contributions in this regard are summarized below.

1. A set of metamodels for direct application in coastal flood early warning

The RISCOPE project works on a ***real life case study***, focused on the coastal French municipality of Gâvres. This region has experienced more than ten coastal flooding events since 1864, two of the most dramatic ones during the 21st century. We have worked in direct contact with the mayor, representatives of the fire department, and other local authorities in order to make the results of the project as useful as possible for them. The GeoHyd group, partner in the project, is in charge of the development of a web-based ***decision support system for early warning***. This tool is expected to implement various of the metamodels obtained from this thesis.

2. An algorithm for model selection in the frame of Gaussian process models

As explained in the description of the research context, the types of inputs used by the hydrodynamic computer codes, along with our decision to work with Gaussian process models, result in a number of potential model structures, one of which will be the optimal choice. We consider a total of five model features to calibrate, those being: (i) the state of each input in the model (inactive or active), (ii) the dimension reduction method to use for each input (e.g., B-splines, PCA), (iii) the projection dimension for each input, (iv) the kernel function of the model (e.g., Gaussian, Matérn 5/2), and (v) the distance to measure similarities between functional input coordinates. We develop an ***Ant Colony based algorithm*** able to find, in a short time, a combination of structural parameters leading to a model of ***superior predictability***. We use this algorithm to tune the models for the RISCOPE application.

3. An R package implementing the metamodeling techniques used here

This thesis gives rise to the ***R package funGp***. This package tackles regression problems involving scalar and/or functional inputs and a scalar output through the fairly general Gaussian process model. Rather than imposing or requiring any particular parametric input-output relationship in advance, it learns this information directly from the data. The package offers ***built-in dimension reduction*** methods oriented to simplify the representation of the functional inputs in order to get lighter models. funGp also implements the ***Ant Colony algorithm*** developed here for the calibration of the structural parameters of the model. Both the package and its manual have been structured with the intention of them being usable for users within a wide range of knowledge in mathematics or statistics. Therefore, all the functions offer default

parameters, but great degree of customization is also offered for the more experienced users. funGp has been made *available in GitHub and CRAN*.

In an excursion outside the coastal flooding application, we also worked on theoretical aspects of Gaussian process metamodeling. In many applications the data presents some signs of non-Gaussianity, such as boundary constraints or skewness. This study is motivated by the question of what we can expect in those cases, if we still model the output as Gaussian. Our contributions in this front are explained below.

4. Results on the asymptotics of ML and CV for trans-Gaussian processes

We study the estimation of the covariance parameters of transformed Gaussian processes through Gaussian Maximum Likelihood (ML) and Cross Validation (CV). We show that both estimators are consistent and asymptotically normal. These results can be interpreted as a proof of robustness of Gaussian ML in the case of non-Gaussian processes. We conclude that by modeling a random process as Gaussian, we can estimate well its covariance parameters, even if the process is not Gaussian in reality. In the regression framework, this points out to the conclusion that we can remove the assumption of Gaussianity of the data and still manage to build a Gaussian process model well suited to the data.

1.3 Outline of the manuscript

The remaining of this manuscript is composed as follows:

Chapter 2: Gaussian process metamodeling of functional-input code for coastal flood hazard assessment

This chapter presents our first approach to address the metamodeling problem studied in the RISCOPE project. It is an exact copy of our article [7], published in *Reliability Engineering & System Safety*. The chapter covers good part of the foundations of this thesis. It includes a discussion on related literature, a description of RISCOPE dataset, a synthesis of modeling through Gaussian processes, along with a detailed explanation of our metamodeling approach. Hence, this chapter is fundamental for the effective comprehension of the others.

At the time of writing the article, the target hydrodynamic code was in calibration. Therefore, we were using a simplified version of it, which ignored some of the hydrometeorological forcing conditions. The simplified code was much quicker to evaluate but less detailed and precise. This code, however, allowed us to generate a large number of observations, which were of great importance for the stabilization and optimization of our metamodeling routines. On the other hand, we were able to test various ways to include the functional inputs in the model. Through a large number of computer experiments, we showed the potential impact of the structural parameters in the predictability of the model. This article gave rise to our first prototype of methodology for the calibration of the structural parameters of the model, which was inspired by the Response Surface Methodology [8].

While relatively simple, the proposed methodology proved its effectiveness through an analytical case study and our coastal flooding application. In both cases it led to metamodel configurations of outstanding performance, able to accurately predict the output of the code. We proved the efficiency of this methodology through comparison against an exhaustive search approach. In most cases our methodology was able to find the optimal configuration,

and made time savings of up to 76.7% and 38.7% with respect to the time spent by the exhaustive search in the analytic and the coastal flooding case, respectively. Despite such positive results, we recognized the limitation of this approach to problems with few number of structural parameters and levels of those. Having proved the pertinence of this research line, we developed a second methodology able to explore the space in a more efficient manner. Such a methodology is the topic of [Chapter 3](#).

Chapter 3: Ant Colony based model selection for functional-input Gaussian process regression

The previous chapter presents our first prototype of methodology for the calibration of structural parameters of the metamodel. We used that methodology to tune the metamodel for a simplified version of our hydrodynamic code, which received only part of the variables used by our target code. Although the proposed methodology proved to be effective for such application, we concluded the need for more sophisticated exploration techniques in order to scale to bigger problems involving (i) a larger number of inputs, (ii) more structural parameters and (iii) more levels per structural parameter. In [Chapter 3](#) we address this need by introducing an Ant Colony based smart exploration algorithm.

Ant colony optimization (ACO) encompasses a large variety of optimization metaheuristics derived from the seminal work of Dorigo et al. in the early 90s [9, 10]. Since then, ACO based heuristics have been proved to give remarkable results in a wide range of optimization problems, including DNA sequencing [11], scheduling [12], protein-ligand docking [13], assembly line balancing [14] and packet-switched routing [15]. ACO has been recognized as one of the most successful research lines in the area of swarm intelligence [16, 17], and always seats beside evolutionary algorithms, iterated local search, simulated annealing, and tabu search among the top metaheuristic techniques [18]. This chapter is an exact copy of our technical report [19], which recalls the foundations of Ant Colony Optimization and elaborates on top of them to make the algorithm suitable for our structural optimization problem. To the best of our knowledge, this is the first algorithm addressing the structural optimization problem for Gaussian process models.

We use the proposed algorithm to calibrate the metamodel for three analytic black-box functions. The models obtained were in all cases of outstanding prediction quality. In [Chapter 5](#) we use the algorithm to calibrate the metamodel for the full hydrodynamic code of the RISCOPE application.

Chapter 4: Gaussian process regression for scalar and functional inputs with funGp: The in-depth tour

The two previous chapters led us to the creation of the R package called **funGp** [20], which is oriented to the construction and smart selection of Gaussian process models with emphasis on the treatment of functional inputs. [Chapter 4](#) is an exact copy of its user manual [21].

What does funGp bring to the table?

- **Flexible modeling of functional-input regression problems**

A few R packages address regression with functional inputs (e.g., time series). The vast majority of those packages rely on models limited by strong assumptions on the relationship between inputs and outputs (e.g., Linear, Generalized Linear or Generalized Additive Models). **funGp** tackles regression problems involving scalar and/or functional inputs

through the fairly general Gaussian process model, which removes any need to set a particular input-output parametric relationship in advance, and rather learns this information directly from the data.

- **Built-in dimension reduction**

funGp is self-contained in the sense that it does not depend on other packages to perform dimension reduction on the functional inputs. At this point, it offers projection onto B-splines or PCA bases. The package was designed to enable a straightforward extension to other bases in further versions.

- **Heuristic model selection**

The possibilities offered by a package often translate into alternative model structures. Decision support is rarely offered in order to select a suitable configuration for the problem at hand. We acknowledge the potential impact of such a decision in the performance of the model [7, 22] and also the practical difficulties that arise from offering possibilities without decision support. Therefore, **funGp** was equipped with a model selection functionality which implements the Ant Colony based algorithm presented in 3. The user has the possibility to decide to either build a model with an arbitrary structural configuration, or call the model factory and let the ants do the work.

- **All-level-user-friendly**

We aim **funGp** to be a helpful tool for users within a wide range of knowledge in mathematics or statistics. Thus, we have made an effort to make simple and intuitive the way the package works. Most of the arguments in the functions have been provided default values so that the user can start experimenting with them at its own pace. Once the user gets ready, it will be able to start playing with the nugget effect, basis type, kernel type, multi-start option, parallelization and even the parameters of the heuristic for structural configuration. However, to have a first **funGp** model built, the only thing to do is to provide the data.

funGp relies on a variety of concepts at a crossroad between mathematics, statistics and optimization. As such, it merits a sufficiently wide documentation helping the user to shorten its learning curve. It does not matter what a tool has to offer if people do not understand well how to use it. The user manual presented in Chapter 4 explains all the functionalities of the package through a set of short examples in the form of code snippets copy/pasteable directly to R. We are confident that, with the help of this manual, the user will be able to have its first **funGp** model working in a matter of just a few minutes.

Chapter 5 Structural parameter optimization in the coastal flooding RISCOPE case

Chapters 2, 3 and 4 address the introduction to the RISCOPE coastal flooding application, the development of an Ant Colony algorithm for the efficient optimization of the structural parameters of the metamodel, and the consolidation of the R package **funGp** [20], respectively. Chapter 5 is a follow up to the RISCOPE case. The description given in Chapter 2 for this application remains mostly valid, except for the three following aspects:

- 1) **Updated hydrodynamic code:** at the time of writing Chapter 2, the target hydrodynamic code was in calibration. Thus, we used a simplified version of it which was quicker but less precise and detailed. At this time the target hydrodynamic code is complete.

- 2) **Significantly less observations:** the new hydrodynamic code takes much more time per computation (several hours to days) than the simplified version used in [Chapter 2](#) (around 20 seconds). Thus, current developments are much more limited in the amount of simulations that we can conduct.
- 3) **Several output variables:** the new hydrodynamic code enables the extraction of diverse types of information of interest such as the maximum flooded area, the water height at surveillance points and coefficients of trafficability of critical roads. In this chapter we consider a total of 13 scalar outputs and we build a metamodel for each of them.

The Ant Colony algorithm proved its pertinency and effectiveness by finding high quality structural configurations for all the 13 outputs under analysis. In all cases, the selected configuration outperformed several others, including the default choices of using: (i) only a scalar representation of each functional input; (ii) the full set of scalar and functional representations of the inputs; and (iii) a scalar representation of each functional input, plus the functional representation of some key input variables. Even for the variables the most difficult to fit, our algorithm was able to find a configuration superior to the aforementioned alternatives. All the analysis was conducted using our R package `funGp` [20].

Chapter 6: Asymptotic properties of the maximum likelihood and cross validation estimators for transformed Gaussian processes

This chapter presents a theoretical study which is independent from the RISCOPE coastal flooding application. The chapter is an exact copy of our article [23], published in *Electronic Journal of Statistics*. There, we study the asymptotics of the maximum likelihood (ML) and cross validation (CV) estimators for the covariance parameters of a non-Gaussian process. We are motivated by the fact that many real applications involve output variables that present certain markedly non-Gaussian characteristics such as nonnegativity (e.g., [24]) and monotonicity (e.g., [25]), but they are still modeled as Gaussian without further considerations. This gives rise to the questions of: what can we expect from the estimation of the covariance parameters if we model a non-Gaussian process as Gaussian? in case of suspicion of non-Gaussianity would it be beneficial to apply some transformation to the output before implementing the Gaussian process model? Chapter 6 addresses the first question and gives the bases to undertake the second one.

In particular, we consider the case where the non-Gaussian process results from an unknown non-linear transformation of a Gaussian process. We further assume that the transformation is not modeled or estimated. We show that the ML and CV estimators are consistent and asymptotically normal, although they are defined as if the process was Gaussian. Our results can thus be interpreted as a robustness of (Gaussian) ML and CV towards non-Gaussianity. This study could be extended to the case where the transformation parameters are estimated along with the covariance parameters. We expect such an approach to be beneficial at least for taking into account non-Gaussian characteristics of the process of interest such as the aforementioned nonnegativity or more generally speaking, boundary constraints.

Chapter 7: Global conclusions and perspectives

This chapter summarizes the general conclusions of this thesis and presents some promising research lines for future development.

Chapter 2

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

The chapter in brief

This chapter presents our first approach to address the metamodeling problem studied in the RISCOPE project. It is an exact copy of our article [7], published in *Reliability Engineering & System Safety*. The chapter covers good part of the foundations of this thesis. It includes a discussion on related literature, a description of RISCOPE dataset, a synthesis of modeling through Gaussian processes, along with a detailed explanation of our metamodeling approach. Hence, this chapter is fundamental for the effective comprehension of the others.

At the time of writing the article, the target hydrodynamic code was in calibration. Therefore, we were using a simplified version of it, which ignored some of the inputs of the target code. The simplified code was much quicker to evaluate but less detailed and precise. This code, however, allowed us to generate a large number of observations, which were of great importance for the stabilization and optimization of our metamodeling routines. On the other hand, we were able to test various ways to include the functional inputs in the model. Through a large number of computer experiments, we showed the potential impact of the structural parameters in the predictability of the model. This article gave rise to our first prototype of methodology for the calibration of the structural parameters of the model, which was inspired by the Response Surface Methodology [8].

While relatively simple, the proposed methodology proved its effectiveness through an analytical case study and our coastal flooding application. In both cases it led to metamodel configurations of outstanding performance, able to accurately predict the output of the code. We proved the efficiency of this methodology through comparison against an exhaustive search approach. In most cases our methodology was able to find the optimal configuration, and made time savings of up to 76.7% and 38.7% with respect to the time spent by the exhaustive search in the analytic and the coastal flooding case, respectively.

Contents

2.1	Introduction	9
2.2	Motivating case: coastal flooding prediction at Gâvres, France	12
2.2.1	Hydrodynamic code	12
2.2.2	Dataset	13
2.3	Theoretical background	14
2.3.1	Scalar and functional inputs of computer codes	14
2.3.2	Gaussian process metamodeling of scalar-input codes	15
2.3.3	Gaussian process metamodeling of functional-input codes	16
2.3.3.1	Three distances for functional inputs	17
2.3.4	Gaussian process metamodeling with scalar and functional inputs	19
2.4	Exploration strategy	19
2.4.1	Scope of the methodology	21
2.4.2	Analytic case	22
2.4.2.1	Dataset	23
2.4.2.2	Screening	24
2.4.2.3	Cleaning and descent	27
2.5	Coastal flooding case study	30
2.5.1	Screening	30
2.5.2	Dimension selection based on projection error	36
2.5.2.1	Selecting the dimension for each input	36
2.5.2.2	Efficiency of configurations based on projection error	38
2.6	Robustness to changes in the amount of training/validation data	40
2.6.1	Experiment setting	40
2.6.2	Performance statistics	40
2.6.3	Analysis	41
2.7	Conclusions	44
Appendix 2.A	Dataset constitution	46
Appendix 2.B	Setting the projection coefficients	47
Appendix 2.C	Conditions and results for analytic case	49
Appendix 2.D	Conditions and results for coastal flooding case	50

José Betancourt^{1,2}, François Bachoc¹, Thierry Klein^{1,2}, Déborah Idier³,
Rodrigo Pedreros³, Jérémy Rohmer³

¹ Institut de Mathématiques de Toulouse, UMR 5219, Université de Toulouse, CNRS, UPS
IMT, 31062 Toulouse Cedex 9, France

² ENAC - Ecole Nationale de l'Aviation Civile, Université de Toulouse, France

³ 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 process 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

2.1 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 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 [26, 27, 28]. 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) [29, 30]. This difficulty is usually resolved by creating quick-to-evaluate mathematical emulators of those numerical codes, based on a limited collection of runs [6, 31, 32]; 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 difficulty 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 than 1000 time points (see e.g., [30]). 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 [33, 34, 30, 29]. The projection is sometimes preceded by time warping if an input shows a cyclical pattern which is consistent among functions [35]. 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 [30] or by minimizing the projection error [36]. Some alternative techniques better incorporate the idea of focusing on metamodel predictability. These include *scalar-on-function regression* [37, 38, 39], methods in the field of *active subspaces* [40], as well as stack models composed of a dimensionality reduction and a metamodeling technique put together and trained using backpropagation [41, 42, 43, 44]. Despite the advantages of these methods in terms of simplicity or predictability, 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 [45], 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 [46, 47, 48]. 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 [29]. 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 [48]. These weights are then optimized, typically through the Maximum Likelihood or Cross Validation method [49]. 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 [30]. In that case, each projection coefficient would be assigned a weight. Another alternative is to acknowledge 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 [29]. 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 2.2](#) 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 2.3](#). [Section 2.4](#) 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 2.5](#), we apply the exploration strategy to setup the metamodel for the coastal flooding application. In [Section 2.6](#), 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.

2.2 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 [50]. 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.

2.2.1 Hydrodynamic code

Here we consider a simplified fast running code defined on a cross-shore transect model (see Figure 2.1, 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 peak wave period (Tp). Each input should be provided to the system in a time series format, so that $\mathbf{Td} = (Td_t)_{t=1,\dots,L}$, 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*.

Calculations in the computer code involve the statistical model SWAN [51] and the EurOtop equations [52], both described below.

Inputs $\rightarrow$ **SWAN** $\rightarrow$ **EurOtop** $\rightarrow$ 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 ([52], 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.

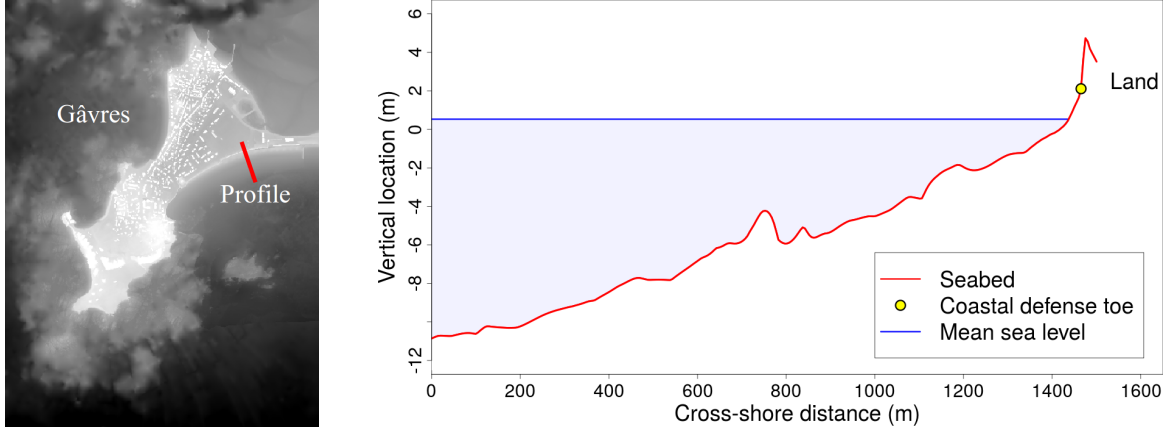
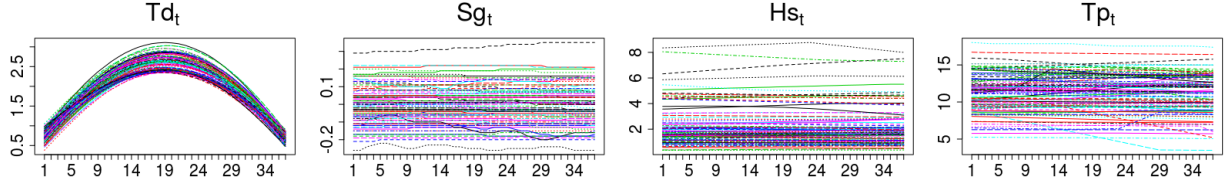


Figure 2.1 – Illustration of the cross-shore transect considered in the RISCOPE application. Vertical location is referenced to the altimetric French system IGN69.

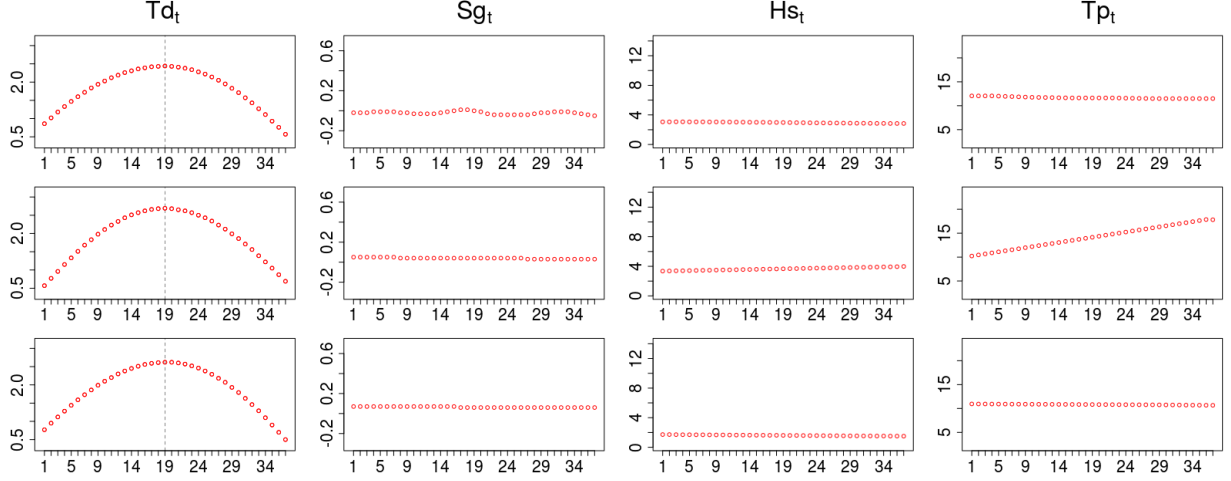
We are aware that the adoption of a cross shore configuration does not allow to properly model all the complexities of the phenomenon under study. However, in the frame of the RISCOPE project, the analysis presented here is considered an intermediate step in the development of methodologies for functional 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.

2.2.2 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 2.A), with bias corrections between the hindcasts through a quantile-quantile correction method (for more details, see [53]). 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 ± 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 2.2). 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 2.2 – 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 output depending on the interactions that happen within the code.

2.3 Theoretical background

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

- (b) When the code is $\mathbf{f} \mapsto f_{\text{code}}(\mathbf{f})$, with $\mathbf{f} = (f^{(1)}, \dots, f^{(df)})'$ and $f^{(k)} : T_k \subset \mathbb{R} \rightarrow \mathbb{R}$ for $k = 1, \dots, df$, we say that the code has $df \in \mathbb{N}$ functional inputs, we call $f^{(k)}$ for $k = 1, \dots, df$ a functional input, and we call $\mathbf{f}$ a vector of functional inputs. For simplicity, we may also refer to $\mathbf{f}$ as functional inputs.
- (c) When the code is $(\mathbf{x}, \mathbf{f}) \mapsto f_{\text{code}}(\mathbf{x}, \mathbf{f})$, we say that the code has ds scalar inputs and df functional inputs, and we use the same vocabulary as before for $\mathbf{x}$ and $\mathbf{f}$.

2.3.2 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 $\mathbf{x} = (x^{(1)}, \dots, x^{(ds)})' \in \mathbb{R}^{ds}$ and an output variable of interest $y \in \mathbb{R}$, with $y = f_{\text{code}}(\mathbf{x})$. 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 $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$. 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 $\mathbf{x}, \tilde{\mathbf{x}} \in \mathbb{R}^{ds}$, the Gaussian process model can be written as:

$$\xi(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot)), \quad (2.1)$$

with

$$m(\mathbf{x}) = \mathbb{E}[\xi(\mathbf{x})] \quad \text{and} \quad k(\mathbf{x}, \tilde{\mathbf{x}}) = \mathbb{E}[(\xi(\mathbf{x}) - m(\mathbf{x}))(\xi(\tilde{\mathbf{x}}) - m(\tilde{\mathbf{x}}))]. \quad (2.2)$$

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 [48] for more details). Furthermore, the Gaussian process model often has a very high prediction power compared to other approaches [29]. 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 $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^\top$ be the $n \times ds$ inputs matrix extracted from the learning set (where $\mathbf{x}_i$ for $i = 1, \dots, n$ is a column vector), and let $\mathbf{y} = (y_1, \dots, y_n)^\top$ be the vector of corresponding output values. Similarly, let $\mathbf{X}_* = (\mathbf{x}_{*,1}, \dots, \mathbf{x}_{*,n_*})^\top$ be a $n_* \times ds$ inputs matrix of prediction points. The Gaussian conditioning theorem (see e.g., [48]) implies that, conditionally to $\mathbf{y}$, ξ is a Gaussian process with mean and covariance functions m_n and k_n defined by

$$m_n(\mathbf{X}_*) := \mathbb{E}[\xi(\mathbf{X}_*) | \mathbf{y}] = K(\mathbf{X}_*, \mathbf{X})K(\mathbf{X}, \mathbf{X})^{-1}\mathbf{y} \quad (2.3)$$

and

$$\begin{aligned} k_n(\mathbf{X}_*, \mathbf{X}_*) &:= \text{Cov}[\xi(\mathbf{X}_*), \xi(\mathbf{X}_*) | \mathbf{y}] \\ &= K(\mathbf{X}_*, \mathbf{X}_*) - K(\mathbf{X}_*, \mathbf{X})K(\mathbf{X}, \mathbf{X})^{-1}K(\mathbf{X}, \mathbf{X}_*), \end{aligned} \quad (2.4)$$

where $K(\mathbf{X}, \mathbf{X})$ denotes the $n \times n$ matrix of covariances $(k(\mathbf{x}_i, \mathbf{x}_j))_{1 \leq i, j \leq n}$ among all pairs of training input points, and similarly for the other entries $K(\mathbf{X}, \mathbf{X}_*)$, $K(\mathbf{X}_*, \mathbf{X})$ and $K(\mathbf{X}_*, \mathbf{X}_*)$. We remark that $m_n(\mathbf{X}_*)$ and $k_n(\mathbf{X}_*, \mathbf{X}_*)$ are of the form

$$m_n(\mathbf{X}_*) = (\mathbb{E}[\xi(\mathbf{x}_{*,i}) | \xi(\mathbf{x}_1) = y_1, \dots, \xi(\mathbf{x}_n) = y_n])_{1 \leq i \leq n_*},$$

$$k_n(\mathbf{X}_*, \mathbf{X}_*) = (\text{Cov}[\xi(\mathbf{x}_{*,i}), \xi(\mathbf{x}_{*,j}) | \xi(\mathbf{x}_1) = y_1, \dots, \xi(\mathbf{x}_n) = y_n])_{1 \leq i, j \leq n_*}.$$

In practice the conditional mean (2.3) is used as an estimation of the true function f_{code} at the test points $\mathbf{X}_*$, while the conditional variance (2.4) is often interpreted as a measure of the local error of the prediction [32].

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 [48]. The selection of the covariance function encodes assumptions such as the degree of regularity of the underlying process [54]. A general expression for the covariance between any pair of scalar input points $\mathbf{x}, \tilde{\mathbf{x}} \in \mathbb{R}^{ds}$ is given by

$$k(\mathbf{x} - \tilde{\mathbf{x}}; \sigma^2, \boldsymbol{\theta}_s) = \sigma^2 R(\mathbf{x} - \tilde{\mathbf{x}}; \boldsymbol{\theta}_s), \quad (2.5)$$

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 $\boldsymbol{\theta}_s = (\theta_s^{(1)}, \dots, \theta_s^{(ds)})$. 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 [49] and [55]. 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

$$k(\boldsymbol{\tau}; \sigma^2, \boldsymbol{\theta}_s) = \sigma^2 \left(1 + \sqrt{5} \|\boldsymbol{\tau}\|_{L^2, \boldsymbol{\theta}_s} + \frac{5 \|\boldsymbol{\tau}\|_{L^2, \boldsymbol{\theta}_s}^2}{3} \right) \exp \left(-\sqrt{5} \|\boldsymbol{\tau}\|_{L^2, \boldsymbol{\theta}_s} \right), \quad (2.6)$$

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

$$\|\mathbf{x} - \tilde{\mathbf{x}}\|_{L^2, \boldsymbol{\theta}_s} = \sqrt{\sum_{k=1}^{ds} \frac{\|x^{(k)} - \tilde{x}^{(k)}\|^2}{(\theta_s^{(k)})^2}}. \quad (2.7)$$

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 $\mathbf{x} = \tilde{\mathbf{x}}$, then the correlation is 1, whereas if the distance between both vectors tends to infinity, then the correlation tends to 0.

2.3.3 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 $\mathbf{f} = (f^{(1)}, \dots, f^{(df)})'$, with $f^{(k)} : T_k \subset \mathbb{R} \rightarrow \mathbb{R}$ for $k = 1, \dots, df$, and an output variable of interest $y \in \mathbb{R}$, so that $y = f_{\text{code}}(\mathbf{f})$. Similarly to the scalar-input case, we assume that there is available a learning set $D = \{(\mathbf{f}_1, y_1), \dots, (\mathbf{f}_n, y_n)\}$. 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.

2.3.3.1 Three distances for functional inputs

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

$$\|\mathbf{f} - \tilde{\mathbf{f}}\|_{L^2, \boldsymbol{\theta}_f} = \sqrt{\sum_{k=1}^{df} \frac{\|f^{(k)} - \tilde{f}^{(k)}\|^2}{(\theta_f^{(k)})^2}}, \quad (2.8)$$

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

Note that (2.8) is just the straightforward extension of (2.7) 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 $\ell = 1, \dots, df$ have finite L^2 norm. In that case, (2.8) becomes

$$\|\mathbf{f} - \tilde{\mathbf{f}}\|_{F, \boldsymbol{\theta}_f} := \sqrt{\sum_{k=1}^{df} \frac{\int_{T_k} (f^{(k)}(t) - \tilde{f}^{(k)}(t))^2 dt}{(\theta_f^{(k)})^2}}, \quad (2.9)$$

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 $k = 1, \dots, df$ onto a subspace of finite, small or moderate dimension, and then use the L^2 norm of the projections in (2.8) instead of the L^2 norm of the original functions. For illustration, let $\Pi(f^{(k)})$ and $\Pi(\tilde{f}^{(k)})$ denote the projections of $f^{(k)}$ and $\tilde{f}^{(k)}$ onto the space generated by a basis $\mathbf{B}^{(k)} = \{B_1^{(k)}, \dots, B_{p_k}^{(k)}\}$. For $k = 1, \dots, df$, the expression to obtain $\Pi(f^{(k)})$ and $\Pi(\tilde{f}^{(k)})$ can then be written as

$$\Pi(f^{(k)})(t) = \sum_{r=1}^{p_k} \alpha_r^{(k)} B_r^{(k)}(t) \quad \text{and} \quad \Pi(\tilde{f}^{(k)})(t) = \sum_{r=1}^{p_k} \tilde{\alpha}_r^{(k)} B_r^{(k)}(t), \quad (2.10)$$

respectively. The projection dimension p_k has to be chosen strategically so that the functions are represented well enough and computations for the metamodel remain tractable. The projection coefficients $\boldsymbol{\alpha}^{(k)} = (\alpha_1^{(k)}, \dots, \alpha_{p_k}^{(k)})$ and $\tilde{\boldsymbol{\alpha}}^{(k)} = (\tilde{\alpha}_1^{(k)}, \dots, \tilde{\alpha}_{p_k}^{(k)})$ 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 displayed in (2.10).

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 $\|\Pi(f^{(k)}) - \Pi(\tilde{f}^{(k)})\|_{L^2}$ in (2.9) to obtain

$$\|\Pi(\mathbf{f}) - \Pi(\tilde{\mathbf{f}})\|_{D, \boldsymbol{\theta}_f} := \sqrt{\sum_{k=1}^{df} \frac{\int_{T_k} \left(\sum_{r=1}^{p_k} (\alpha_r^{(k)} - \tilde{\alpha}_r^{(k)}) B_r^{(k)}(t) \right)^2 dt}{(\theta_f^{(k)})^2}}. \quad (2.11)$$

As noted by [29], an efficient computation of $\|\Pi(f^{(k)}) - \Pi(\tilde{f}^{(k)})\|_{L^2}$ is possible, by reduction to a norm in $\mathbb{R}^{p_k}$:

$$\begin{aligned}
\|\Pi(f^{(k)}) - \Pi(\tilde{f}^{(k)})\|_{L^2}^2 &= \int_{T_k} \left(\sum_{r=1}^{p_k} (\alpha_r^{(k)} - \tilde{\alpha}_r^{(k)}) B_r^{(k)}(t) \right)^2 dt \\
&:= \int_{T_k} \left(\sum_{r=1}^{p_k} \delta_r^{(k)} B_r^{(k)}(t) \right)^2 dt \\
&= (\boldsymbol{\delta}^{(k)})' \mathbf{J}^{(k)} (\boldsymbol{\delta}^{(k)}) \\
&= \|\boldsymbol{\delta}^{(k)}\|_{\mathbf{J}^{(k)}}^2,
\end{aligned} \tag{2.12}$$

where $\mathbf{J}^{(k)}$ is the $p_k \times p_k$ Gram matrix $\left(\int_{T_k} B_i^{(k)}(t) B_j^{(k)}(t) dt \right)_{1 \leq i, j \leq p_k}$. The interesting fact about (2.12) is that $\mathbf{J}^{(k)}$ 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), $\mathbf{J}^{(k)}$ 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 (2.9) and (2.11):

$$\|\Pi(\mathbf{f}) - \Pi(\tilde{\mathbf{f}})\|_{S, \dot{\boldsymbol{\theta}}_{\mathbf{f}}} := \sqrt{\sum_{k=1}^{df} \sum_{r=1}^{p_k} \frac{(\alpha_r^{(k)} - \tilde{\alpha}_r^{(k)})^2}{(\dot{\theta}_{f,r}^{(k)})^2}}. \tag{2.13}$$

Note that here we denote the vector of length-scale coefficients by $\dot{\boldsymbol{\theta}}_{\mathbf{f}}$, with elements $(\dot{\theta}_{f,r}^{(k)})_{1 \leq r \leq p_k, 1 \leq k \leq df}$, to differentiate with the shorter vector $\boldsymbol{\theta}_{\mathbf{f}}$, with elements $(\theta_f^{(k)})_{1 \leq k \leq df}$ used in (2.9) and (2.11). Also note that (2.13) can be interpreted as if each projection coefficient $\alpha_r^{(k)}$ was taken as an individual scalar input of the model, since (2.13) matches the structure of the anisotropic L^2 norm for scalars shown in (2.7).

For applications of the three approaches, the reader is referred to [56], [29] and [30], in the corresponding order. For the sake of theory, we expressed (2.9), (2.11) and (2.12) in terms of infinite-dimensional inputs $f^{(k)} : T_k \subset \mathbb{R} \rightarrow \mathbb{R}$. However, in practice one typically does not have access to the infinite-dimensional function, but to a vectorial representation of it $\left(f^{(k)}(t_1^{(k)}) \dots, f^{(k)}(t_{L_k}^{(k)}) \right)'$, with $\{t_1^{(k)}, \dots, t_{L_k}^{(k)}\} \subset T_k$, as is the case in our coastal flooding application. In (2.9), 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 (2.11) and (2.12) 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 (2.11) and (2.13).

2.3.4 Gaussian process metamodeling with scalar and functional inputs

Our coastal flooding application matches the functional-input setting $\mathbf{f} \mapsto f_{\text{code}}(\mathbf{f})$, with $\mathbf{f} = (Td, Sg, Hs, Tp)'$. 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 $\mathbf{x} = (x^{(1)}, \dots, x^{(ds)})' \in \mathbb{R}^{ds}$, a vector of functional inputs $\mathbf{f} = (f^{(1)}, \dots, f^{(df)})'$, with $f^{(k)} : T_k \subset \mathbb{R} \rightarrow \mathbb{R}$ for $k = 1, \dots, df$, and an output variable of interest $y \in \mathbb{R}$, with $y = f_{\text{code}}(\mathbf{x}, \mathbf{f})$. We model the correlation between scalar points and the correlation between functional points as described in Sections 2.3.2 and 2.3.3, respectively. To integrate both types of inputs in the model, we follow the approach in [29] and adopt an anisotropic, tensor-product kernel of the form

$$\text{Cov}(\xi(\mathbf{x}, \mathbf{f}), \xi(\tilde{\mathbf{x}}, \tilde{\mathbf{f}})) = \sigma^2 R(\mathbf{x} - \tilde{\mathbf{x}}; \boldsymbol{\theta}_s) R(\mathbf{f} - \tilde{\mathbf{f}}; \boldsymbol{\theta}_z), \quad (2.14)$$

with $\boldsymbol{\theta}_z$ denoting either the vector $\boldsymbol{\theta}_f$ or the vector $\dot{\boldsymbol{\theta}}_f$, depending on whether the distance $\|\cdot\|_{D, \boldsymbol{\theta}_f}$ or $\|\cdot\|_{S, \dot{\boldsymbol{\theta}}_f}$ is used. To illustrate, if we take our tensor-product kernel from the Matérn 5/2 family (2.6) and we use the distance $\|\cdot\|_{D, \boldsymbol{\theta}_f}$ for the functional inputs, we obtain:

$$\begin{aligned} \text{Cov}(\xi(\mathbf{x}, \mathbf{f}), \xi(\tilde{\mathbf{x}}, \tilde{\mathbf{f}})) = \sigma^2 & \left(1 + \sqrt{5} \|\mathbf{x} - \tilde{\mathbf{x}}\|_{L^2, \boldsymbol{\theta}_s} + \frac{5 \|\mathbf{x} - \tilde{\mathbf{x}}\|_{L^2, \boldsymbol{\theta}_s}^2}{3} \right) \\ & \exp(-\sqrt{5} \|\mathbf{x} - \tilde{\mathbf{x}}\|_{L^2, \boldsymbol{\theta}_s}) \\ & \left(1 + \sqrt{5} \|\mathbf{f} - \tilde{\mathbf{f}}\|_{D, \boldsymbol{\theta}_f} + \frac{5 \|\mathbf{f} - \tilde{\mathbf{f}}\|_{D, \boldsymbol{\theta}_f}^2}{3} \right) \\ & \exp(-\sqrt{5} \|\mathbf{f} - \tilde{\mathbf{f}}\|_{D, \boldsymbol{\theta}_f}). \end{aligned} \quad (2.15)$$

2.4 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 2.3 and its main steps are briefly described below.

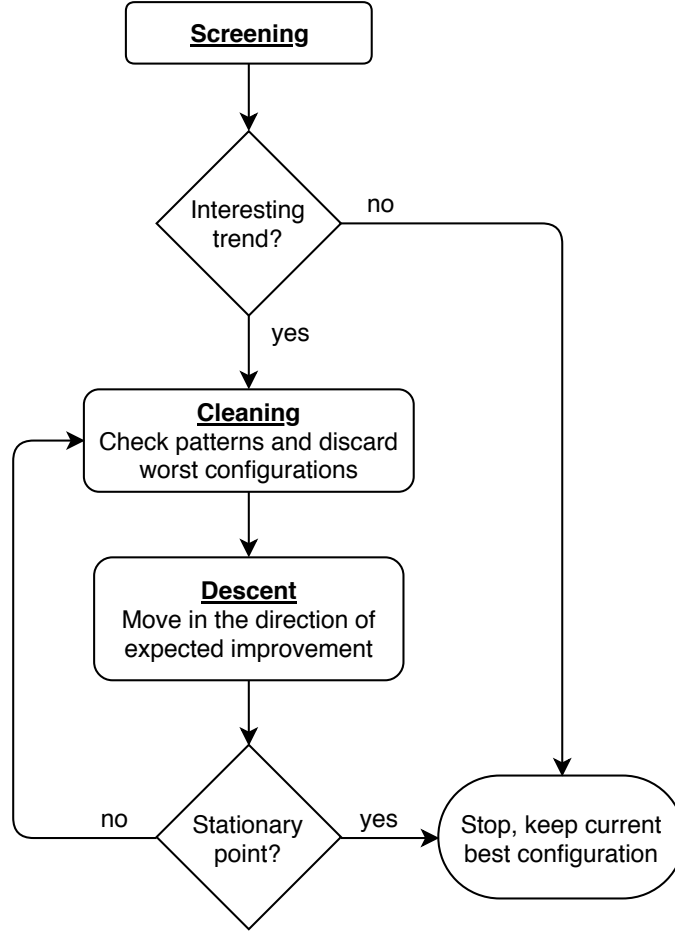


Figure 2.3 – Exploration strategy flowchart.

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 instance, 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.

Since 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 [8, 57], this stage is repeated until a stationary point is identified.

2.4.1 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 2.6](#). For cases of very limited data the performance of each configuration could be assessed by means of cross-validation/bootstrap methods [58]. 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 [59, 60] and polynomial chaos expansions [35].

- *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 [61], Artificial bee colony programming [62] and Genetic programming [63] 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., [64] or [24]). If the functional inputs are functions from $T \subset \mathbb{R}^d \rightarrow \mathbb{R}$, then for tensorized projection spaces, the projection dimension is of the form $p^{(1)} \times \dots \times p^{(d)}$, where $p^{(1)}, \dots, p^{(d)}$ 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., [65]). 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., [34] or [65]). 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 objective 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., [66] and [67], 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) [68] or the quantile function [69] 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.

2.4.2 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 [29], with a slight different domain for the functional inputs. In [29], 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 $[0, 1]$ to $\mathbb{R}$. Consider a black box computer code receiving the scalar inputs $\mathbf{x} = (x^{(1)}, x^{(2)}) \in [0, 1]^2$ and the continuous functional inputs $\mathbf{f} = (f^{(1)}, f^{(2)}) \in \mathcal{F}^2$ defined as:

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

Note that $\mathcal{G}$ is an instrinsic 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.

2.4.2.1 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 uniformly sampled over their domain. For the functional part, we followed the approach proposed in [29] 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 [70] 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 [29]. The full dataset and a set of 25 trajectories of the function $f^{(1)}$ are shown in Figures 2.4a and 2.4b, respectively.

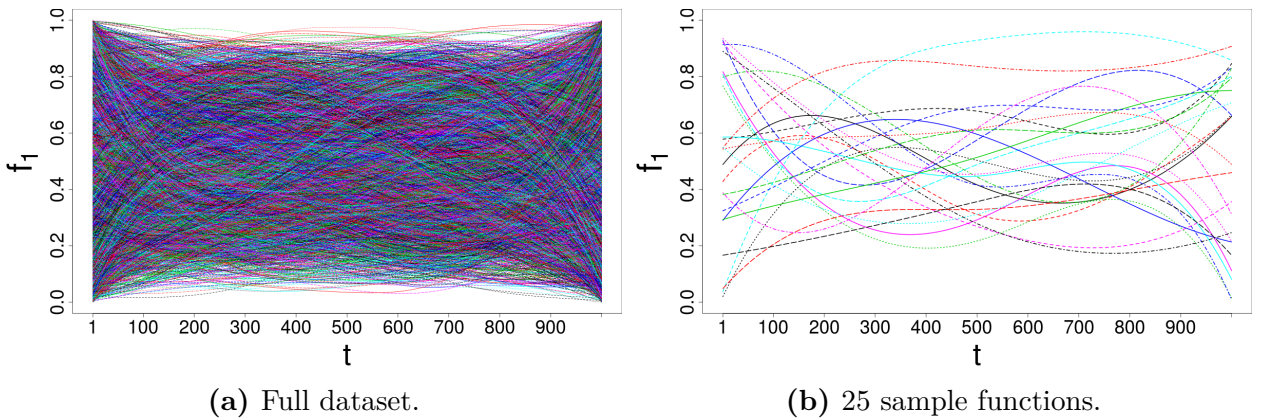


Figure 2.4 – Illustration of the functional input f_1 .

2.4.2.2 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 $\ddot{\mathbf{f}} = (\ddot{f}^{(1)}, \ddot{f}^{(2)})$ 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:

$$\begin{aligned} \mathcal{M}_{00} : [0, 1]^2 \times [0, 1]^2 &\rightarrow \mathbb{R}, \\ (\mathbf{x}, \ddot{\mathbf{f}}) &\mapsto \mathcal{M}_{00}(\mathbf{x}, \ddot{\mathbf{f}}). \end{aligned} \quad (2.16)$$

For the functional metamodels, let us consider a shifted version of $\mathbf{f}$, computed as $\tilde{\mathbf{f}} = \mathbf{f} - \ddot{\mathbf{f}}$. Then, let $\mathbf{\Pi} = (\Pi_1, \Pi_2)$ denote the vector of projections of the elements in $\tilde{\mathbf{f}}$ onto a space of dimension p . For every functional metamodel, we keep $\mathbf{x}$ and $\ddot{\mathbf{f}}$ as inputs and we add at least one element of $\mathbf{\Pi}_p$. 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 (2.17), (2.18) and (2.19), respectively.

$$\begin{aligned} \mathcal{M}_{f0} : [0, 1]^2 \times [0, 1]^2 \times \mathbb{R}^p &\rightarrow \mathbb{R}, \\ (\mathbf{x}, \ddot{\mathbf{f}}, \Pi_1) &\mapsto \mathcal{M}_{f0}(\mathbf{x}, \ddot{\mathbf{f}}, \Pi_1). \end{aligned} \quad (2.17)$$

$$\begin{aligned} \mathcal{M}_{0f} : [0, 1]^2 \times [0, 1]^2 \times \mathbb{R}^p &\rightarrow \mathbb{R}, \\ (\mathbf{x}, \ddot{\mathbf{f}}, \Pi_2) &\mapsto \mathcal{M}_{0f}(\mathbf{x}, \ddot{\mathbf{f}}, \Pi_2). \end{aligned} \quad (2.18)$$

$$\begin{aligned} \mathcal{M}_{ff} : [0, 1]^2 \times [0, 1]^2 \times (\mathbb{R}^p)^2 &\rightarrow \mathbb{R}, \\ (\mathbf{x}, \ddot{\mathbf{f}}, \mathbf{\Pi}) &\mapsto \mathcal{M}_{ff}(\mathbf{x}, \ddot{\mathbf{f}}, \mathbf{\Pi}). \end{aligned} \quad (2.19)$$

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, (2.17), (2.18) and (2.19), 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 2.1), 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 $\|\cdot\|_{D, \theta_f}$ as $\|\cdot\|_{D, \theta}$ and $\|\cdot\|_{S, \dot{\theta}_f}$ as $\|\cdot\|_{S, \dot{\theta}}$. 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., [29, 71, 33] for applications of B-splines

and [30, 72] for applications of PCA). Both methods work with a projection of the form (2.10) 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 [71] and [73], respectively. Other projection methods such as PLS [74] 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 $p_1 = \dots = p_{df} = p$, with the notation of Section 2.3.

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	$\ \cdot\ _{D,\theta}, \ \cdot\ _{S,\theta}$

Table 2.1 – Analytic case: parameters and levels for the screening stage.

In all cases, we set the projection coefficients $\alpha_1^{(k)}, \dots, \alpha_{p_k}^{(k)}$ using an ordinary least squares formulation (see Appendix 2.B). We used the Matérn 5/2 kernel (2.6) and estimated the hyperparameters of each metamodel by maximizing the joint likelihood of the data [75, 49] in a similar way to the R package DiceKriging [76]. 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 determination R^2 for a test sample, i.e., for prediction residuals [77]. For a test set of n_* output values $y_{*,1}, \dots, y_{*,n_*}$, with average denoted by $\bar{y}_*$, and corresponding predictions $\hat{y}_{*,1}, \dots, \hat{y}_{*,n_*}$, the Q^2 is defined as

$$Q^2 = 1 - \frac{\sigma_E^2}{\sigma_T^2}, \quad (2.20)$$

with

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

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 (2.20) for each of the 30 samples and then taking the average of the results:

$$\tilde{Q}^2 := \frac{1}{30} \sum_{s=1}^{30} Q_s^2. \quad (2.21)$$

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:

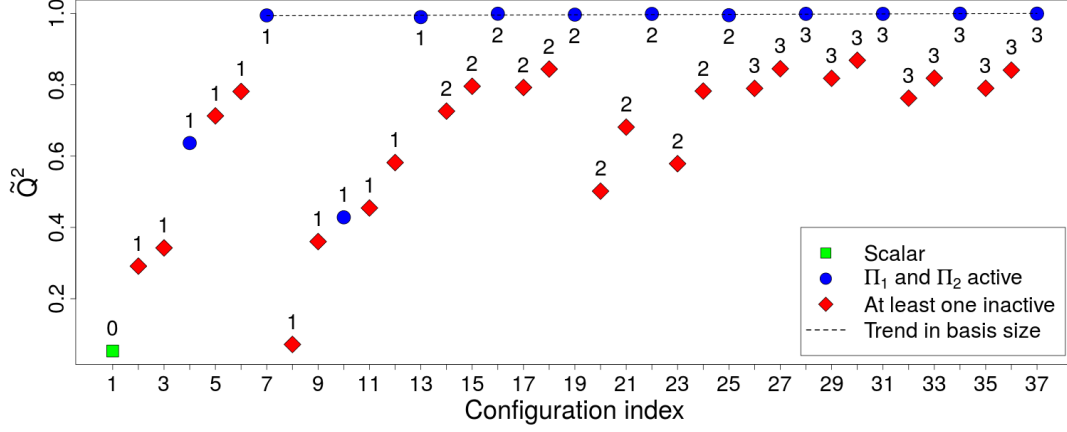
- (i) Stop if the slope of a linear least squares fitting to the best $\tilde{Q}^2$ values of the screening is lower than a reference value m^* ;
- (ii) Stop if a stationary point or plateau is reached. Each time the experiment is expanded to a greater level $\dot{p}$ of projection dimension, compute 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 [78] and heuristics [79].

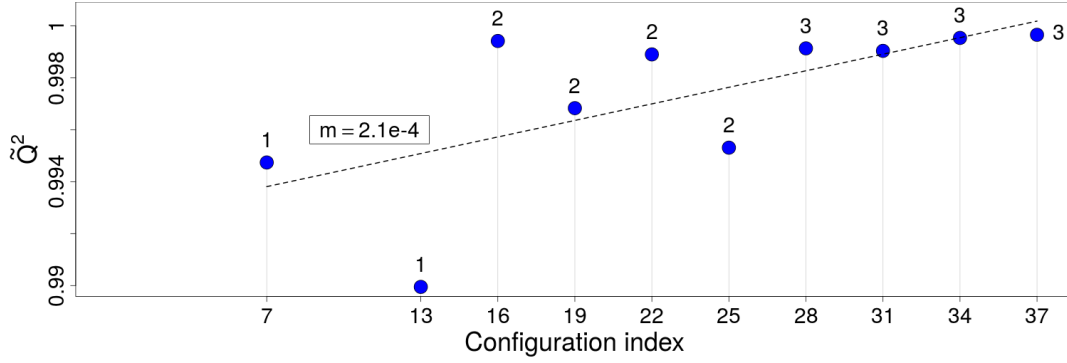
Figure 2.5 illustrates the performance in terms of $\tilde{Q}^2$ of the 37 metamodel configurations, using 800 training points and 1500 validation points. Those results were obtained using $m^* = 10^{-4}$ and $z = 3$ for the stopping conditions, which based on a set of preliminary tests seem to provide 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 2.C](#), [Table 2.7](#).



(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.

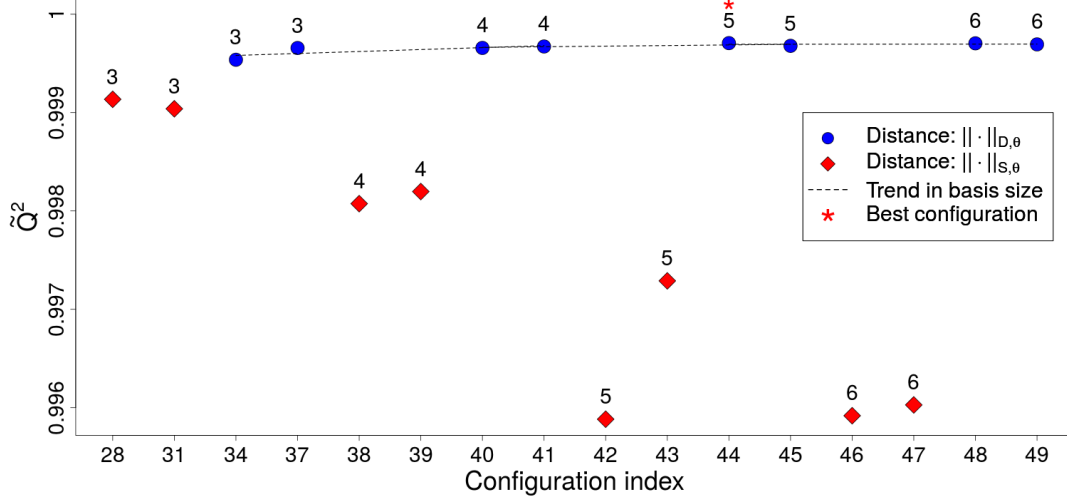
Figure 2.5 – Analytic case: results of the screening experiment. Points are labeled by projection dimension p ; the label 0 corresponds to the scalar metamodel.

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 $16 + 3i$, $i = 0 \dots, 7$) 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.

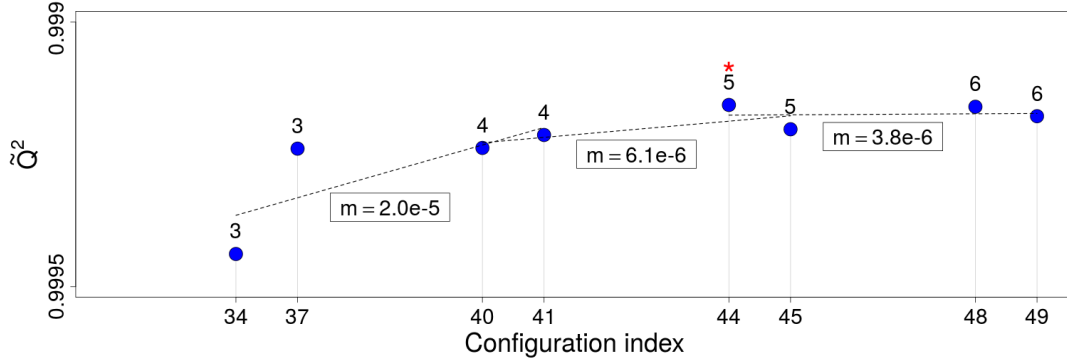
2.4.2.3 Cleaning and descent

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 2.6. A detailed list of the experimental conditions and corresponding results of this stage is provided in Appendix 2.C, Table 2.8. 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 $\|\cdot\|_{S,\theta}$, as those were clearly dominated by configurations using the decomposition-based distance $\|\cdot\|_{D,\theta}$. Then, $p = 5$ to $p = 6$ we registered a third flat step, which fulfilled our second 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 2.6 – 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 $\|\cdot\|_{D,\theta}$ (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 (condition 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 2.7. For this sample, the proportion of output values lying within the confidence intervals at 99%, 95% and 90% was 89%, 77% and 68%, respectively.

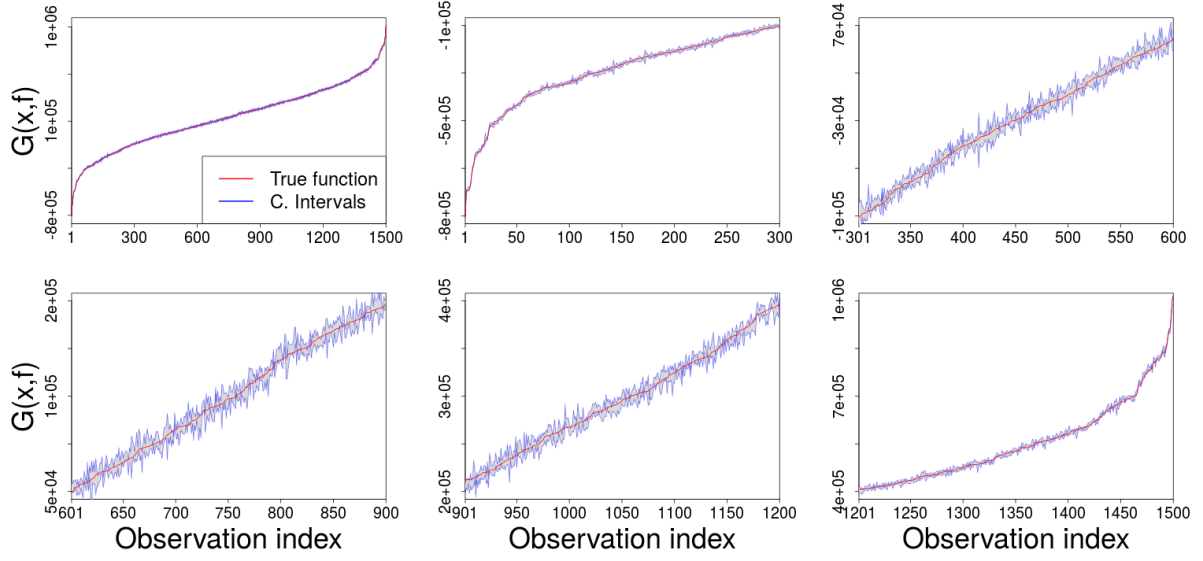


Figure 2.7 – Analytic case: fitting of the best performing sample of the best configuration. Left-top subplot illustrates the whole set of 1500 output values in increasing order; this subplot is then subdivided to generate the remaining 5 subplots by splitting the abscissa into 5 sections and zooming the resulting figures.

On the other hand, the calibration plot for the best and worst samples of the best configuration are presented in Figures 2.8a and 2.8b, respectively. Based on these results, we may conclude that this metamodel provides good predictions with no evident fitting problems (e.g., skewness, heavy tails).

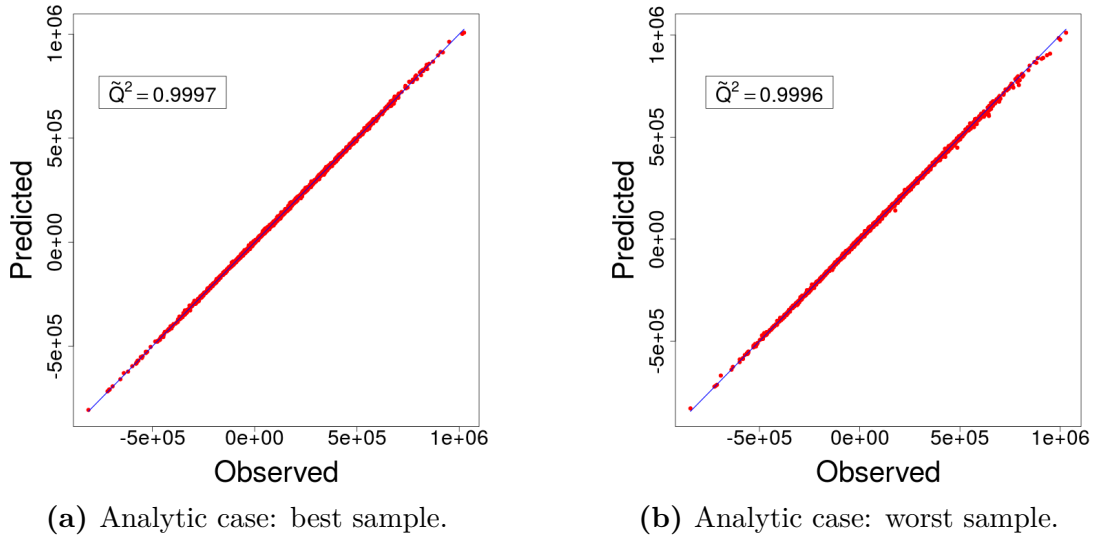


Figure 2.8 – Analytic case: calibration plot with 1500 data points for the best and worst samples of the best performing metamodel (configuration 44).

2.5 Coastal flooding case study

In this section, we address the application case introduced in [Section 2.2](#). 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 2.2.1](#)). 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 [80, 81]. 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 comparison of the two approaches for our application case as an interesting topic of future research.

2.5.1 Screening

Following the exploration methodology described in [Section 2.4](#), 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 2.2](#)) 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 $\mathbf{f}$. From the physical perspective, the $\mathbf{Td}$ 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. For $\mathbf{Sg}$, $\mathbf{Hs}$ and $\mathbf{Tp}$ we use the average of the series over 37 time points, given their smooth and almost constant behavior in the historical dataset (see [Figure 2.2](#)). Using a similar notation to that used for the analytic case, we define the scalar benchmark metamodel as:

$$\begin{aligned} \mathcal{M}_{000} : \mathbb{R}^4 &\rightarrow \mathbb{R}, \\ \ddot{\mathbf{f}} &\mapsto \mathcal{M}_{000}(\ddot{\mathbf{f}}). \end{aligned} \tag{2.22}$$

As before, the functional metamodels require the definition of a shifted version of $\mathbf{f}$ computed as $\tilde{\mathbf{f}} = \mathbf{f} - \ddot{\mathbf{f}}$. 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 2.4.1](#) 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 2.9](#)). 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 2.4.2](#)).

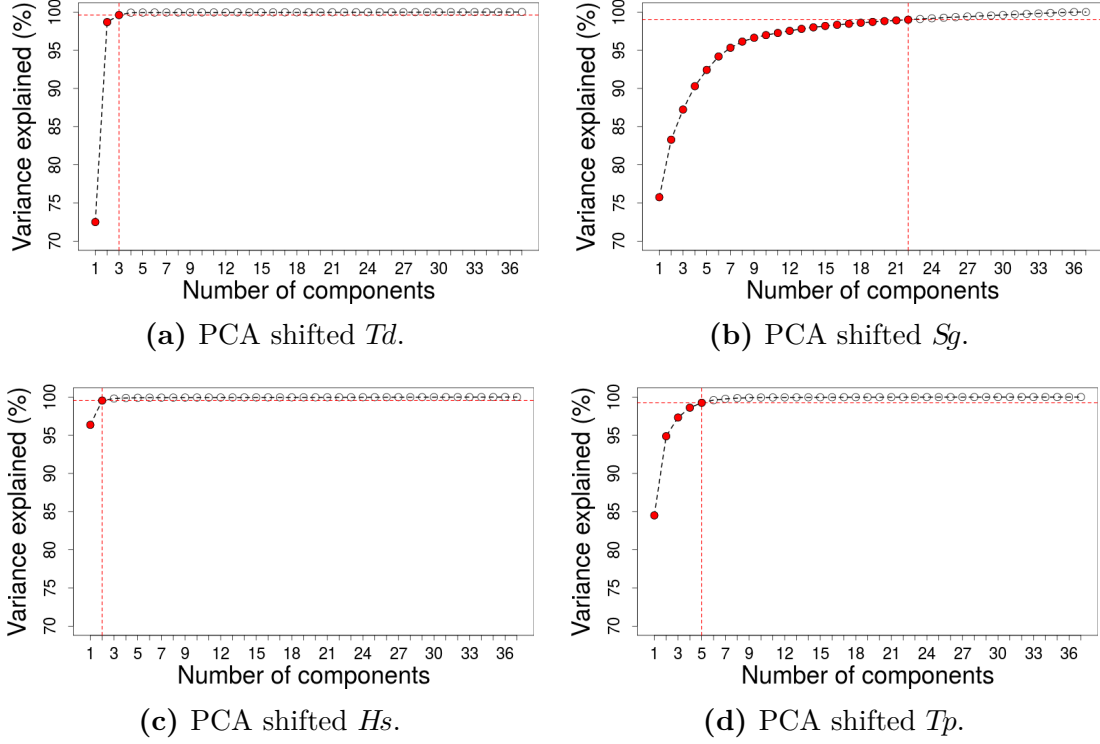


Figure 2.9 – Coastal flooding case: PCA on the shifted inputs; the dotted lines indicate the number of principal components required to explain at least the 99% of the data variability.

Based on the plots, $\tilde{H}s$ is the input requiring fewer components to be well described, while $\tilde{S}g$ and $\tilde{T}p$ require a considerable number of components. Hence, we discard $\tilde{H}s$ as a functional input of the metamodel, but keep its scalar representation $\tilde{H}s$ active as it does not affect the number of experimental conditions to run, but may help to improve predictions. Although $\tilde{T}d$ 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 $\tilde{T}d$ in the experiment.

Now we let $\mathbf{\Pi} = (\Pi_1, \Pi_2, \Pi_3)$ denote the vector of projections of dimension p for $\tilde{T}d$, $\tilde{S}g$ and $\tilde{T}p$. For every functional metamodel, we keep all the elements in $\tilde{\mathbf{f}}$ active and we add at least one element of $\mathbf{\Pi}$ as a functional input. Functional metamodels are defined similarly to 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 (2.23), (2.24) and (2.25), respectively.

$$\begin{aligned} \mathcal{M}_{f00} : [0, 1]^4 \times \mathbb{R}^p &\rightarrow \mathbb{R}, \\ (\tilde{\mathbf{f}}, \Pi_1) &\mapsto \mathcal{M}_{f00}(\tilde{\mathbf{f}}, \Pi_1). \end{aligned} \quad (2.23)$$

$$\begin{aligned} \mathcal{M}_{0ff} : [0, 1]^4 \times \mathbb{R}^p \times \mathbb{R}^p &\rightarrow \mathbb{R}, \\ (\tilde{\mathbf{f}}, \Pi_2, \Pi_3) &\mapsto \mathcal{M}_{0ff}(\tilde{\mathbf{f}}, \Pi_2, \Pi_3). \end{aligned} \quad (2.24)$$

$$\begin{aligned} \mathcal{M}_{fff} : [0, 1]^4 \times (\mathbb{R}^p)^3 &\rightarrow \mathbb{R}, \\ (\tilde{\mathbf{f}}, \mathbf{\Pi}) &\mapsto \mathcal{M}_{fff}(\tilde{\mathbf{f}}, \mathbf{\Pi}). \end{aligned} \quad (2.25)$$

The interpretation of this notation is analogous to that for the analytic case. The subscript of $\mathcal{M}$ indicates which functional decompositions are active. The notation remains generic in the sense that (2.23), (2.24) and (2.25), 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. Those correspond to the scalar metamodel plus all combinations of the levels of the parameters listed in Table 2.2, 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 2.D, Table 2.9.

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	$\ \cdot\ _{D,\theta}$, $\ \cdot\ _{S,\theta}$

Table 2.2 – 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 were 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 2.B). A constrained or weighted constrained formulation could also be suitable choices here. Those are used for further analysis in Section 2.5.2.

For the weighted least squares formulation we denote the vector of weights by $\mathbf{w} = (w_1, \dots, w_T)$, with $T = 37$ and w_t given by:

$$w_t = \begin{cases} 1 & : \text{ if } t = t_* \\ \lambda & : \text{ if } 0 < |t - t_*| \leq \delta \\ \lambda \exp\left(-\frac{(|t - t_*| - \delta)^2}{2\sigma^2}\right) & : \text{ if } |t - t_*| > \delta, \end{cases} \quad (2.26)$$

with $\sigma^2 = -(\omega^2)/(2\ln(\gamma))$ controlling the decay rate of the function. Models like (2.26) are often used to represent the relevance of results for queries on search engines [82]. 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_* .

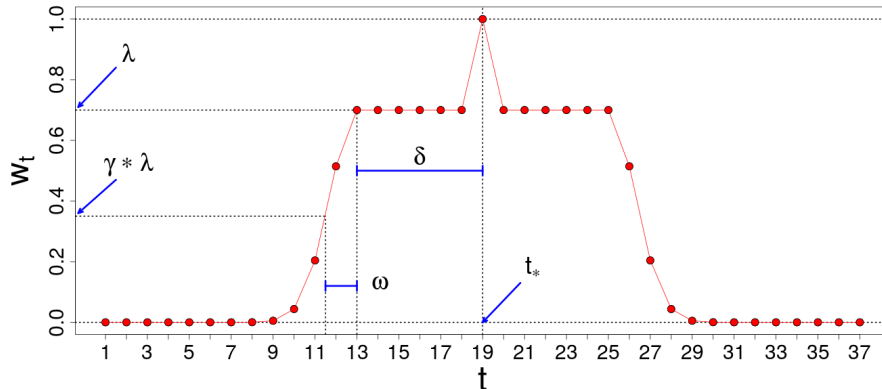
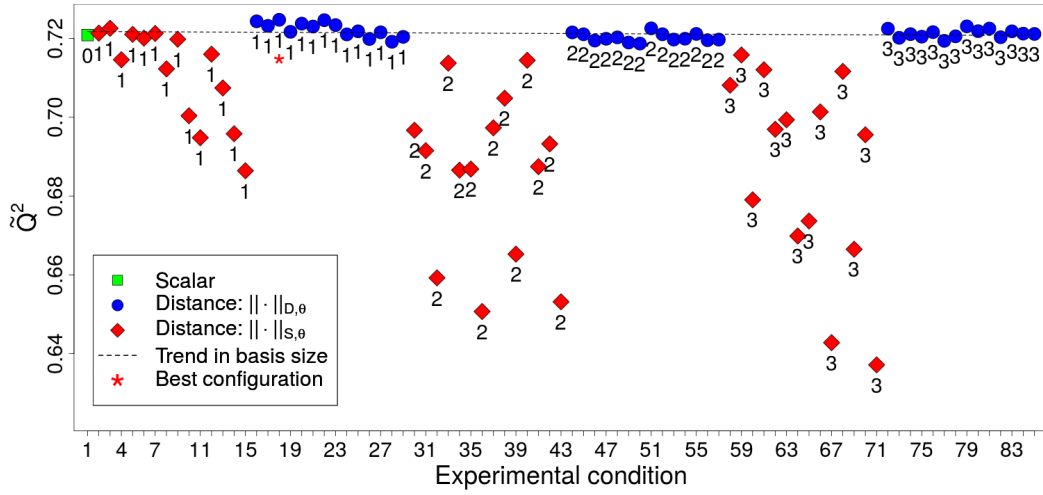


Figure 2.10 – Weighting function for the coastal flooding case. The parameters λ , γ , ω and δ , controlling the shape of the curve, were set to 0.7, 0.5, 1.5 and 6, respectively.

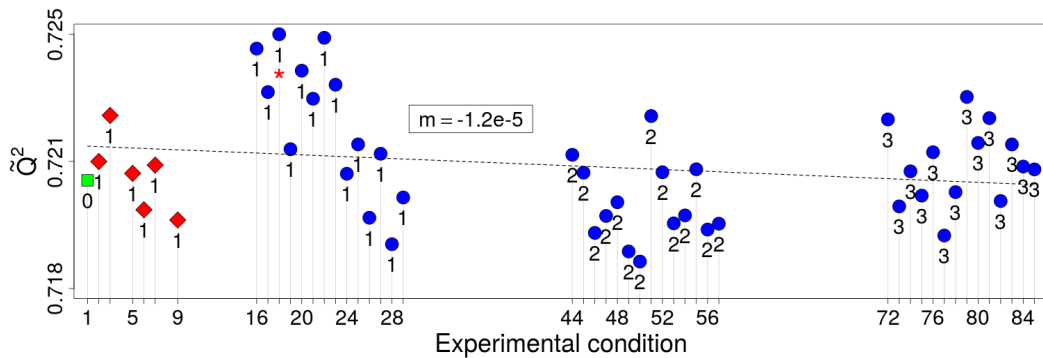
A particular shape can be given to the curve by setting the value of its parameters as follows. First, δ should be chosen from $[0, t_*]$. Then, ω should be set in $[0, t_* - \delta]$. Finally, λ and γ should be set, each in $[0, 1]$. This setting along with the first case of (2.26) ensure that the greatest possible score in $\mathbf{w}$ is 1. The weighting curve produced by such a model is illustrated in Figure 2.10 with the parameterization used for the coastal flooding case.

Figure 2.11 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 exposition, Figure 2.11 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 $\|\cdot\|_{D,\theta}$,
- (iii) all metamodels using a scalar distance $\|\cdot\|_{S,\theta}$.



(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 2.11 – 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 $\|\cdot\|_{D,\theta}$ performed similarly and outperformed almost every configuration using the scalar distance $\|\cdot\|_{S,\theta}$, 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 fulfill stopping condition number two. 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 $\tilde{Td}$ and $\tilde{Sg}$, using the decomposition-based distance $\|\cdot\|_{D,\theta}$. However, in practice any of the dominant configurations included in [Figure 2.11b](#) 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 2.D, Table 2.9](#)).

It is inquiring to see that the $\tilde{Q}^2$ values reported in [Figure 2.11](#) 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 2.12](#) 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 $\tilde{Q}^2$.

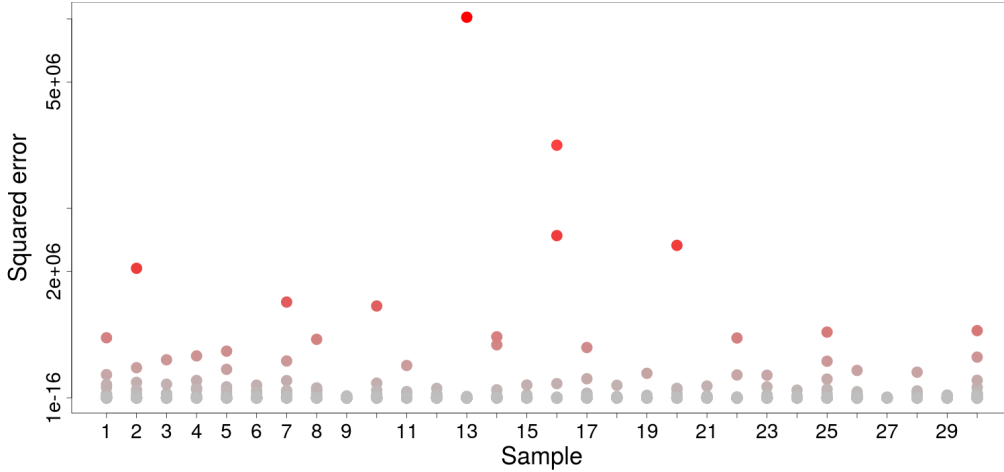
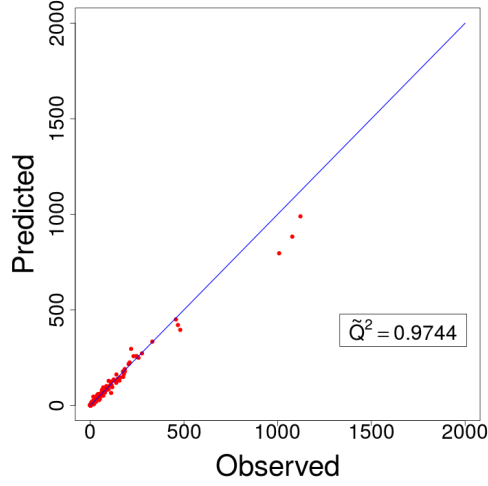
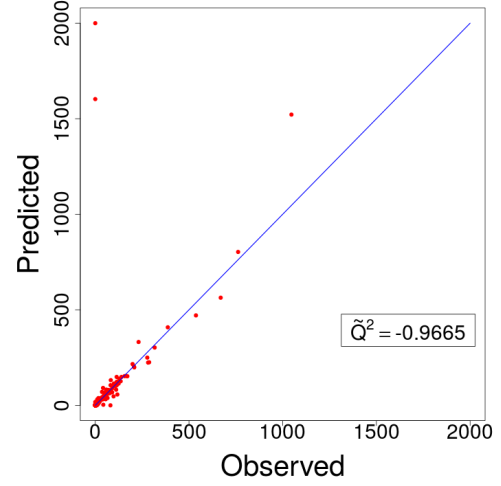


Figure 2.12 — Coastal flooding case: squared errors for each of the 1500 validation points in each of the 30 samples of configuration 18.

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 2.2](#)), we found 90% of the output values below 4 m^3 although the largest output value found was 3455 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 2.13](#)). A possible way to deal with this issue is to use sequential design techniques [\[83, 84\]](#) 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.



(a) Coastal flooding: best sample.



(b) Coastal flooding: worst sample.

Figure 2.13 – 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 appropriate way to assess the absolute quality of each metamodel is to compute the $\tilde{Q}^2$, which we define as the $\tilde{Q}^2$ in (2.21), but using $\tilde{\sigma}_E^2 = \text{median}(\{(y_{*,i} - \hat{y}_{*,i})^2\}_{i=1,\dots,n_*})$ instead of σ_E^2 and $\tilde{\sigma}_T^2 = \text{median}(\{(y_{*,i} - \bar{y}_*)^2\}_{i=1,\dots,n_*})$ instead of σ_T^2 . Here $\text{median}(\{u_1, \dots, u_a\})$ 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 $\tilde{Q}^2$, 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.

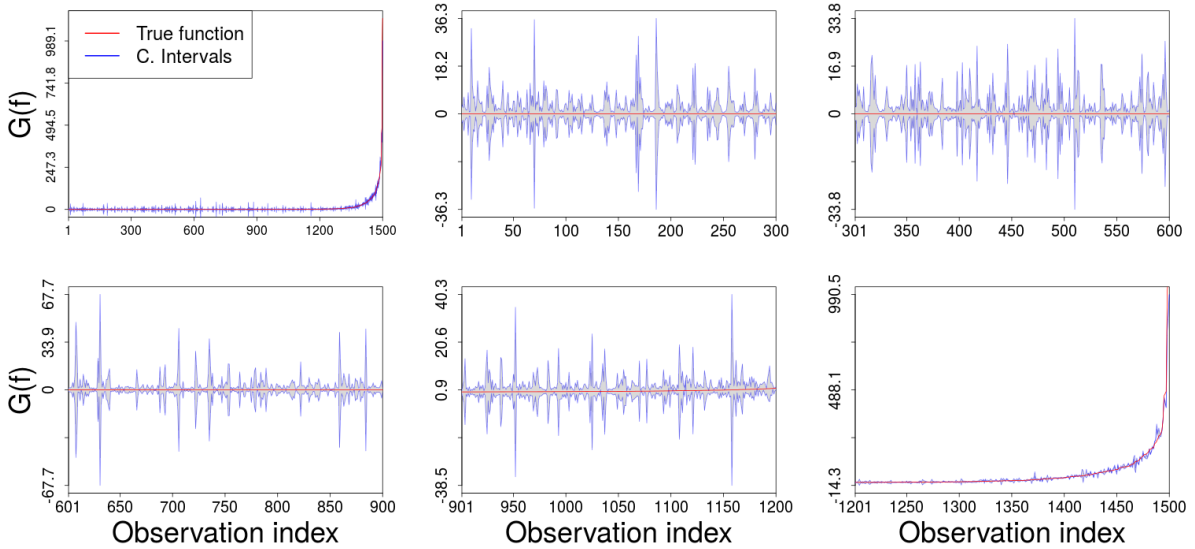


Figure 2.14 – Coastal flooding case: fitting of the best performing sample of the best configuration. Left-top subplot illustrates the whole set of 1500 output values in increasing order; this subplot is then subdivided to generate the remaining 5 subplots by splitting the abscissa into 5 sections and zooming the resulting figures.

The fitting of the ordered true output for the best performing sample of the best configuration is illustrated in [Figure 2.14](#). 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 2.2.1](#), 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 2.4.2](#). There, the best configuration found was a metamodel using a projection of dimension 5.

2.5.2 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.

2.5.2.1 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 $t = \{13, \dots, 25\}$, which corresponds to the moment of maximum tide ± 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 2.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 2.15](#).

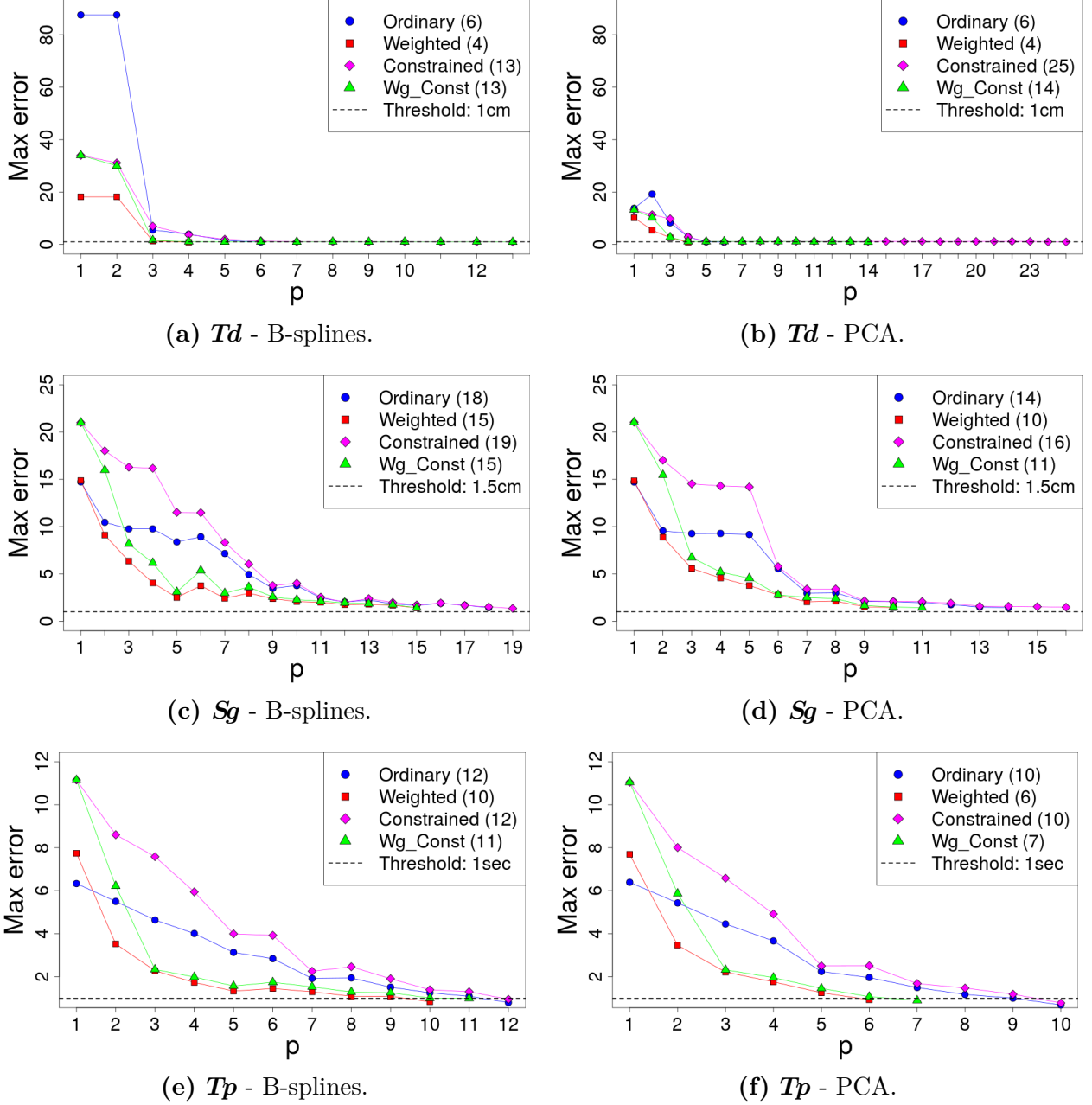


Figure 2.15 – Maximum projection error within the critical time window as a function of the projection dimension p . Td and Sg error in centimeters and Tp error in seconds. In the legends, the quantity in parenthesis indicates the value of p needed to meet the tolerance.

As suggested in [Section 2.4](#), 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 2.16](#) 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 were 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 2.15e. At $p = 5$ the error was considerably low, however, it required $p = 10$ to meet the tolerance.

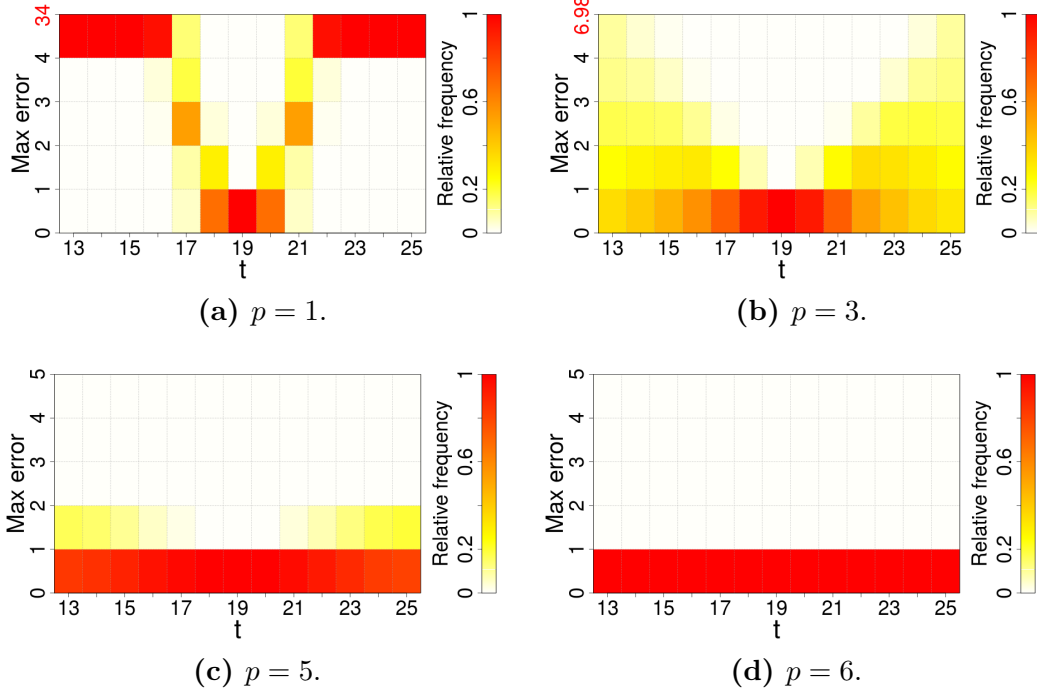


Figure 2.16 – Distribution of errors (in centimeters) for points in the critical window using the constrained least squares optimization formulation and B-splines for the projection of Td .

2.5.2.2 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 ($\|\cdot\|_{S,\theta}$ or $\|\cdot\|_{D,\theta}$). 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 2.3 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 2.5.1.

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

Table 2.3 – 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 2.D](#), [Table 2.10](#). The $\tilde{Q}^2$ of each of these experimental conditions is reported in [Figure 2.17](#), 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 $\mathbf{Td}$, using the decomposition-based distance $\|\cdot\|_{D,\theta}$.

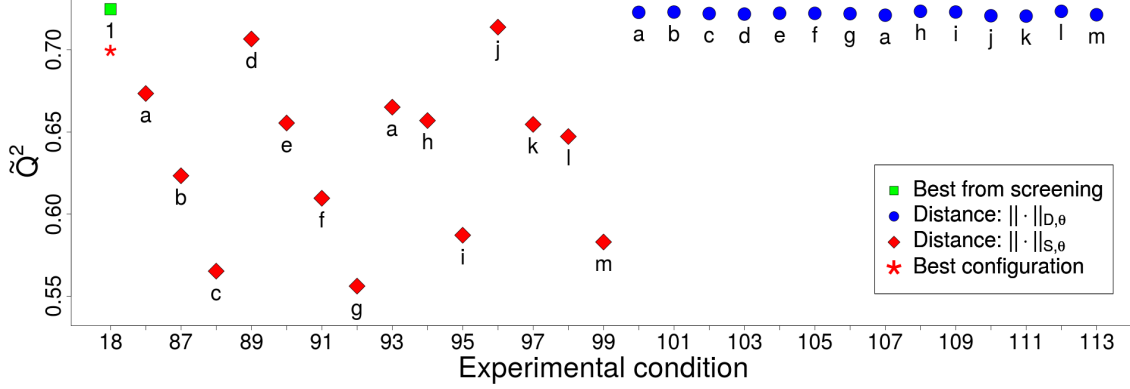


Figure 2.17 – Coastal flooding case: performance of metamodels with the projection dimension based on projection error. Points are labeled by projection dimension; as it varies from one input to the other in the new experiment, those points are labeled using an alphabetic convention where each letter is matched to a triple of integers denoting the projection dimension for each input in the order $\mathbf{Td}, \mathbf{Sg}, \mathbf{Tp}$: $a : (4, 0, 0)$, $b : (0, 15, 0)$, $c : (4, 15, 0)$, $d : (0, 0, 10)$, $e : (4, 0, 10)$, $f : (0, 15, 10)$, $g : (4, 15, 10)$, $h : (0, 10, 0)$, $i : (4, 10, 0)$, $j : (0, 0, 6)$, $k : (4, 0, 6)$, $l : (0, 10, 6)$, $m : (4, 10, 6)$.

Once again, results with the decomposition-based distance $\|\cdot\|_{D,\theta}$ 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 2.17](#) 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 computationally demanding. Our exploration strategy eludes this problem by only increasing the projection dimension if there is evidence of potential improvement in accuracy.

2.6 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 2.4.2 and 2.5.1, 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.

2.6.1 Experiment setting

The experiment is based on the two case studies previously 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 comparison 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 2.4.2 and 2.5. 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 2.4.2 and 2.5 (with $m^* = 10^{-4}$ and $z = 3$), or until reaching a corner of the solution space.

2.6.2 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:

1. *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:

$$\Delta\tilde{Q}^2 := \frac{\tilde{Q}_{\text{ES}}^2 - \tilde{Q}_{\text{SS}}^2}{\tilde{Q}_{\text{ES}}^2} \times 100\%, \quad (2.27)$$

with $\tilde{Q}_{\text{ES}}^2$ and $\tilde{Q}_{\text{SS}}^2$ denoting the $\tilde{Q}^2$ of the best solution for the ES and the SS method, respectively, computed with (2.21).

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

$$\Delta\text{Time} := \frac{T_{\text{ES}} - T_{\text{SS}}}{T_{\text{ES}}} \times 100\%, \quad (2.28)$$

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 comparison 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.

2.6.3 Analysis

Statistics for the analytic and the coastal flooding case are presented in Tables 2.4 and 2.5, 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 $\|\cdot\|_{D,\theta}$ could be in general a better choice than the scalar distance $\|\cdot\|_{S,\theta}$. The former was the optimal choice in almost all cases except for two instances for the coastal flooding application.

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 2.4 and 2.5 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.

Val Tr	500		1000		1500		2000	
100	B – 3	B – 3	B – 3	B – 3	B – 3	B – 3	B – 3	B – 3
	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$
	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$
	$\Delta\tilde{Q}^2$: 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%	
400	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
	P – 4	P – 4	P – 8	P – 6	P – 4	P – 4	P – 8	P – 4
	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$
	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$
	$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 1.1e–7%		$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 6.4e–8%	
700	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
	P – 5	P – 5	B – 6	B – 6	B – 6	B – 6	B – 6	B – 6
	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$
	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$
1000	$\Delta\tilde{Q}^2$: 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
	B – 6	B – 6	B – 6	B – 6	B – 6	B – 6	B – 6	B – 6
	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$	$\mathcal{M}_{ff}$
1000	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$
	$\Delta\tilde{Q}^2$: 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
	$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%	

Table 2.4 – 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 $\blacksquare$) and (ii) the SS method (lighter colored cell $\square$), as well as the performance statistics computed with (2.27) and (2.28). 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 ($1, \dots, 8$) 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 ($\|\cdot\|_{S,\theta}, \|\cdot\|_{D,\theta}$). 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 $\|\cdot\|_{D,\theta}$. T_{ES} and T_{SS} are provided in minutes (m), hours (h) and days (d).

Val Tr	500		1000		1500		2000	
100	NA	NA	NA	NA	NA	NA	NA	NA
	$\mathcal{M}_{000}$	$\mathcal{M}_{000}$	$\mathcal{M}_{000}$	$\mathcal{M}_{000}$	$\mathcal{M}_{000}$	$\mathcal{M}_{000}$	$\mathcal{M}_{000}$	$\mathcal{M}_{000}$
	NA	NA	NA	NA	NA	NA	NA	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%	
400	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
	B – 4	P – 3	B – 4	P – 3	B – 4	P – 3	B – 4	P – 3
	$\mathcal{M}_{00f}$	$\mathcal{M}_{00f}$	$\mathcal{M}_{00f}$	$\mathcal{M}_{00f}$	$\mathcal{M}_{00f}$	$\mathcal{M}_{00f}$	$\mathcal{M}_{00f}$	$\mathcal{M}_{00f}$
700	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$
	$\Delta\tilde{Q}^2$: 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
1000	P – 1	P – 1	P – 1	P – 1	P – 1	P – 1	P – 1	P – 1
	$\mathcal{M}_{0f0}$	$\mathcal{M}_{0f0}$	$\mathcal{M}_{f0f}$	$\mathcal{M}_{f0f}$	$\mathcal{M}_{f0f}$	$\mathcal{M}_{f0f}$	$\mathcal{M}_{f00}$	$\mathcal{M}_{f00}$
	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$
	$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%		$\Delta\tilde{Q}^2$: 0.0%	
1000	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
	P – 1	P – 1	B – 2	B – 2	P – 1	P – 1	B – 3	B – 3
	$\mathcal{M}_{00f}$	$\mathcal{M}_{00f}$	$\mathcal{M}_{0f0}$	$\mathcal{M}_{0f0}$	$\mathcal{M}_{f0f}$	$\mathcal{M}_{f0f}$	$\mathcal{M}_{00f}$	$\mathcal{M}_{00f}$
1000	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{S,\theta}$	$\ \cdot\ _{S,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{D,\theta}$	$\ \cdot\ _{S,\theta}$	$\ \cdot\ _{S,\theta}$
	$\Delta\tilde{Q}^2$: 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 2.5 – 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 $\blacksquare$) and (ii) the SS method (lighter colored cell $\square$), as well as the performance statistics computed with (2.27) and (2.28). The convention is analogous to that of Table 2.4, except that the projection dimension takes values from $(1, \dots, 4)$. T_{ES} and T_{SS} are provided in minutes (m), hours (h) and days (d).

2.7 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 measure 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 measure similarity among functional input points, the decomposition-based distance $\|\cdot\|_{D,\theta}$ consistently reported better results than the scalar distance $\|\cdot\|_{S,\theta}$ 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 [85] and Ant Colony Optimization [86]. 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 [35].

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 [37]. 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 [87] or try to adapt the work of [88] 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 [Appendix 2.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 enhanced the quality of this work.

Appendix 2.A Dataset constitution

Input	Period	Initial time step	Name	Provider	Source	
					Reference	Website
Tide	1900-2016	No constrained. Set at 10min.	FES2014	LEGOS	[89]	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	[90]	https://reanalyses.org/atmosphere/overview-current-atmospheric-reanalyses#TWENTV2c
	1979-2005	1h	CFSR (sea surface pressure)	NOAA	[91]	https://climatedataguide.ucar.edu/climate-data/climate-forecast-system-reanalysis-cfsr
	2006-2016	15min	MARC	Ifremer and LOPS	[92]	http://marc.ifremer.fr/
	1900-1957	6h	Sonel (waves)	Liens	[93]	http://www.sonel.org/-Waves-.html?lang=en
	1958-09/2002	1h	BoBWA	BRGM	[94]	http://bobwa.brgm.fr/
	10/2002-2007	1h	Homere	Ifremer and LOPS	[95]	http://marc.ifremer.fr/en/produits/rejeu_d_etats_de_mer_homere
	2008-2016	1h	Iowaga/Norgasug	Ifremer and LOPS	[95]	https://wz.ifremer.fr/iowaga/Products

Table 2.6 – Sources of data for the coastal flooding application case.

Appendix 2.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 $\mathbf{t} = \{t_1, \dots, t_L\}$, with $L \in \mathbb{N}$. Let $\mathbf{F}$ 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 (2.10) to discretized functions, and generalizing for the simultaneous projection of multiple curves, we obtain

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

where $\mathbf{B}$ is a $L \times p$ matrix containing p basis functions discretized into the L points in $\mathbf{t}$ 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 $\mathbf{B}$ 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 $\mathbf{A}_{i,\bullet}$ to denote the i -th row of a matrix $\mathbf{A}$ and similarly $\mathbf{A}_{\bullet,j}$ to denote its j -th column. In addition, the orientation of the elements holds so that $\mathbf{A}_{i,\bullet}$ is a row vector whilst $\mathbf{A}_{\bullet,j}$ 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 $\mathbf{W}$ of dimension $L \times L$ indicating the importance of each point in $\mathbf{t}$. Then, the projection coefficients can be found by minimizing the weighted sum of squared residuals. For $j = 1, \dots, n$, the optimization problem can be written:

$$\min_{\boldsymbol{\alpha}_{\bullet,j} \in \mathbb{R}^p} (\mathbf{F}_{\bullet,j} - \mathbf{B}\boldsymbol{\alpha}_{\bullet,j})' \mathbf{W} (\mathbf{F}_{\bullet,j} - \mathbf{B}\boldsymbol{\alpha}_{\bullet,j}). \quad (2.29)$$

The integrated solution by derivatives for the n problems yields:

$$\hat{\boldsymbol{\alpha}} = (\mathbf{B}'\mathbf{W}\mathbf{B})^{-1} \mathbf{B}'\mathbf{W}\mathbf{F}. \quad (2.30)$$

- Ordinary Least Squares (OLS)

If all points in $\mathbf{t}$ are equally important and the information at all points is equally reliable, the matrix $\mathbf{W}$ can be replaced by the identity matrix of dimension $L \times L$ in (2.29) and (2.30) 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 $t_{i*} \in \mathbf{t}$ 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 (2.29), subject to the constraint

$$B_{i*,\bullet}\alpha_{\bullet,j} - F_{i*,j} = 0. \quad (2.31)$$

To solve this problem we use the well known method of Lagrange multipliers [96] which allows to include equality constraints as part of the objective function in order to solve the problem by derivatives. For $j = 1, \dots, n$, the Lagrange function to minimize can be written as

$$\begin{aligned} \mathcal{L}(\alpha_{\bullet,j}, \lambda_j) &= (F_{\bullet,j} - B\alpha_{\bullet,j})' W (F_{\bullet,j} - B\alpha_{\bullet,j}) \\ &\quad + \lambda_j (B_{i*,\bullet}\alpha_{\bullet,j} - F_{i*,j}), \end{aligned} \quad (2.32)$$

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

$$\hat{\alpha} = (B'WB)^{-1} \left(B'WF - \frac{1}{2} B'_{i*,\bullet} \hat{\lambda} \right), \quad (2.33)$$

with

$$\hat{\lambda} = \frac{2 [B_{i*,\bullet} (B'WB)^{-1} B'WF - F_{i*,\bullet}]}{B_{i*,\bullet} (B'WB)^{-1} B'_{i*,\bullet}}. \quad (2.34)$$

- 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 $\mathbf{W}$ in (2.33) and (2.34) by the identity matrix of dimension $L \times L$.

We remark that the closed form solutions provided in (2.30), (2.33) and (2.34) 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 2.C Conditions and results for analytic case

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

Table 2.7 – 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				$\tilde{Q}^2$	CPU time (sec)	
							Train	Pred
38	1	1	B-splines	$\ \cdot\ _{S,\theta}$	4	0.9981	43.2	2.2
39	1	1	PCA	$\ \cdot\ _{S,\theta}$	4	0.9982	54.9	2.3
40	1	1	B-splines	$\ \cdot\ _{D,\theta}$	4	0.9997	36.9	3.3
41	1	1	PCA	$\ \cdot\ _{D,\theta}$	4	0.9997	49.0	3.3
42	1	1	B-splines	$\ \cdot\ _{S,\theta}$	5	0.9959	52.8	2.5
43	1	1	PCA	$\ \cdot\ _{S,\theta}$	5	0.9973	211.9	2.8
44	1	1	B-splines	$\ \cdot\ _{D,\theta}$	5	0.9997	34.2	3.5
45	1	1	PCA	$\ \cdot\ _{D,\theta}$	5	0.9997	48.3	3.5
46	1	1	B-splines	$\ \cdot\ _{S,\theta}$	6	0.9959	62.1	2.6
47	1	1	PCA	$\ \cdot\ _{S,\theta}$	6	0.9960	365.3	2.8
48	1	1	B-splines	$\ \cdot\ _{D,\theta}$	6	0.9997	34.0	3.7
49	1	1	PCA	$\ \cdot\ _{D,\theta}$	6	0.9997	43.0	3.7

Table 2.8 – Analytic case: experimental conditions and results from cleaning and descent stages. The training and prediction times are computed as described in Table 2.7. For the functional input, 1 denotes active and 0 denotes inactive.

Appendix 2.D Conditions and results for coastal flooding case

Conf.	Functional input			Projection method	Covariance function	Projection dimension	Results		
	Td	Sg	Tp				$\tilde{Q}^2$	CPU time (sec)	
								Train	Pred
1	0	0	0	-	-	-	0.7209	27.7	2.0
2	1	0	0	B-splines	$\ \cdot\ _{S,\theta}$	1	0.7214	32.0	2.0
3	0	1	0	B-splines	$\ \cdot\ _{S,\theta}$	1	0.7226	31.2	2.1
4	1	0	0	B-splines	$\ \cdot\ _{S,\theta}$	1	0.7146	36.3	2.1
5	0	0	1	B-splines	$\ \cdot\ _{S,\theta}$	1	0.7211	29.0	2.0
6	1	1	1	B-splines	$\ \cdot\ _{S,\theta}$	1	0.7201	34.8	2.1
7	0	1	1	B-splines	$\ \cdot\ _{S,\theta}$	1	0.7213	33.9	2.1
8	1	1	1	PCA	$\ \cdot\ _{S,\theta}$	1	0.7123	42.6	2.1
9	1	0	0	PCA	$\ \cdot\ _{S,\theta}$	1	0.7198	29.1	2.1
10	0	1	0	PCA	$\ \cdot\ _{S,\theta}$	1	0.7004	29.1	2.1
11	1	1	0	PCA	$\ \cdot\ _{S,\theta}$	1	0.6948	30.6	2.1
12	0	0	1	PCA	$\ \cdot\ _{S,\theta}$	1	0.7160	36.7	2.1
13	1	0	1	PCA	$\ \cdot\ _{S,\theta}$	1	0.7074z	40.2	2.1
14	0	1	1	PCA	$\ \cdot\ _{S,\theta}$	1	0.6958z	39.3	2.1
15	1	1	1	PCA	$\ \cdot\ _{S,\theta}$	1	0.6864z	45.3	2.1
16	1	0	0	B-splines	$\ \cdot\ _{D,\theta}$	1	0.7244	35.8	2.7
17	0	1	0	B-splines	$\ \cdot\ _{D,\theta}$	1	0.7232	34.3	2.6
18	1	1	0	B-splines	$\ \cdot\ _{D,\theta}$	1	0.7247	38.5	2.8
19	0	0	1	B-splines	$\ \cdot\ _{D,\theta}$	1	0.7217	48.8	2.7
20	1	0	1	B-splines	$\ \cdot\ _{D,\theta}$	1	0.7238	61.9	2.8
21	0	1	1	B-splines	$\ \cdot\ _{D,\theta}$	1	0.7230	57.0	2.7
22	1	1	1	B-splines	$\ \cdot\ _{D,\theta}$	1	0.7247	62.5	2.8
23	1	0	0	PCA	$\ \cdot\ _{D,\theta}$	1	0.7234	33.5	2.6

Continued on next page

Table 2.9 – *Continued from previous page*

24	0	1	0	PCA	$\ \cdot\ _{D,\theta}$	1	0.7210	33.4	2.6
25	1	1	0	PCA	$\ \cdot\ _{D,\theta}$	1	0.7218	37.7	2.7
26	0	0	1	PCA	$\ \cdot\ _{D,\theta}$	1	0.7199	52.0	2.6
27	1	0	1	PCA	$\ \cdot\ _{D,\theta}$	1	0.7216	62.4	2.7
28	0	1	1	PCA	$\ \cdot\ _{D,\theta}$	1	0.7192	59.1	2.7
29	1	1	1	PCA	$\ \cdot\ _{D,\theta}$	1	0.7204	64.9	2.8
30	1	0	0	B-splines	$\ \cdot\ _{S,\theta}$	2	0.6967	32.0	2.3
31	0	1	0	B-splines	$\ \cdot\ _{S,\theta}$	2	0.6915	33.2	2.3
32	1	0	0	B-splines	$\ \cdot\ _{S,\theta}$	2	0.6592	42.3	2.4
33	0	0	1	B-splines	$\ \cdot\ _{S,\theta}$	2	0.7137	37.8	2.3
34	1	1	1	B-splines	$\ \cdot\ _{S,\theta}$	2	0.6866	48.1	2.4
35	0	1	1	B-splines	$\ \cdot\ _{S,\theta}$	2	0.6869	48.3	2.4
36	1	1	1	PCA	$\ \cdot\ _{S,\theta}$	2	0.6507	67.0	2.6
37	1	0	0	PCA	$\ \cdot\ _{S,\theta}$	2	0.6973	31.4	2.3
38	0	1	0	PCA	$\ \cdot\ _{S,\theta}$	2	0.7048	34.0	2.3
39	1	1	0	PCA	$\ \cdot\ _{S,\theta}$	2	0.6653	44.4	2.4
40	0	0	1	PCA	$\ \cdot\ _{S,\theta}$	2	0.7145	42.4	2.3
41	1	0	1	PCA	$\ \cdot\ _{S,\theta}$	2	0.6875	60.5	2.5
42	0	1	1	PCA	$\ \cdot\ _{S,\theta}$	2	0.6932	61.3	2.5
43	1	1	1	PCA	$\ \cdot\ _{S,\theta}$	2	0.6532	95.1	2.6
44	1	0	0	B-splines	$\ \cdot\ _{D,\theta}$	2	0.7215	33.9	2.7
45	0	1	0	B-splines	$\ \cdot\ _{D,\theta}$	2	0.7211	35.7	2.7
46	1	1	0	B-splines	$\ \cdot\ _{D,\theta}$	2	0.7195	38.4	2.8
47	0	0	1	B-splines	$\ \cdot\ _{D,\theta}$	2	0.7199	55.0	2.8
48	1	0	1	B-splines	$\ \cdot\ _{D,\theta}$	2	0.7203	60.8	2.9
49	0	1	1	B-splines	$\ \cdot\ _{D,\theta}$	2	0.7190	61.2	2.9
50	1	1	1	B-splines	$\ \cdot\ _{D,\theta}$	2	0.7187	65.8	3.0
51	1	0	0	PCA	$\ \cdot\ _{D,\theta}$	2	0.7226	33.2	2.7
52	0	1	0	PCA	$\ \cdot\ _{D,\theta}$	2	0.7211	37.2	2.7
53	1	1	0	PCA	$\ \cdot\ _{D,\theta}$	2	0.7197	35.5	2.8
54	0	0	1	PCA	$\ \cdot\ _{D,\theta}$	2	0.7199	52.0	2.7
55	1	0	1	PCA	$\ \cdot\ _{D,\theta}$	2	0.7212	58.7	2.9
56	0	1	1	PCA	$\ \cdot\ _{D,\theta}$	2	0.7196	59.5	2.9
57	1	1	1	PCA	$\ \cdot\ _{D,\theta}$	2	0.7197	63.0	3.0
58	1	0	0	B-splines	$\ \cdot\ _{S,\theta}$	3	0.7081	36.3	2.3
59	0	1	0	B-splines	$\ \cdot\ _{S,\theta}$	3	0.7158	35.9	3.4
60	1	0	0	B-splines	$\ \cdot\ _{S,\theta}$	3	0.6790	50.7	2.5
61	0	0	1	B-splines	$\ \cdot\ _{S,\theta}$	3	0.7121	53.8	2.4
62	1	1	1	B-splines	$\ \cdot\ _{S,\theta}$	3	0.6969	84.2	2.6
63	0	1	1	B-splines	$\ \cdot\ _{S,\theta}$	3	0.6994	81.1	2.6
64	1	1	1	PCA	$\ \cdot\ _{S,\theta}$	3	0.6699	130.6	2.7
65	1	0	0	PCA	$\ \cdot\ _{S,\theta}$	3	0.6737	38.0	2.4
66	0	1	0	PCA	$\ \cdot\ _{S,\theta}$	3	0.7014	37.0	2.4
67	1	1	0	PCA	$\ \cdot\ _{S,\theta}$	3	0.6428	55.8	2.6
68	0	0	1	PCA	$\ \cdot\ _{S,\theta}$	3	0.7117	56.5	2.4
69	1	0	1	PCA	$\ \cdot\ _{S,\theta}$	3	0.6665	122.9	2.6
70	0	1	1	PCA	$\ \cdot\ _{S,\theta}$	3	0.6956	94.4	2.6
71	1	1	1	PCA	$\ \cdot\ _{S,\theta}$	3	0.6371	182.7	2.7
72	1	0	0	B-splines	$\ \cdot\ _{D,\theta}$	3	0.7225	33.9	2.7
73	0	1	0	B-splines	$\ \cdot\ _{D,\theta}$	3	0.7202	36.1	2.8
74	1	1	0	B-splines	$\ \cdot\ _{D,\theta}$	3	0.7211	37.5	2.9
75	0	0	1	B-splines	$\ \cdot\ _{D,\theta}$	3	0.7205	53.3	2.8
76	1	0	1	B-splines	$\ \cdot\ _{D,\theta}$	3	0.7216	67.7	3.0
77	0	1	1	B-splines	$\ \cdot\ _{D,\theta}$	3	0.7194	60.7	3.0
78	1	1	1	B-splines	$\ \cdot\ _{D,\theta}$	3	0.7206	65.8	3.2
79	1	0	0	PCA	$\ \cdot\ _{D,\theta}$	3	0.7231	31.5	2.7

Continued on next page

Table 2.9 – Continued from previous page

80	0	1	0	PCA	$\ \cdot\ _{D,\theta}$	3	0.7219	36.9	2.7
81	1	1	0	PCA	$\ \cdot\ _{D,\theta}$	3	0.7225	35.7	2.9
82	0	0	1	PCA	$\ \cdot\ _{D,\theta}$	3	0.7203	52.4	2.8
83	1	0	1	PCA	$\ \cdot\ _{D,\theta}$	3	0.7218	58.0	3.0
84	0	1	1	PCA	$\ \cdot\ _{D,\theta}$	3	0.7212	60.0	3.0
85	1	1	1	PCA	$\ \cdot\ _{D,\theta}$	3	0.7212	63.3	3.2

Table 2.9 – Coastal flooding case: experimental conditions and results from screening stage. The training and prediction times are computed as described in Table 2.7. For the functional input, 1 denotes active and 0 denotes inactive.

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

Table 2.10 – Coastal flooding case: evaluation of projection dimension selected based on projection error. The training and prediction times are computed as described in Table 2.7.

Chapter 3

Ant Colony based model selection for functional-input Gaussian process regression

The chapter in brief

[Chapter 2](#) presented our first prototype of methodology for the calibration of structural parameters of the metamodel. We used that methodology to tune the metamodel for a simplified version of our hydrodynamic code, which received only part of the variables used by our target code. Although the proposed methodology proved to be effective for such application, we concluded the need for more sophisticated exploration techniques in order to scale to bigger problems involving (i) a larger number of inputs, (ii) more structural parameters and (iii) more levels per structural parameter. In [Chapter 3](#) we address this need by introducing an Ant Colony based smart exploration algorithm.

Ant colony optimization (ACO) encompasses a large variety of optimization metaheuristics derived from the seminal work of Dorigo et al. in the early 90s [9, 10]. Since then, ACO based heuristics have been proved to give remarkable results in a wide range of optimization problems, including DNA sequencing [11], scheduling [12], protein-ligand docking [13], assembly line balancing [14] and packet-switched routing [15]. ACO has been recognized as one of the most successful research lines in the area of swarm intelligence [16, 17], and always seats beside evolutionary algorithms, iterated local search, simulated annealing, and tabu search among the top metaheuristic techniques [18]. This chapter is an exact copy of our technical report [19], which recalls the foundations of Ant Colony Optimization and elaborates on top of them to make the algorithm suitable for our structural optimization problem. To the best of our knowledge, this is the first algorithm addressing the structural optimization problem for Gaussian process models.

We use the proposed algorithm to calibrate the metamodel for three analytic black-box functions. The models obtained were in all cases of outstanding prediction quality. In [Chapter 5](#) we use the algorithm to calibrate the metamodel for the full hydrodynamic code of the RISCOPE application.

Contents

3.1	Introduction	57
3.2	The ant colony system	57
3.2.1	Biological inspiration	57
3.2.2	The optimization algorithm	58
3.2.3	Adaption to model selection	59
3.3	Analytic test cases	64
3.3.1	Black-box functions	64
3.3.2	Data generation and heuristic setup	65
3.3.3	Results	66
3.4	Conclusions	68
3.5	Aknowledgments	68



Risk-Based System For Coastal Flooding Early Warning
RISCOPE.fr 

Ant Colony Based Model Selection for Functional-Input Gaussian Process Regression

Technical Report - April 2020

Deliverable: D3.b

Reference: WP3.2



José Betancourt, François Bachoc, Thierry Klein, Fabrice Gamboa

Financed by



Certified by





RISCOPE research project ([RISCOPE.fr](https://riscope.fr) ) is funded by the French Agence Nationale de la Recherche (ANR) for the period 2017-2021 (ANR, project No. 16CE04-0011). This publication reflects the views only of the author's, and the ANR cannot be considered liable for any use that may be made of the information contained therein.

Recommended citation

Betancourt, J., Bachoc, F., Klein, T., and Gamboa, F. (2020). Technical Report: Ant Colony Based Model Selection for Functional-Input Gaussian Process Regression. Ref. D3.b (WP3.2), *RISCOPE project*.

3.1 Introduction

Gaussian process models (a.k.a. Kriging models) are one of the preferred choices for meta-modeling nowadays [75], competing with a few others like polynomials, splines, generalized linear models and neural networks [97, 98]. While sometimes similar in prediction accuracy to the other cited methods, Gaussian processes present among other advantages: (i) the capability to reproduce complex (and a priori unknown) nonlinear input-output relationships, (ii) the ability to interpolate the observations, and (iii) the interpretability of predictions which include an estimation of the uncertainty at each prediction point.

Gaussian process metamodels were originally developed for scalar inputs, but are now also available for functional inputs. The extension to functional inputs gives rise to a number of questions about the proper way to represent them in the metamodel: (i) which functional inputs are worth keeping as predictors, (ii) which dimension reduction method (DR) is ideal to use (e.g., B-splines, PCA, PLS), (iii) which is a suitable projection dimension, and given our choice to work with Gaussian process metamodels, also the question of (iv) which is a convenient distance to measure similarities between functional input points within the kernel function. Some of these characteristics - hereon called structural parameters - of the model and some others such as the family of kernel (e.g., Gaussian, Matérn 5/2) are often chosen a priori, either based on the familiarity of the modeler with certain methods or in the results of other metamodeling experiences. As one may intuit and has been shown by us through experiments in [7], the configuration of the structural parameters of the model has a strong impact on its prediction capability. In this report, we introduce an heuristic optimization method for the selection of a convenient combination of structural parameters in the context of functional-input Gaussian process models. The architecture of the algorithm was made custom to the aforementioned problem, however, its principles are general enough and the method can be easily extended to other model selection frameworks.

3.2 The ant colony system

Ant colony optimization (ACO) encompasses a large variety of optimization metaheuristics derived from the seminal work of Dorigo et al. in the early 90s [9, 10]. Since then, ACO based heuristics have been proved to give remarkable results in a wide range of optimization problems, including DNA sequencing [11], scheduling [12], protein-ligand docking [13], assembly line balancing [14] and packet-switched routing [15]. ACO has been recognized as one of the most successful research lines in the area of swarm intelligence [16, 17], and always seats beside evolutionary algorithms, iterated local search, simulated annealing, and tabu search among the top metaheuristic techniques [18].

3.2.1 Biological inspiration

ACO algorithms work based on *stigmergy*, a mechanism for indirect inter-agent communication through traces left in the environment. Ants employ this type of communication to find efficient routes during foraging¹. The way it works is traditionally explained through a picture similar to the one displayed in Figure 3.1. The frame sequence in alphabetical order illustrates the variant of the *double bridge* experiment performed by Goss et al. in the 80s [99]. In the experiment, the nest of a colony of ants was connected to a food source by two

¹Foraging: searching for wild food resources.

paths, one significantly longer than the other (frame A). At first the ants began to explore the environment by randomly distributing themselves in the two paths (frame B). Along its way, each ant left pheromone trails noticeable by its mates. As expected, the ants that took the shortest path met the food before (frame C). Most part of the ants that started first the way back to the nest, perceived the larger load of pheromones in the shortest path and went through it. The shortest path kept receiving pheromones at a incrementally higher rate than the longer one, gradually reducing the chances of an ant taking this last (frame D). After some time, the whole colony converged towards the use of the shortest path (frame E).

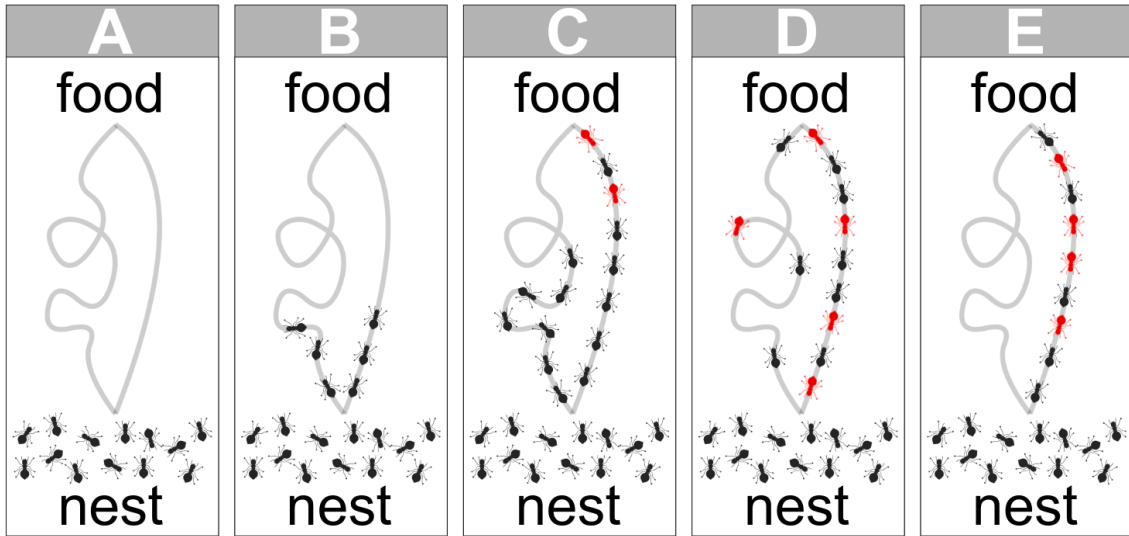


Figure 3.1 – Stigmergy used by ants to find efficient paths towards a food source. Red ants represent the ones going back from the food source to the nest.

3.2.2 The optimization algorithm

In ACO algorithms, a colony of artificial ants evaluate solutions to the optimization problem at hand. The quality of those solutions is informed to the colony through virtual pheromone trails which help the algorithm to converge towards a high quality solution. To this day there is more than a dozen ACO metaheuristics and probably hundreds of ACO based heuristics². The algorithm proposed here is inspired in the ant colony system (ACS), introduced by Dorigo and Gambardella in the late 90s [100]. In this report we proceed directly with the explanation of our heuristic for model selection. The aforementioned reference is recommended to the reader interested in the original version of ACS.

The ACS operates over a decision network that must be defined in advance based on the structure of the solution space. To start, each ant is located on a base spot and the pheromone value of each link is initialized. Each ant generates a solution to the optimization problem by adding nodes of the decision network to its path according to a pseudo-random system of rules biased by the pheromone loads in the network. Each time an ant traverses a link, a local pheromone update takes place; the pheromone load of the link is slightly reduced in order to foment the diversification of solutions (principle of *exploration*). Once all ants have made a complete solution, the quality of each solution is evaluated and a global pheromone

²Metaheuristic vs. heuristic: a metaheuristic is a generic solution technique that can be applied to a broad set of problems; an heuristic is a solution technique designed to resolve a particular problem. An heuristic might be an adaption of an metaheuristic to the particularities of the optimization task at hand.

update occurs; the pheromone load of the links traversed by the best ants is increased in proportion to the quality of their corresponding solutions, striving for convergence towards high quality solutions (principle of *exploitation*). The process is then iterated until some stopping conditions are reached. A pseudocode for the ACS is presented in [Algorithm 1](#).

Algorithm 1 Generic ACS structure

```

1: while <stopping conditions remain unsatisfied> do
2:   create a new population of ants
3:   for <i=1:Psize> do
4:     locate ant i at its base spot
5:     tag ant i as partial
6:   end for
7:   while <there are still partial ants> do
8:     randomly pick a partial ant
9:     apply transition rule to select next node in its sequence
10:    reduce pheromone load of chosen link
11:   end while
12:   evaluate the solution made by each ant
13:   increase pheromone load of links in best solutions
14:    $P^* \leftarrow$  best solution so far
15: end while
16: return  $P^*$ 

```

3.2.3 Adaption to model selection

Decision network

Considering the framework described in [Section 3.1](#) for a set of ds scalar inputs and df functional inputs our optimization problem consists on making the following decisions:

- State of the i -th scalar input, from {inactive, active};
- State of the j -th functional input, from {inactive, active};
- Projection basis for the j -th functional input, from $\{B_1, \dots, B_z\}$;
- Projection dimension for the j -th functional input, from $\{0, \dots, k_j\}$;
- Distance for the j -th functional input, from $\{D_1, \dots, D_w\}$;
- Kernel type, from $\{K_1, \dots, K_x\}$,

with $i \in \{1, \dots, ds\}$, $j \in \{1, \dots, df\}$ and k_j the original dimension of input j . The sets $\{B_1, \dots, B_z\}$, $\{D_1, \dots, D_w\}$ and $\{K_1, \dots, K_x\}$ correspond to the basis, distance and kernel families to be considered, in that order. The projection dimension 0 denotes no projection. In order to find a suitable combination of the parameters listed above, we let our artificial ants to move through a network with a structure similar to the one depicted in [Figure 3.2](#). Such a structure prevents the constitution of senseless solutions (e.g., an input being both, inactive and active) and helps to keep the network data structures considerably simple by only defining strictly necessary links.

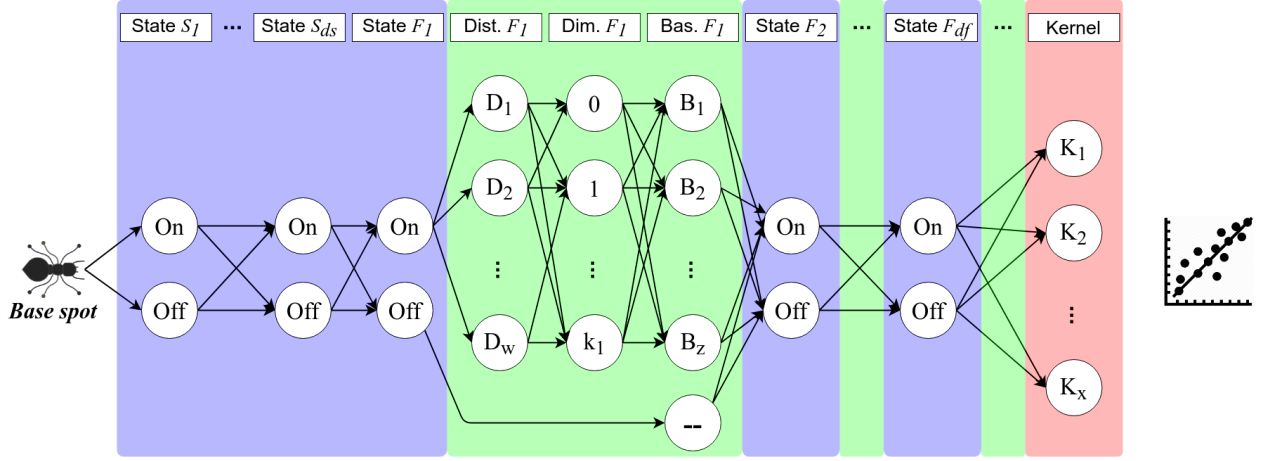


Figure 3.2 – Prototype of the network used in our ACO heuristic.

Transition rules

ACO is an iterative algorithm. At each iteration, a group of artificial ants are located at a base spot. Each ant builds a feasible model structure by walking from node to node, always respecting the direction of the links. At each step, an ant selects the next node based on a pseudo-stochastic mechanism defined by (3.1):

$$\text{rule} = \begin{cases} \text{Rule 1} & \text{if } q \leq q_0, \\ \text{Rule 2} & \text{otherwise,} \end{cases} \quad (3.1)$$

with q_0 a parameter $\in [0, 1]$ and q a random value from $U(0, 1)$. For ant r located at node a , our transition rules are defined as follows:

Rule 1. Move to the feasible neighbor node with greatest pheromone load. Mathematically, it is to move to the node s specified by

$$s = \underset{b \in J_r(a)}{\operatorname{argmax}} \tau_{ab}, \quad (3.2)$$

with $J_r(a)$ the set of feasible neighbor nodes for ant r located at node a , and τ_{ab} the pheromone load of link (a, b) .

Rule 2. Pick the next node based on a probability distribution proportional to the pheromone load of the feasible neighbor nodes. Formally, this can be expressed as moving to node b with probability

$$P(b) = \frac{\tau_{ab}}{\sum_{b \in J_r(a)} \tau_{ab}}, \quad (3.3)$$

with $J_r(a)$ and τ_{ab} interpreted as in rule 1.

Note that the proposed algorithm does not make use of the heuristic visibility value considered in [100] and often present in ACO algorithms. The role of this value, typically

denoted η_{ab} for the link (a, b) , is to introduce a priori information about the potential benefit of including each link in the solution. For many optimization problems like routing-oriented and scheduling-oriented ones, the visibility of a link is naturally set to be a function of its inverse generalized cost (see e.g., [101] and [102]). For optimization problems involving categorical variables (such as model selection problem at hand), this set up is often less intuitive since the order of preferences over different levels of the same factor and the relative degree of preference of each level are hard to estimate. For instance, consider the questions of: (i) what would be the order of preference over a set of 5 types of basis families, and (ii) how preferred each basis family should be. Since there is no published evidence of the superiority of certain basis family over the others irrespective of the regression problem at hand, it could be hard to answer these two questions. In order to include η_{ab} in the algorithm, the same two questions would have to be answered for the distance and the kernel family, and every other feature of the regression model. Hence, we decided to remove the visibility from our model, while keeping in mind that a priori information can also be introduced in the algorithm through the initial pheromone load. This possibility will be discussed later, at the end of this section.

Pheromone update

As explained in Section 3.2.2, ACS implements two pheromone update mechanisms – local and global – responsible for the diversification of solutions and the exploitation of acquired knowledge about the structure of high quality solutions. The local pheromone update is triggered each time an ant adds a note to its sequence. The pheromone load of the traverse link is slightly reduced and as a consequence, other ants are less motivated to use the same link in further decisions. In the proposed algorithm, the local pheromone update operates on the link (a, b) based on the assignment

$$\tau_{ab} \leftarrow (1 - \rho_l) \cdot \tau_{ab} + \rho_l \cdot \tau_0, \quad (3.4)$$

where τ_{ab} is the current pheromone load of the link, τ_0 is its initial pheromone load, and $\rho_l \in [0, 1]$ is a parameter that can be interpreted as the pheromone evaporation rate.

On the other hand, global pheromone update takes place each time that a colony becomes complete, i.e., each time all ants in a colony complete a solution. This time, the pheromone load of links belonging to the best ants is increased in proportion to the quality of the corresponding solutions. For each high quality ant, the global pheromone operates on the link (a, b) based on the assignment

$$\tau_{ab} \leftarrow (1 - \rho_g) \cdot \tau_{ab} + \rho_g \cdot \psi, \quad (3.5)$$

where τ_{ab} is interpreted as in (3.4), ψ is a measure of the quality of the solution, and $\rho_g \in [0, 1]$ is a parameter that can be interpreted as the learning reinforcement rate. If multiple ants are used in the global update, (3.4) is applied in an iterated manner over the set of best ants.

Initial pheromone load

The ACS initiates with a base pheromone load on every link. This quantity is modified by the virtual ants during the optimization to communicate actions and learning with their peers. The initial pheromone load must be set with caution, since this value will be determinant on the proper functioning of the algorithm;

- If it is set too low, the heuristic will be prematurely and irreversibly biased towards the best solution of the first iteration, breaking down the learning capability of the system. In addition, the first iteration does not count with any learned information. Thus, the solution at which the system would get stuck might be of regular quality.
- If it is conversely set too high, the system will struggle to converge (if it manages to do so). High quality solutions will not drag the attention they deserve and ants will not be able to focus their exploration around them.

In the proposed algorithm, pheromones are implicitly configured to take values exclusively in $[0, 1]$. Thanks to the structure of the updating queries (3.4) and (3.5), this is easily achieved by just setting the initial pheromone load of every link in the range $[0, 1]$ and using a quality measure ψ for the solutions in that same range. For the model selection problem, one can pick for instance the Leave-one-out (LOO) cross-validated squared correlation coefficient Q_{loocv}^2 (3.6), or alternatively its hold-out analogous, the predictive squared correlation coefficient Q_{hout}^2 (see e.g., [77]).

$$Q_{loocv}^2 := \left[1 - \frac{\sum_{i=1}^{n.tr} (y_i - \hat{y}_{i,-i})^2}{\sum_{i=1}^{n.tr} (y_i - \bar{y})^2} \right]_0, \quad (3.6)$$

with $(y_i)_{i=1,\dots,n.tr}$ the vector of observed output values, $\bar{y}$ the average of that vector, $\hat{y}_{i,-i}$ the LOO estimation of y_i , and the operator $\lfloor \cdot \rfloor_l$ defined as:

$$\lfloor x \rfloor_l = \begin{cases} x & \text{if } x \geq l, \\ l & \text{otherwise.} \end{cases}$$

Several numerical trials allowed us to identify an initial pheromone load of 0.1, combined with populations of size 10, as a suitable configuration taking into account the setup described in the previous paragraphs. During those trials, we observed the affectation of the pheromone level and the overall behavior of the algorithm along the iterations. In addition, we were able to corroborate the drawbacks associated to excessively low or high τ_0 values, explained in the introductory paragraph of this subsection.

A special treatment for the assignment of the initial pheromone load was given to the links connecting a distance type with a projection dimension (see Figure 3.2). The framework presented here is general enough to account for any number and type of distance families, however, for practical purposes we adopted the norms $\|\cdot\|_{D,\theta_f}$ and $\|\cdot\|_{S,\hat{\theta}_f}$ defined in [7] as a baseline. The selection of one of these two norms not only might have a relevant impact on the predictability of the model, but also on its tractability. The norm $\|\cdot\|_{D,\theta_f}$ requires a single length-scale parameter per functional input, indifferently of its projection dimension. In contrast, for each functional input, the norm $\|\cdot\|_{S,\hat{\theta}_f}$ requires as many length-scale coefficients as projection terms. That means that the optimization of the hyperparameters of the model will almost always involve a larger number of decisions variables when using the norm $\|\cdot\|_{S,\hat{\theta}_f}$. This norm must be considered as an option as it could be the optimal choice in terms of predictability. However, if the projection dimension is allowed to take too high values, the norm $\|\cdot\|_{S,\hat{\theta}_f}$ may imply substantially harder and more time consuming hyperparameters

learning sessions than the norm $\|\cdot\|_{D,\theta_f}$. As a mechanism of regularization, we set up the initial pheromone load for the links pointing out to the norm $\|\cdot\|_{S,\dot{\theta}_f}$, based on a loss function of the form

$$\tau_f(x; k_j, \tau_0, \delta_j, w_j) = \begin{cases} \tau_0 \exp\left(-\frac{|k_j - 1| - \delta_j}{2\sigma^2}\right) & \text{if } x = 0, \\ \tau_0 & \text{if } 0 < x \leq \delta_j, \\ \tau_0 \exp\left(-\frac{|x - 1| - \delta_j}{2\sigma^2}\right) & \text{otherwise,} \end{cases} \quad (3.7)$$

where k_j is the original dimension of input j , x takes integer values corresponding to its possible projection dimensions, τ_0 denotes the general initial pheromone load of the heuristic, and

$$\sigma^2 = -\frac{w_j^2}{2 \log(.5)}.$$

The parameters δ_j and w_j draw the shape of the loss function by specifying the extension of its flat section and the smoothness of its decreasing section (see [Figure 3.3](#)).

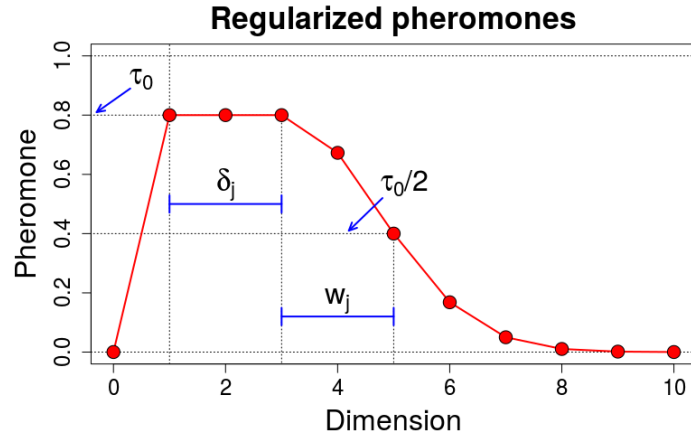
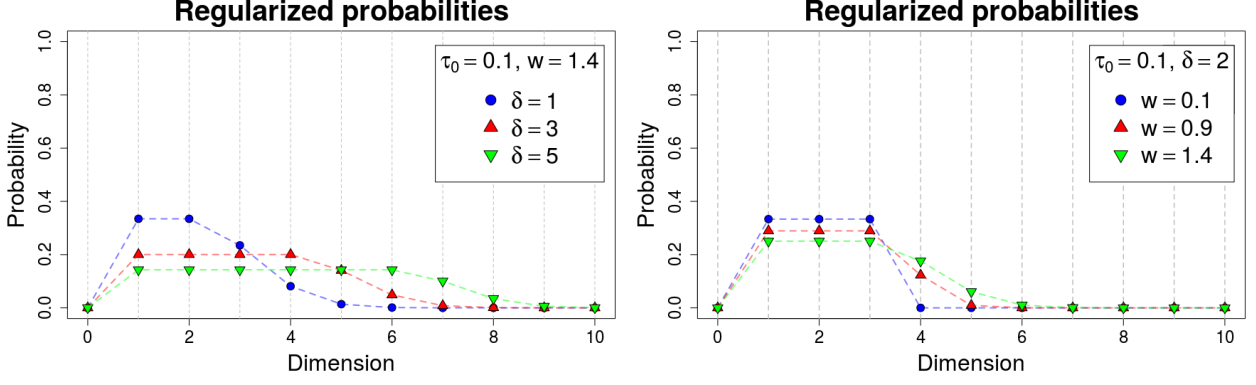


Figure 3.3 – Loss function used for regularization in dimension reduction for functional inputs using the norm $\|\cdot\|_{S,\dot{\theta}_f}$. Built for an hypothetical functional input of dimension 10.

The first condition in (3.7) controls the case where the projection dimension is equal to zero, which as stated earlier in the document denotes no projection. For every functional input, the values of δ_j and w_j can be tuned by observing the normalized loss curve with sum equal to 1. This curve matches the values of the probability pie defined in transition rule 2 (see (3.3)), and will be used by the ants during the first iteration of the algorithm, each time that this rule is implemented. Some examples of the normalized loss function are provided in [Figure 3.4](#), where we illustrate the effect of w and δ on the shape of the curve. Suitable w and δ values will vary depending on particularities of the regression task. For instance, if there are many candidate functional inputs, both values could be set relatively low in order to prevent the heuristic from building too heavy models.

As a closing remark, the initial pheromone value can be used to induce preferences on the behavior of the ants. Whenever there is some hint that one level of some feature will perform



(a) Effect of δ on the normalized loss function. (b) Effect of w on the normalized loss function.

Figure 3.4 – Normalized loss function for hypothetical input of dimension 10.

better than the others, this information can be directly placed in the initial pheromone loads of that feature. By doing so, the ants will be stimulated to test more often configurations including the expected best performing level of the feature. The advantage of specifying this information through the pheromones and not through a visibility value (see the discussion on the transition rules) is that in case the induced bias was erroneous, the ants will be able to systematically remove it through the local pheromone update. Conversely, the ants will be able to reinforce the bias through the global pheromone update if they find it fruitful.

3.3 Analytic test cases

Let us now check the performance of the heuristic. We set ourselves in a metamodeling framework where an expensive-to-evaluate computer code is to be substituted by a light-to-run statistical model (see e.g., [103] or [7] for more details on the metamodeling problem). We consider three analytic black-box functions and we undertake the model selection problem for each of them. We proceed below with the definition of our black-box functions.

3.3.1 Black-box functions

Let $\mathcal{F}$ be the set of continuous functions from $[0, 1]$ to $\mathbb{R}$. Let $\mathbf{x} = (x_1, x_2) \in [0, 1]^2$ and $\mathbf{f} = (f_1, f_2) \in \mathcal{F}^2$ be the vectors of scalar and continuous functional inputs in that order. Then, consider the black-box computer codes specified by the following analytic functions:

- **Analytic black-box 1**

$$\mathcal{G}_1 : [0, 1]^2 \times \mathcal{F}^2 \rightarrow \mathbb{R},$$

$$(\mathbf{x}, \mathbf{f}) \mapsto x_1 \sin(x_2) + x_1 \int_0^1 f_1 dt_1 - x_2^2 \left((\max_{T_2} f_2) - (\min_{T_2} f_2) \right). \quad (3.8)$$

- **Analytic black-box 2**

$$\mathcal{G}_2 : [0, 1]^2 \times \mathcal{F}^2 \rightarrow \mathbb{R},$$

$$(\mathbf{x}, \mathbf{f}) \mapsto x_1 \sin(x_2) + \int_0^1 \exp(x_1 t_1) f_1 dt_1 - x_2^2 \int_0^1 f_2 t_2 dt_1. \quad (3.9)$$

- **Analytic black-box 3**

$$\mathcal{G}_3 : [0, 1]^2 \times \mathcal{F}^2 \rightarrow \mathbb{R},$$

$$(\mathbf{x}, \mathbf{f}) \mapsto x_1 - 2x_2 + 4 \int_0^1 t_1 f_1 dt_1 + \int_0^1 f_2 dt_2. \quad (3.10)$$

Note that the three functions above are totally independent and the only purpose of resolving the model selection problem for all of them is to check the robustness of our proposed algorithm to variations on the structure of the underlying input-output true model.

3.3.2 Data generation and heuristic setup

Here, we focus on the solution of the model selection problem and we keep the generation of synthetic input data (the experimental design) very simple. For all the three analytic functions we use the same input and output data. We generate the scalar part of the design from a grid over $[0, 1]^2$. We assume that the functional inputs f_1 and f_2 are represented by vectors of size 10 and 22, respectively. We made this choice in order to include functional inputs with heterogeneous discretization in the experiment. We sampled all the values of each function randomly from $U(0, 1)$. In total, we generated an arbitrary number of 100 input points. For each point, we computed the corresponding output values using (3.8), (3.9) and (3.10).

As mentioned earlier in the discussion about the initial pheromone load (Section 3.2.3), a series of numerical trials allowed us to identify the combination $\tau_0 = 0.1$ with a population of size 10 to work properly for our application. During these tests, we also allowed the decision probability boundary q_0 , the evaporation rate ρ_l and the learning reinforcement rate ρ_g vary. More specifically, we tried values in the vicinity of $(0.9, 0.1, 0.1)$, the values used for those parameters (in the corresponding order) by Dorigo et al. in [100]. We found the triple $(0.95, 0.1, 0.1)$ to offer a good trade-off between solution discovery and convergence. Finally, we fixed the regularization parameters δ and w at 2 and 1.4, respectively for both functional inputs. This setup produces the normalized loss functions displayed in Figure 3.5.

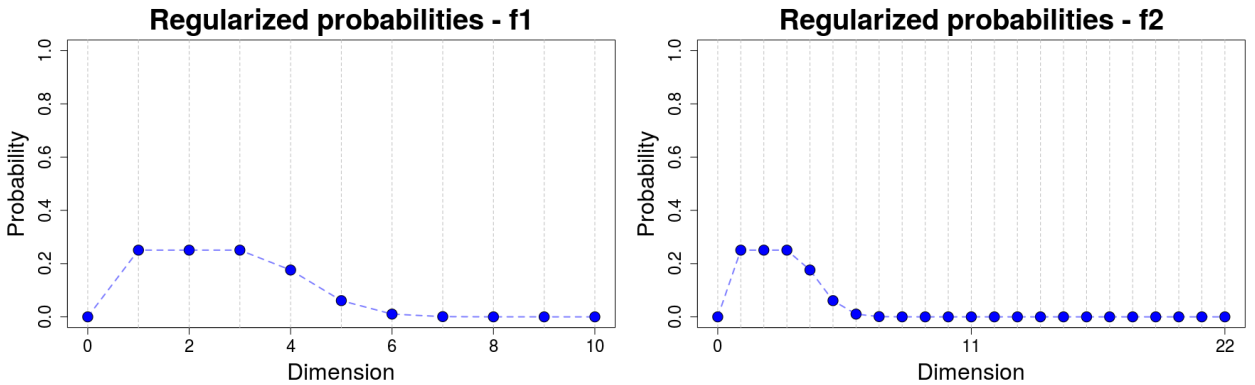


Figure 3.5 – Normalized loss functions used for the analytic cases.

3.3.3 Results

In this section we check the performance of our algorithm from three different perspectives:

- 1) the absolute quality of the selected model;
- 2) the relative quality of the selected model;
- 3) the evolution of the solutions found throughout the iterations.

Each of them is better described in the corresponding subsection. An ideal general number of iterations that will work well for any model selection problem may likely not exist. Since our purpose is merely to show the performance of the heuristic, we used an arbitrary number of 20 iterations. For actual applications where the interest is to find the best possible solution the algorithm can give, we recommend instead using a stopping condition based on processing time. This way, we will have the best possible solution given our time constraints.

Absolute model quality

The absolute quality of the model refers to its predictability, regardless of the quality of the other explored models. This dimension of quality can be assessed, for instance, by means of the Q^2_{loocv} or Q^2_{hout} statistics (see the discussion about the initial pheromone load in [Section 3.2.3](#)). Those are conveniently interpretable measures thanks to the fact that they get values in $[0, 1]$ (for any worthwhile model), with a value of 1 indicating perfect fitting and 0 indicating a very poor one. Here we optimize the model structure in terms of the Q^2_{loocv} . [Figure 3.6](#) displays the calibration plot and the Q^2_{loocv} for the model selected for each of the black-box functions.

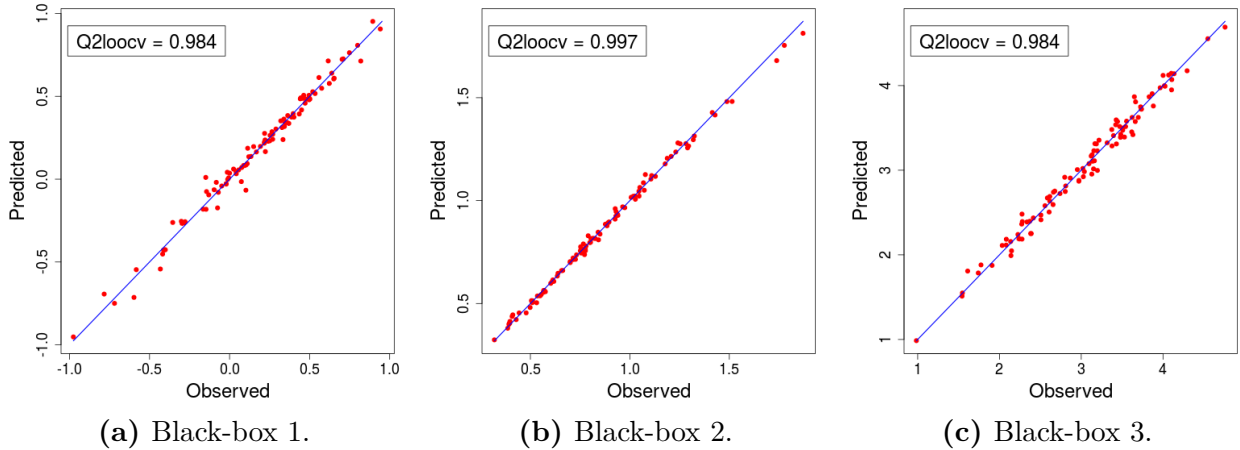


Figure 3.6 – Calibration plot for selected models.

Based on the calibration plots, all the models are good in terms of accuracy and precision. No evident fitting problems (e.g., skewness, heavy tails) are present in any of the models.

Relative model quality

The relative quality of the selected model is assessed by comparing it to the other explored models. This dimension of quality allows to see how special our model is. For instance, a Q^2_{loocv} of 0.95 is not that especial if the vast majority of models had a metrics over 0.92, and similarly, a Q^2_{loocv} of 0.58 is not that bad if most of the models had metrics below 0.35. In

Figure 3.7 we display the Q_{loocv}^2 of all the models explored during the optimization for each of the black-box functions.

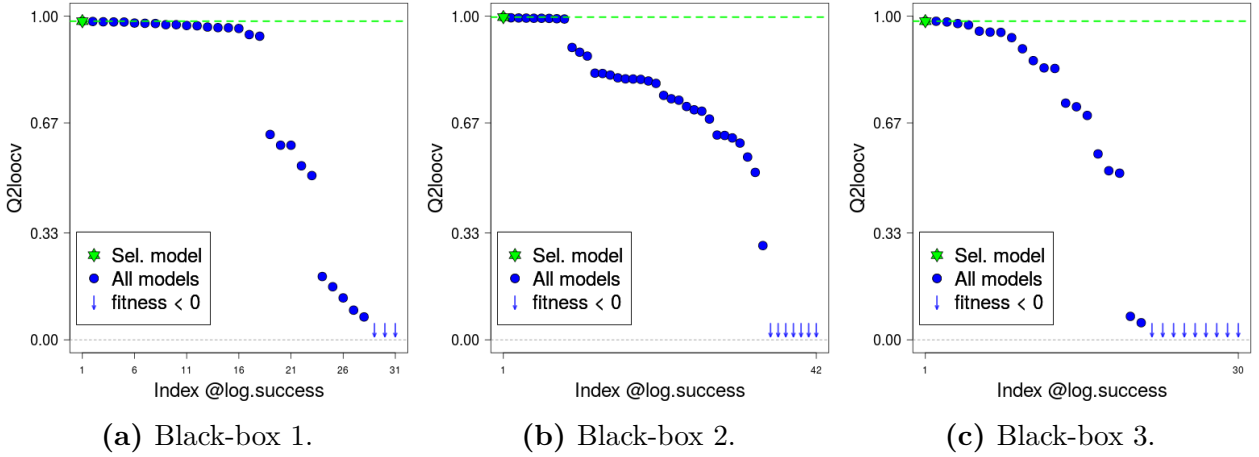


Figure 3.7 – Relative quality of selected models.

In the three cases an important portion of the explored models reported a Q_{loocv}^2 that fell well below that of the selected model. Two important takeaways from these plots are:

- the arbitrary selection of the structural parameters is an unsafe move that may be the difference between a poorly performing model and a model of high prediction quality;
- even if there are multiple relatively good models (as for black-boxes 1 and 2), the proposed heuristic will go one step forward and deliver simply the best one.

Evolution

Finally, the analysis of the evolution of the solutions found by the algorithm along the iterations helps to verify that its learning mechanism is working properly. In Figure 3.8 we plot the Q_{loocv}^2 of the models explored at each iteration of the heuristic during the optimization for each black-box function, along with the per-iteration maximum and median Q_{loocv}^2 values.

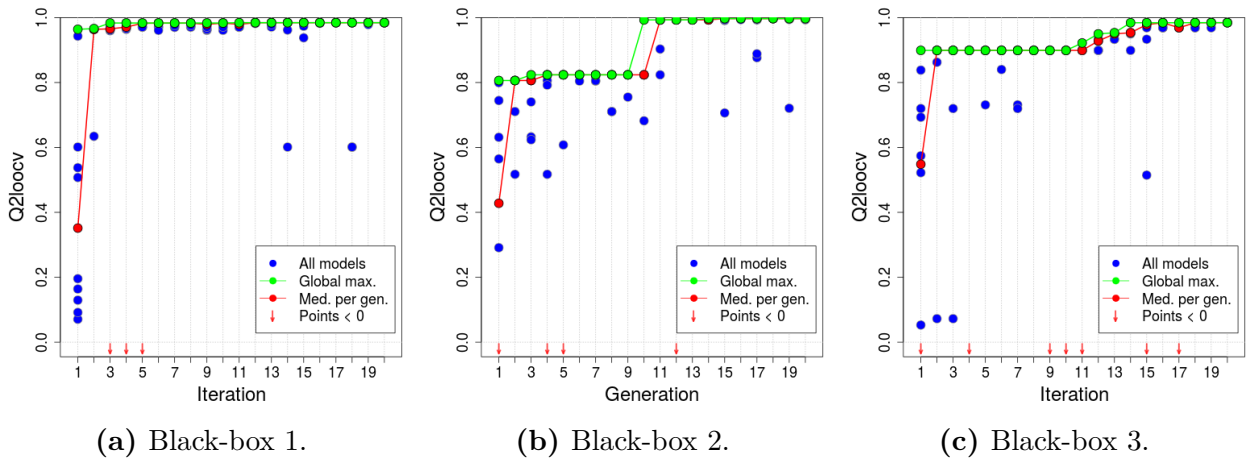


Figure 3.8 – Evolution of the heuristic.

In all cases, the maximum and median Q_{loocv}^2 progressively improve, and the models gradually converge towards the best solution found. The sporadic drops of the median are

just an effect of the randomness and the degree of exploration requested via the parameter q_0 (see (3.1)). This phenomenon is by no means a bad thing, since it prevents the algorithm from getting trapped in local optima and allows it to keep improving the best solution as we see in the plots. The optimizations presented here ran in 77.7, 30.4 and 25.8 seconds, for black-boxes 1, 2 and 3, respectively.

3.4 Conclusions

This technical report introduces an ant colony based algorithm for model selection, specially oriented to the treatment of functional inputs. After almost 30 decades of their introduction to the scientific community, the ant colony algorithms remain as powerful optimization tools for concurrent research problems of notable relevance. In this report, we validated the ability of our algorithm to develop high quality regression models through three analytic test cases. In all of them the results were satisfactory. It is important to note that the absence of a smart exploration tool like this, does not leave one exposed to the possibility of selecting exclusively one of the other high quality model structures. As evidenced through the plots on the relative quality of the models, in all the three test cases, a large number of models had regular to bad quality. Thus, the absence of this type of tool actually leaves one vulnerable to end up with a low quality model, or at least one much inferior to the one that a smart method could have found. We have already done some tests with the RISCOPE data (see [7] and [104] for more details on the data) and the results seem promising as well. An R package for Gaussian process regression with scalar and functional inputs is currently under implementation in the frame of the RISCOPE project. The proposed ant colony based algorithm is expected to be one of the main components of the package, allowing the user not only to make individual particular models, but also to find high quality model structures.

3.5 Acknowledgments

This study was conducted in the frame of the RISCOPE project, funded by the French Agence Nationale de la Recherche (ANR). We thank the ANR for this support. The application case addressed in RISCOPE motivated the development of the algorithm presented here. We are grateful to Déborah Idier and Jérémy Rohmer from BRGM for their contributions regarding the metamodeling aspects of RISCOPE.

Chapter 4

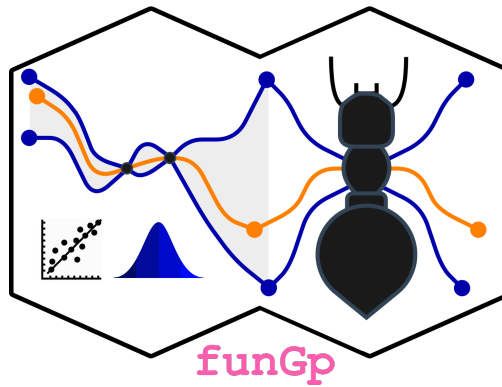
Gaussian process regression for scalar and functional inputs with funGp: The in-depth tour

The chapter in brief

The two previous chapters led us to the creation of the R package called **funGp** [20], which is oriented to the construction and smart selection of Gaussian process models with emphasis on the treatment of functional inputs. This chapter is an exact copy of its user manual [21]. **funGp** relies on a variety of concepts at a crossroad between mathematics, statistics and optimization. As such, it merits a sufficiently wide documentation helping the user to shorten its learning curve. It does not matter what a tool has to offer if people do not understand well how to use it. The user manual presented in this chapter explains all the functionalities of the package through a set of short examples in the form of code snippets copy/pasteable directly to R. We are confident that, with the help of this manual, the user will be able to have its first **funGp** model working in a matter of just a few minutes.

Contents

In-code notation	73
4.1 Base functionalities	74
4.1.1 Create a <code>funGp</code> model	74
4.1.2 Predict using a <code>funGp</code> model	76
4.1.3 Simulate from a <code>funGp</code> model	79
4.1.4 Update a <code>funGp</code> model	81
4.2 Model customizations	83
4.2.1 Kernel family	84
4.2.2 Projection basis	84
4.2.3 Projection dimension	85
4.2.4 Distance for functions	85
4.3 Heuristic model selection	87
4.3.1 Concept	87
4.3.2 Using the model factory in <code>funGp</code>	88
4.4 Parallelization in <code>funGp</code>	97
4.4.1 Parallelized hyperparameters optimization	98
4.4.2 Parallelized model selection	99
Closing discussion	99
Acknowledgements	100



Gaussian Process Regression for Scalar and Functional Inputs with funGp

The in-depth tour

This is a comprehensive guide to creating and manipulating Gaussian process regression models using the R package **funGp**. It illustrates through examples, the usage of every function in the package and each example is accompanied by a code snippet to shorten the learning curve through direct usage of the functions.

Authors: José Betancourt, François Bachoc, Thierry Klein.

Contributors: Déborah Idier, Jérémy Rhomer.

This manual is for **funGp**, version 0.1.0 (2020), downloadable from [CRAN](#) and [GitHub](#).

Recommended citation: Betancourt, J., Bachoc, F., Klein, T. (2020). R Package Manual: Gaussian Process Regression for Scalar and Functional Inputs with funGp - The in-depth tour. *RISCOPE project*.



funGp was first developed in the frame of the RISCOPE research project, funded by the French Agence Nationale de la Recherche (ANR) for the period 2017-2021 (ANR project No. 16CE04-0011, [RISCOPE.fr](#)), and certified by SAFE Cluster.

What does **funGp** bring to the table?

- **Flexible modeling of functional-input regression problems**

A narrow class of R packages address regression with functional inputs (e.g., time series). The vast majority of those packages rely on models limited by strong assumptions on the relationship between inputs and outputs (e.g., Linear, Generalized Linear or Generalized Additive Models). The few ones that suppress these limitations through more general models (e.g., Kernel Smoothing) often require the output to be a function defined over the same domain as the functional inputs, which is frequently not the case and leaves the scalar-output problem unresolved. **funGp** tackles regression problems involving scalar and/or functional inputs and a scalar output through the fairly general Gaussian process model. This is a non-parametric type of model which removes any need to set a particular input-output parametric relationship in advance, and learns this information directly from the data.

- **Built-in dimension reduction**

A common practice when working with functional data is to start by making a projection of it onto a space of lower dimension, a procedure known as dimension reduction (DR). This allows to reduce the complexity of the model while preserving the main statistical or geometric characteristics of the functions. **funGp** is self-contained in the sense that it does not depend on other packages to perform DR on the functional inputs. At this point, we provide projection onto B-splines or PCA bases. The package was designed to enable a straightforward extension to other bases in further versions.

- **Heuristic model selection**

The possibilities offered by a package often translate into alternative model structures. Just to give an example, most packages that support Gaussian process models allow to select the kernel function from a set of standard families (e.g., Gaussian, Matérn 5/2, Matérn 3/2). However, decision support is rarely offered in order to select a suitable configuration for the problem at hand. We acknowledge the potential impact of such a decision in the performance of the model [7, 22] and also the practical difficulties that arise from offering possibilities without decision support. Thus, **funGp** was equipped with a model selection functionality that allows the user to automatically search for a good combination of the so-called structural parameters of the model. At this point, an Ant-Colony-based algorithm is implemented to perform this task.

- **All-level-user-friendly**

We aim **funGp** to be a helpful tool for users within a wide range of knowledge in mathematics or statistics. Thus, we have made an effort to make simple and intuitive the way the package work. Most of the arguments in the functions have been provided default values so that the user can start experimenting with them at its own pace. Once you get ready, you will be able to start playing with the nugget effect, basis type, kernel type, multi-start option, parallelization and even the parameters of the heuristic for model selection. However, to have your first model built by **funGp**, the only thing you need to provide is your data.

In-code notation

<code>n.tot</code>	Number of points used for prediction
<code>n.tr</code>	Number of points using for learning of hyperparameters
<code>n.pr</code>	Number of prediction points
<code>n.sm</code>	Number of simulation points
<code>ds</code>	Number of scalar inputs
<code>df</code>	Number of functional inputs
<code>k</code>	Array of dimensions for the df functional inputs
<code>p</code>	Array of projection dimensions for the functional inputs
<code>K.tt</code>	Training auto-covariance matrix
<code>K.pp</code>	prediction auto-covariance matrix
<code>K.tp</code>	Training-prediction cross-covariance matrix
<code>L</code>	Lower diagonal matrix of a Cholesky decomposition

4.1 Base functionalities

This section starts from the bottom with the fundamental tasks implemented in **funGp**. Those are: (i) **creation** of regression models, (ii) **prediction** of the output at unobserved input points, (iii) **simulation** of trajectories from the underlying Gaussian process linked to any **funGp** model, and (iv) updating an existing model.

The workflow for each of the four functionalities listed above is illustrated through a follow-along example based on the analytic black-box function

$$\mathcal{G}_1 : [0, 1]^2 \times \mathcal{F}^2 \rightarrow \mathbb{R},$$
$$(\mathbf{x}, \mathbf{f}) \mapsto x^{(1)} + 2x^{(2)} + 4 \int_0^1 t f^{(1)}(t) dt + \int_0^1 f^{(2)}(t) dt,$$

with $\mathbf{x} = (x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}, x^{(5)})$ the scalar inputs, $\mathbf{f} = (f^{(1)}, f^{(2)})$ the functional inputs, and $\mathcal{F}$ the set of continuous functions from $[0, 1]$ to $\mathbb{R}$. This function corresponds to the first analytic example presented in [29], and is accessible in **funGp** through the black-box function **fgp_BB3**.

All code snippets are copy/paste-able directly to R.

4.1.1 Create a funGp model

Let us start by creating a model. To do so, we must first put the input and output data in a suitable format. The scalar inputs should be provided as a **matrix** or **data.frame**. The functional inputs should be provided as a **list** of **matrices**, one per functional input. The output should be provided as an **array** or single-column **matrix**. In the case of the inputs, each row of a **matrix** must correspond to an input point. Here, we will use synthetic data based on the analytic case defined in the introductory paragraph of this section, which involves two scalar inputs and two functional inputs. To generate the input data, we took the scalar input points from a factorial design over $[0, 1]$. For the functional inputs we assumed that those were measured at 10 and 22 time instants. This, just to emphasize the fact that functional inputs with heterogeneous discretization are valid **funGp** inputs. Just to have some data to work with, we sampled all the values of each function randomly from $U(0, 1)$. We also picked an arbitrary number of 25 training points.

```
# generating input data for training
set.seed(100)
n.tr <- 25
sIn <- expand.grid(x1 = seq(0,1,length = sqrt(n.tr)), x2 = seq(0,1,length = sqrt(n.tr)))
fIn <- list(f1 = matrix(runif(n.tr*10), ncol = 10), f2 = matrix(runif(n.tr*22), ncol = 22))

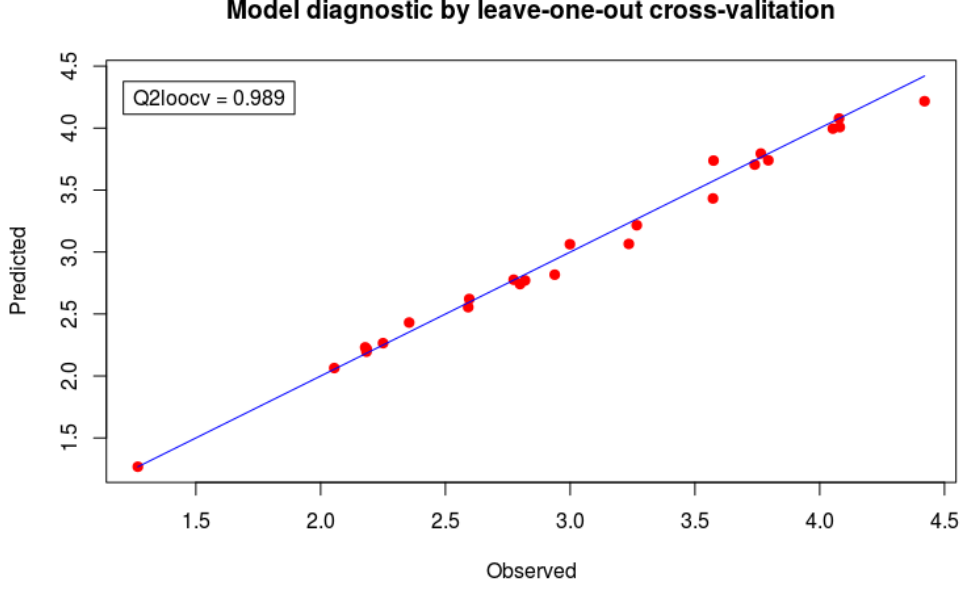
# generating output data for training
sOut <- fgp_BB3(sIn, fIn, n.tr)

# creating a funGp model
m1 <- fgpm(sIn = sIn, fIn = fIn, sOut = sOut)

R output:
** Presampling...
** Optimising...
final value 2.841058 # loglikelihood value
converged
```

The output of the `fgpm` function is an object of class `fgpm`. A calibration plot based on the Leave-one-out (LOO) predictions.

```
# plotting the model
plotLOO(m1)
```



For a design with $n.tr$ points, LOO consists of removing one observation from the design at a time, each time training the model using the remaining $n.tr - 1$ points and computing the prediction at the ignored point. In its basic version, LOO results expensive as it requires training $n.tr$ models using almost all the data each time. For Gaussian processes, the LOO predictions are often approximated based on the virtual LOO formulas [59, 105], which require a single model training.

The model diagnostic plot also displays a measure of the external prediction capability of the model. It corresponds to the LOO cross-validated squared correlation coefficient Q_{loocv}^2 , defined as:

$$Q_{loocv}^2 := 1 - \frac{\sum_{i=1}^{n.tr} (y_i - \hat{y}_{i,-i})^2}{\sum_{i=1}^{n.tr} (y_i - \bar{y})^2},$$

with $(y_i)_{i=1,\dots,n.tr}$ the vector of observed output values, $\bar{y}$ the average of that vector and $\hat{y}_{i,-i}$ the LOO prediction of y_i .

Main features of the model are printed when calling the `show` function on the model:

```
# printing the model
m1 # equivalent to show(m1)

R output:
Gaussian Process Model-----

* Scalar inputs: 2
* Functional inputs: 2

| Input | Orig. dim | Proj. dim | Basis | Distance |
|:-----:|:-----:|:-----:|:-----:|:-----:|
| F1 | 10 | 3 | B-splines | L2_bygroup |
| F2 | 22 | 3 | B-splines | L2_bygroup |

* Total data points: 25
* Trained with: 25

* Kernel type: matern5_2
* Hyperparameters:
  -> variance: 1.6404
  -> length-scale:
      ls(X1): 2.0000
      ls(X2): 2.0000
      ls(F1): 2.5804
      ls(F2): 3.0370
-----
```

The field **Proj. dim** is related to the possibility of requesting DR¹ for the functional inputs. DR allows to project a functional input of dimension k_i onto a space of lower dimension p_i while preserving the main statistical or geometric properties of the variable [33, 34]. This process often leads to $p_i \ll k_i$, which improves the tractability and processing speed of the model. By default, the `fgpm` function sets $p_i = 3$ for all the functional inputs. The user is allowed to pick a custom projection dimension for each input and also not to project some of them. Different projection methods (basis families) are also available. The projection method used for each input is indicated under the field **Basis**. The manipulation of the projection dimension and projection method are discussed in more detail in [Section 4.2.2](#) and [Section 4.2.3](#), respectively.

4.1.2 Predict using a funGp model

Now let us use our model to make predictions. To do so, we must prepare the input data corresponding to the coordinates at which the output is to be estimated. The inputs should have the same format as used for creating the model with the `fgpm` function in [Section 4.1.1](#). The scalar inputs should be provided as a **matrix** or **data.frame** and the functional inputs should be provided as a **list** of **matrices**, one per functional input. This time, each row of an input **matrix** must correspond to a prediction point.

For the example, we generated the input points in a similar way as for training, i.e., the scalar inputs from a factorial design over $[0, 1]$ and the functional values randomly from $U(0, 1)$.

¹DR: dimension reduction.

```

# building the model
set.seed(100)
n.tr <- 25
sIn <- expand.grid(x1 = seq(0,1,length = sqrt(n.tr)), x2 = seq(0,1,length = sqrt(n.tr)))
fIn <- list(f1 = matrix(runif(n.tr*10), ncol = 10), f2 = matrix(runif(n.tr*22), ncol = 22))
sOut <- fgpm_BB3(sIn, fIn, n.tr)
m1 <- fgpm(sIn = sIn, fIn = fIn, sOut = sOut)

# generating input data for prediction
n.pr <- 100
sIn.pr <- as.matrix(expand.grid(x1 = seq(0,1,length = sqrt(n.pr)), x2 = seq(0,1,length = sqrt(n.pr))))
fIn.pr <- list(f1 = matrix(runif(n.pr*10), ncol = 10), matrix(runif(n.pr*22), ncol = 22))

# making predictions
m1.preds <- predict(m1, sIn.pr = sIn.pr, fIn.pr = fIn.pr)

# checking content of the list
summary(m1.preds)

R output:
      Length Class  Mode
mean      100  -none-  numeric
sd         100  -none-  numeric
lower95    100  -none-  numeric
upper95    100  -none-  numeric

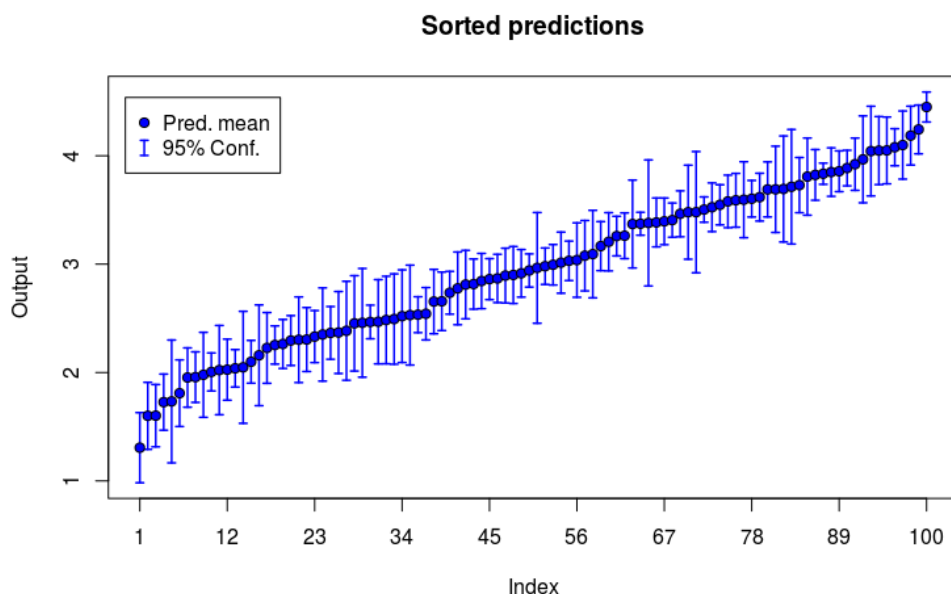
```

The output of `predict` is a **list** containing the estimated mean and standard deviation, along with the lower and upper limits of the 95% confidence intervals for the output at the prediction points. In practice, the estimated mean of a Gaussian process model is used as the prediction of the output while the standard deviation is often interpreted as a measure of the local error of the prediction. Predictions of a `funGp` model can be easily plotted by calling the `plotPreds` function on the **list** returned by `predict`. Note that the model must also be sent in the function call.

```

# plotting predictions
plotPreds(m1, preds = m1.preds)

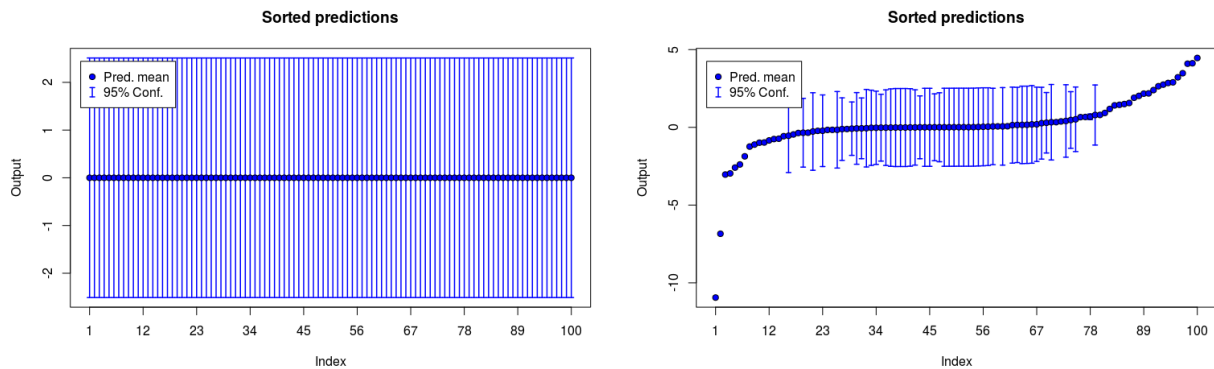
```



With functional inputs, the simple Out-vs-In scatter plots are no longer an option. Thus, `plotPreds` displays the increasingly sorted mean and corresponding confidence intervals instead. This plot can be used as a diagnostic tool for identifying potential problems related to the hyperparameters optimization, for instance:

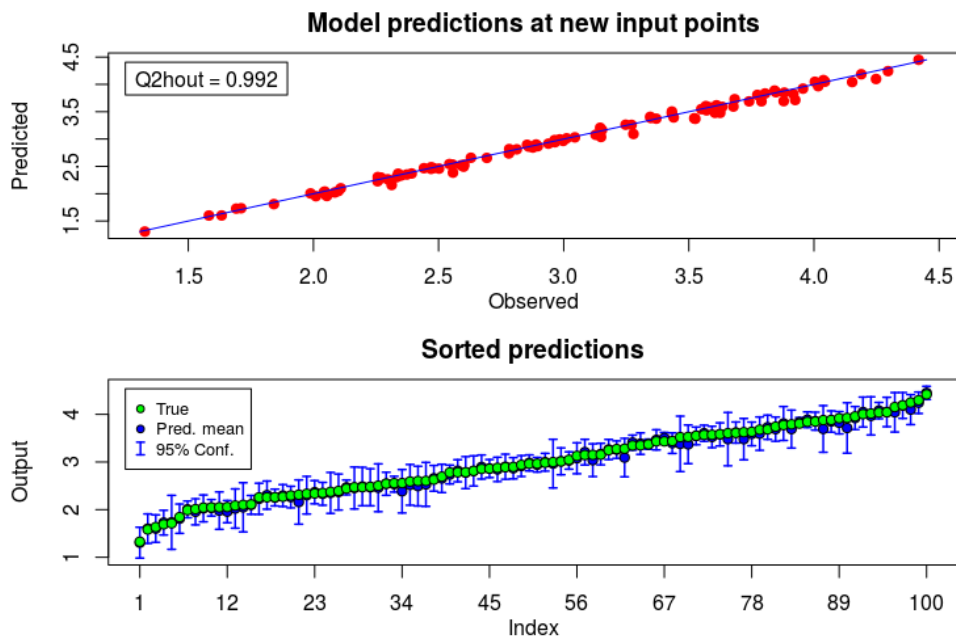
- Excessively wide confidence intervals could indicate a far-from-optimal hyperparameters' estimation, especially if it happens for all or a large number of prediction points;
- When a prediction point is included in the training set, the model interpolates the output and no confidence interval is displayed for that point. In any other case, missing confidence intervals may be indicative of far-from-optimal hyperparameters' estimation.

Figures illustrating the two aforementioned potential scenarios are displayed below.



The `plotPreds` function can also be used to compare predictions against true output values. In that case, an observed-vs-predicted calibration plot will be added on top of the sorted-output plot shown before.

```
# validating against true output
sOut.pr <- fgp_BB3(sIn.pr, fIn.pr, n.pr)
plotPreds(m1, m1.preds, sOut.pr)
```



The calibration plot made by `plotPreds` will display the predictive squared correlation coefficient Q_{hout}^2 , which corresponds to the classical coefficient of determination R^2 for a test sample, i.e., for prediction residuals [77]. On the other hand, the ordering in the sorted-output plot will be lead by the true output vector instead of the predicted mean vector. This way of sorting is convenient for comparing results of different models fitting the same data. Either the calibration plot or the sorted-output plot can be displayed alone by specifying the argument `sortp = FALSE` or `calib = FALSE`, respectively, when calling `plotPreds`.

Note: by default the `predict` function in `funGp` returns so-called *light predictions*, which include the predicted mean, standard deviation and limits of the 95% confidence intervals. Some users might be interested in *full predictions*, which also include the training-prediction cross-covariance matrix `K.tp` and the prediction auto-covariance matrix `K.pp`. To make full predictions, it suffices to set `detail = "full"` when calling `predict`. The behavior of `plotPreds` is not affected by this selection.

```
# making full predictions
m1.preds_f <- predict(m1, sIn.pr = sIn.pr, fIn.pr = fIn.pr, detail = "full")

# checking content of the list
summary(m1.preds_f)

R output:
      Length Class  Mode
mean      100  -none-  numeric
sd         100  -none-  numeric
K.tp       2500  -none-  numeric
K.pp      10000  -none-  numeric
lower95     100  -none-  numeric
upper95     100  -none-  numeric
```

4.1.3 Simulate from a `funGp` model

Simulations in `funGp` are requested through the `simulate` function, in a similar way to predictions. The scalar inputs should be provided as a **matrix** or **data.frame** and the functional inputs should be provided as a **list** of **matrices**, one per functional input. Each row of an input **matrix** will be interpreted as a point at which to provide simulations. By default, `simulate` will perform so-called *light simulations*, returning a $n.rep \times n.sm$ **matrix**, with $n.rep$ the number of replications to produce at each input point and $n.sm$ the number of input points. For the example we took the scalar inputs from a factorial design over $[0,1]$ and the functional values randomly from $U(0,1)$.

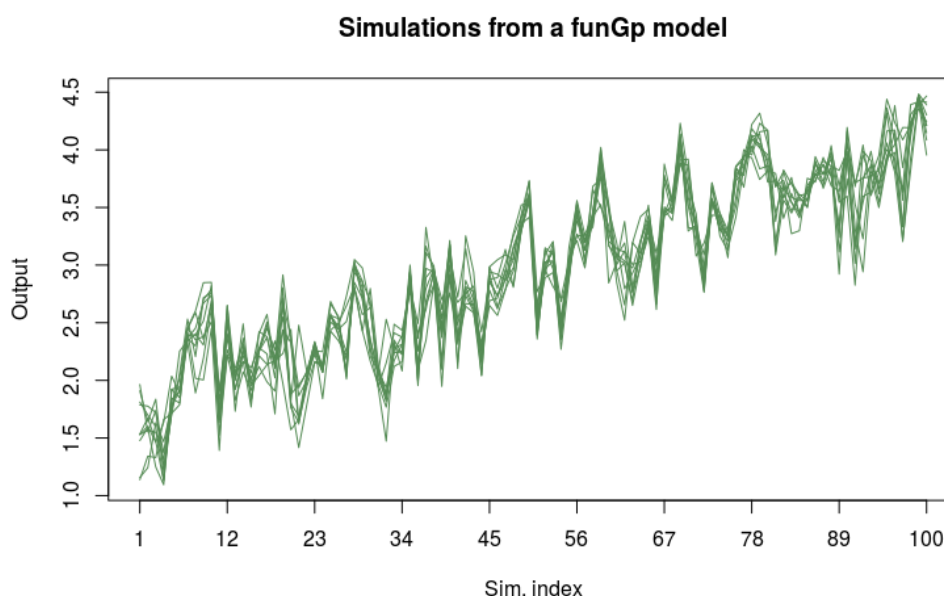
```
# building the model
set.seed(100)
n.tr <- 25
sIn <- expand.grid(x1 = seq(0,1,length = sqrt(n.tr)), x2 = seq(0,1,length = sqrt(n.tr)))
fIn <- list(f1 = matrix(runif(n.tr*10), ncol = 10), f2 = matrix(runif(n.tr*22), ncol = 22))
sOut <- fgpm(sIn, fIn, n.tr)
m1 <- fgpm(sIn = sIn, fIn = fIn, sOut = sOut)

# generating input data for simulation
n.sm <- 100
sIn.sm <- as.matrix(expand.grid(x1 = seq(0,1,length = sqrt(n.sm)), x2 = seq(0,1,length = sqrt(n.sm))))
fIn.sm <- list(f1 = matrix(runif(n.sm*10), ncol = 10), matrix(runif(n.sm*22), ncol = 22))

# making light simulations
m1.sims_l <- simulate(m1, nsim = 10, sIn.sm = sIn.sm, fIn.sm = fIn.sm)
```

Simulations in **funGp** are plotted by the **plotSims** function. In contrast to prediction plots, simulation plots do not have the output sorted in increasing order, but instead, the simulation index corresponding to the input coordinates specified by the user is set in the abscissa.

```
# plotting light simulations
plotSims(m1, m1.sims_l)
```



If requested, **simulate** will return a **list** containing the simulated output, predicted mean, standard deviation and limits of the 95% confidence intervals at the specified input coordinates. This corresponds to a *full simulation*, available through the option **detail = "full"**.

```
# making full simulations
m1.sims_f <- simulate(m1, nsim = 10, sIn.sm = sIn.sm, fIn.sm = fIn.sm, detail = "full")

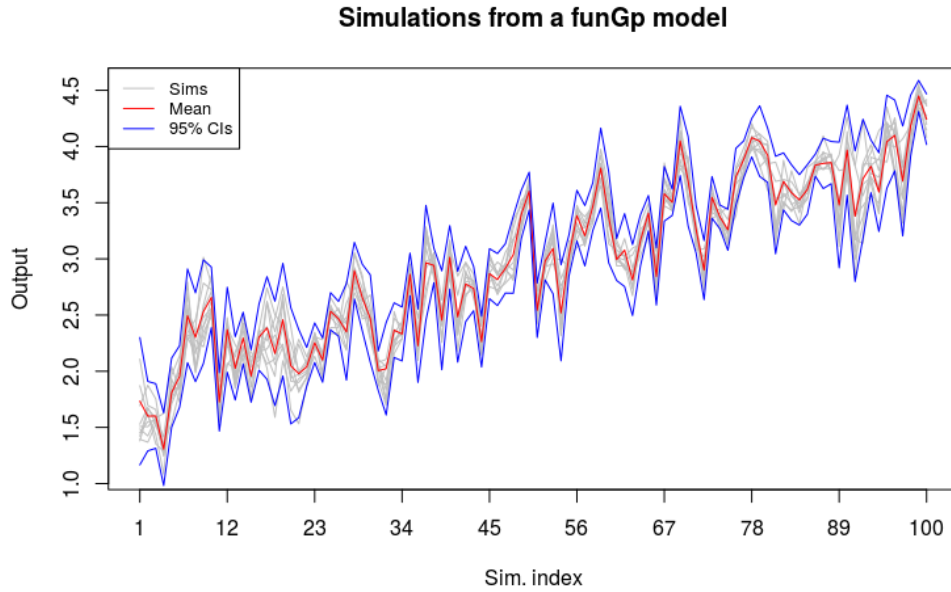
# checking content of the list
summary(m1.sims_f)
```

R output:

	Length	Class	Mode
sims	1000	-none-	numeric
mean	100	-none-	numeric
sd	100	-none-	numeric
lower95	100	-none-	numeric
upper95	100	-none-	numeric

Full simulations can also be plotted using the **plotSims** function. By default, the plot of full simulations will include the predicted mean and limits of the confidence intervals.

```
# plotting full simulations in full mode
plotSims(m1, m1.sims_f)
```



A light plot without the mean and confidence intervals is also available for full simulations by setting `detail = "light"` when calling `plotSims`.

4.1.4 Update a funGp model

As simple as it might appear, the `update` function allows to perform nine different updating tasks on a `funGp` model:

- Operations over the `@sIn`, `@fIn` and `@sOut` slots
 1. Deletion of data points
 2. Substitution of data points
 3. Addition of data points
- Operations over the `@kern@varHyp`, `@kern@s_lsHyps` and `@kern@f_lsHyps` slots
 4. Substitution of the variance hyperparameter
 5. Substitution of the vector of scalar length-scale hyperparameters
 6. Substitution of the vector of functional length-scale hyperparameters
 7. Re-estimation of the variance hyperparameter
 8. Re-estimation of the vector of scalar length-scale hyperparameters
 9. Re-estimation of the vector of functional length-scale hyperparameters

There are many reasons why you might want to modify an existing model; new observations became available, some of those used for training became obsolete, transcription or typing errors were found in the training data, you want to experiment with different values of the hyperparameters, just to mention some. In most cases, part of the work done during the construction of the original model can be exploited to make the updating process much faster than building a new model from zero. The request of the different updating tasks is illustrated in the code snippets below. If you have not built a model yet using the code

provided in previous sections, you can use the following one to obtain a model on which to perform the update tasks of the upcoming examples.

```
# building the model
set.seed(100)
n.tr <- 25
sIn <- expand.grid(x1 = seq(0,1,length = sqrt(n.tr)), x2 = seq(0,1,length = sqrt(n.tr)))
fIn <- list(f1 = matrix(runif(n.tr*10), ncol = 10), f2 = matrix(runif(n.tr*22), ncol = 22))
sOut <- fgpb_B3(sIn, fIn, n.tr)
m1 <- fgpm(sIn = sIn, fIn = fIn, sOut = sOut)
```

• Deletion and addition of data points

```
# deleting two points
ind.dl <- sample(1:m1@n.tot, 2)
m1up <- update(m1, ind.dl = ind.dl)

R output:
* Complete tasks:
  - data deletion

# adding five points
n.nw <- 5
sIn.nw <- matrix(runif(n.nw * m1@ds), nrow = n.nw)
fIn.nw <- list(f1 = matrix(runif(n.nw*10), ncol = 10), f2 = matrix(runif(n.nw*22), ncol = 22))
sOut.nw <- fgpb_B3(sIn.nw, fIn.nw, n.nw)
m1up <- update(m1, sIn.nw = sIn.nw, fIn.nw = fIn.nw, sOut.nw = sOut.nw)

R output:
* Complete tasks:
  - data addition
```

• substitution of data points

```
# generating substituting input data for updating
n.sb <- 2
sIn.sb <- matrix(runif(n.sb * m1@ds), nrow = n.sb)
fIn.sb <- list(f1 = matrix(runif(n.sb*10), ncol = 10), f2 = matrix(runif(n.sb*22), ncol = 22))

# generating substituting output data for updating
sOut.sb <- fgpb_B3(sIn.sb, fIn.sb, n.sb)

# generating indices for substitution
ind.sb <- sample(1:(m1@n.tot), n.sb)

# updating all, the scalar inputs, functional inputs and the output
m1up <- update(m1, sIn.sb = sIn.sb, fIn.sb = fIn.sb, sOut.sb = sOut.sb, ind.sb = ind.sb)

R output:
* Complete tasks:
  - data substitution
```

Substituting points only from some of the data structures is also possible.

```
# substituting some data structures
m1up1 <- update(m1, sIn.sb = sIn.sb, ind.sb = ind.sb) # only the scalar inputs
m1up2 <- update(m1, sOut.sb = sOut.sb, ind.sb = ind.sb) # only the output
m1up3 <- update(m1, sIn.sb = sIn.sb, sOut.sb = sOut.sb, ind.sb = ind.sb) # the scalar inputs and the output

R output:
* Complete tasks:
  - data substitution
```

• Substitution of hyperparameters

```
# defining hyperparameters for substitution
var.sb <- 3
ls_s.sb <- c(2.44, 1.15)
ls_f.sb <- c(5.83, 4.12)

# updating the model
m1up <- update(m1, var.sb = var.sb, ls_s.sb = ls_s.sb, ls_f.sb = ls_f.sb)

R output:
* Complete tasks:
  - var substitution
  - scalar length-scale substitution
  - functional length-scale substitution
```

Substituting only one of the three data structures is possible as well.

```
# updating the model
m1up <- update(m1, var.sb = var.sb) # only the variance
m1up <- update(m1, ls_f.sb = ls_f.sb) # only the functional length-scale parameters
m1up <- update(m1, var.sb = var.sb, ls_s.sb = ls_s.sb) # only the variance and the scalar ls. parameters
```

• Re-estimation of hyperparameters

```
# re-estimating the hyperparameters
m1up <- update(m1, var.re = TRUE) # only the variance
m1up <- update(m1, ls_s.re = TRUE) # only the scalar length-scale parameters
m1up <- update(m1, ls_s.re = TRUE, ls_f.re = TRUE) # all length-scale parameters
m1up <- update(m1, var.re = TRUE, ls_s.re = TRUE, ls_f.re = TRUE) # all hyperparameters

R output:
* Complete tasks:
  - var re-estimation
  - scalar length-scale re-estimation
  - functional length-scale re-estimation
```

It is possible to request multiple tasks from the different categories listed above in a single call to `update`. When doing so, it is convenient to keep in mind that tasks will be performed in the following order:

data deletion/substitution → data addition → hypers substitution/re-estimation

It is also good to remember that the following two combinations are unfeasible:

- Data points deletion and substitution;
- Substitution and re-estimation of the same hyperparameter.

4.2 Model customizations

There are multiple things we can do in order to improve the tractability and predictability of a `funGp` model. In this section we discuss the customization of the model through its so-called structural parameters. It refers to a set of categorical features such as the kernel function or the projection basis, whose levels could be alternated in order to generate different models departing from the same input-output data. Without going too deep into the technical details, this section explains how to start working on these features within `funGp` and the interested reader is referred to [7] for a formal and more detailed explanation of the underlying theory.

4.2.1 Kernel family

The selection of a suitable kernel function is something that naturally comes to mind when working with Gaussian process models. At this point, **funGp** offers the possibility to choose among the Gaussian, Matérn 5/2 and Matérn 3/2 kernels. This selection can be specified when calling the **fgpm** function, through the parameter **kertype**. Valid values for this attribute are **"gauss"**, **"matern5_2"** and **"matern3_2"**. See for instance the example below with the Gaussian kernel.

```
# building the model
set.seed(100)
n.tr <- 25
sIn <- expand.grid(x1 = seq(0,1,length = sqrt(n.tr)), x2 = seq(0,1,length = sqrt(n.tr)))
fIn <- list(f1 = matrix(runif(n.tr*10), ncol = 10), f2 = matrix(runif(n.tr*22), ncol = 22))
sOut <- fgpm_BB3(sIn, fIn, n.tr)
m1 <- fgpm(sIn = sIn, fIn = fIn, sOut = sOut, kerType = "gauss")
```

By default, **fgpm** uses the Matérn 5/2 function, which is a popular choice in the Machine Learning (ML) community.

4.2.2 Projection basis

In earlier sections of the manual we talked about DR², the process of reducing the dimension of your data structures in such a way and extent that the model becomes significantly more tractable and the loss in terms of predictability is negligible, if some. A common DR approach when dealing with functional inputs is to project each functional-input **matrix** onto a space of lower dimension. This method requires the construction of a set of basis vectors on which the original curves are projected. Those vectors (typically referred to as basis functions) may come from diverse families, including among the most popular ones the B-splines [71], PCA [73], PLS [74], wavelets [106] and kPCA [107]. The suitability of a given basis type might depend on the regression instance at hand. The B-splines and PCA bases are currently implemented in **funGp** for the projection of functional inputs. This option is accessible in the **fgpm** function through the parameter **f_basType**, which can be set to the values **"B-splines"** or **"PCA"**. When multiple functional inputs are provided, a custom basis can be selected for each of them, by passing an **array** with the selection for each input. If multiple functional inputs are provided, but a single **f_basType** value is specified, that selection is used for all the inputs. Both cases are illustrated below. By default, all functional inputs use a B-splines basis.

```
# generating input and output data
set.seed(100)
n.tr <- 25
sIn <- expand.grid(x1 = seq(0,1,length = sqrt(n.tr)), x2 = seq(0,1,length = sqrt(n.tr)))
fIn <- list(f1 = matrix(runif(n.tr*10), ncol = 10), f2 = matrix(runif(n.tr*22), ncol = 22))
sOut <- fgpm_BB3(sIn, fIn, n.tr)

# building the model
# different basis for each functional input
m1 <- fgpm(sIn = sIn, fIn = fIn, sOut = sOut, f_basType = c("B-splines", "PCA"))

# same basis for both functional inputs
m1 <- fgpm(sIn = sIn, fIn = fIn, sOut = sOut, f_basType = "PCA")
```

²DR: dimension reduction.

4.2.3 Projection dimension

This parameter is highly influential in both, the predication quality and the tractability of the model. Ideally, one wants to set the projection dimension considerably lower than the original one, but not so low that significant prediction power is lost. In the `fgpm` function, you can specify the projection dimension for each input by setting the argument `f_pdims`. Valid inputs are all the integer numbers from 0 to the original dimension of the curves. The value 0 is used to request to not perform the projection of an input. If there are multiple functional inputs, an **array** can be provided instead of a single value in order to specify custom projection dimensions. If a single value is specified and multiple functional inputs are identified, the value is used as projection dimension for all the functional inputs. By default, all functional inputs are projected onto a space of dimension 3.

```
# generating input and output data
set.seed(100)
n.tr <- 25
sIn <- expand.grid(x1 = seq(0,1,length = sqrt(n.tr)), x2 = seq(0,1,length = sqrt(n.tr)))
fIn <- list(f1 = matrix(runif(n.tr*10), ncol = 10), f2 = matrix(runif(n.tr*22), ncol = 22))
sOut <- fgpm_BB3(sIn, fIn, n.tr)

# building the model
# the first input not projected, the second one projected in dimension 7
m1 <- fgpm(sIn = sIn, fIn = fIn, sOut = sOut, f_pdims = c(0, 7))

# both inputs projected in dimension 5
m1 <- fgpm(sIn = sIn, fIn = fIn, sOut = sOut, f_pdims = 5)
```

4.2.4 Distance for functions

Many regression models require the computation of the distance between the design points in order to determine which ones are the most influential in a given prediction. This is the case of Gaussian process models, which use such distances to compute the correlation between pairs of observations. A set of scaling factors called length-scale coefficients are normally used to quantify the rate of change of the output in terms of each input. For scenarios with only scalar inputs, the rule is simply to use one length-scale parameter per input, which yields the distance

$$\|\mathbf{x} - \tilde{\mathbf{x}}\|_{L^2, \boldsymbol{\theta}_s} := \sqrt{\sum_{k=1}^{ds} \frac{\|x^{(k)} - \tilde{x}^{(k)}\|^2}{(\theta_s^{(k)})^2}}, \quad (4.1)$$

with $\mathbf{x} = (x^{(1)}, \dots, x^{(ds)})$ and $\tilde{\mathbf{x}} = (\tilde{x}^{(1)}, \dots, \tilde{x}^{(ds)})$ two scalar input points, ds the number of scalar inputs in the model, $\|\cdot\|$ the L^2 norm for scalars (just the absolute value), and $\boldsymbol{\theta}_s = (\theta_s^{(1)}, \dots, \theta_s^{(ds)})$ the vector of length-scale parameters for the scalar inputs.

In an instance with functional inputs, the norm $\|\cdot\|$ needs to be replaced by a norm suitable for functions. Two options are currently implemented in `funGp`, both based on a projection of each functional inputs of the form

$$\Pi(f^{(k)})(t) = \sum_{r=1}^{p_k} \alpha_r^{(k)} B_r^{(k)}(t), \quad (4.2)$$

with $f^{(k)}$ a curve of the k -th functional input, $B_r^{(k)}$ the r -th basis function used for its projection, $\alpha_r^{(k)}$ the corresponding projection coefficient, and p_k the projection dimension.

The first type of distance implemented for functions considers each curve as a whole and uses a single length-scale parameter per functional input. This distance is defined as

$$\|\Pi(\mathbf{f}) - \Pi(\tilde{\mathbf{f}})\|_{D, \boldsymbol{\theta}_f} := \sqrt{\sum_{k=1}^{df} \frac{\int_{T_k} \left(\sum_{r=1}^{p_k} (\alpha_r^{(k)} - \tilde{\alpha}_r^{(k)}) B_r^{(k)}(t) \right)^2 dt}{(\theta_f^{(k)})^2}}, \quad (4.3)$$

with $\mathbf{f} = (f^{(1)}, \dots, f^{(df)})$ and $\tilde{\mathbf{f}} = (\tilde{f}^{(1)}, \dots, \tilde{f}^{(df)})$ two functional input points, df the number of scalar inputs in the model, $T_k \subset \mathbb{R}$ the domain of $f^{(k)}$, and $\boldsymbol{\theta}_f = (\theta_f^{(1)}, \dots, \theta_f^{(df)})$ the vector of length-scale parameters for the functional inputs. This distance is identified in the package as **L2_bygroup**, since it uses a single length-scale parameter for the group of projection terms corresponding to one functional input. **funGp** implements an efficient computation of (4.3), introduced in [29] and further studied in [7].

The second type of distance works only with the projection coefficients and disregards the basis functions. The distance is defined as

$$\|\Pi(\mathbf{f}) - \Pi(\tilde{\mathbf{f}})\|_{S, \dot{\boldsymbol{\theta}}_f} := \sqrt{\sum_{k=1}^{df} \sum_{r=1}^{p_k} \frac{(\alpha_r^{(k)} - \tilde{\alpha}_r^{(k)})^2}{(\dot{\theta}_{f,r}^{(k)})^2}}, \quad (4.4)$$

where $\dot{\boldsymbol{\theta}}_f = (\dot{\theta}_{f,r}^{(k)})_{1 \leq r \leq p_k, 1 \leq k \leq df}$ denotes the vector of functional length-scale coefficients. Note that this distance uses one length-scale coefficient per projection term. This might enable a better modeling of the input-output relationship, but in turn it implies a larger number of decision variables involved in the learning process, which makes it a harder/longer task. This distance is identified in the package as **L2_byindex** since it involves a length-scale parameter per projection index. It corresponds to the most common approach nowadays, which is to perform the projection of the inputs and then use each projection coefficient as an individual scalar input of the model.

In the case that no projection is requested for some input, both distances (4.3) and (4.4) use the original function values instead of the projection coefficients, and the identity is used in (4.3) as the matrix of basis functions. Our aim is to keep this manual friendly with users not expert in statistics. Thus, we leave at this point the technical discussion on the distances, and we refer the interested user to [7], where this aspect is discussed formally and in more detail. Below, there are some examples on the selection of the distance through **fgpm**.

```
# generating input and output data
set.seed(100)
n.tr <- 25
sIn <- expand.grid(x1 = seq(0,1,length = sqrt(n.tr)), x2 = seq(0,1,length = sqrt(n.tr)))
fIn <- list(f1 = matrix(runif(n.tr*10), ncol = 10), f2 = matrix(runif(n.tr*22), ncol = 22))
sOut <- fgpm_BB3(sIn, fIn, n.tr)
```

```

# original dimensions
# f1: 10
# f2: 22

# building the model
# the first f. input using by-index distance and no projection -> 10 length-scale parameters
# the second f. input using by-group distance -> 1 length-scale parameter
m1 <- fgpm(sIn = sIn, fIn = fIn, sOut = sOut, f_pdims = c(0,5), f_disType = c("L2_byindex", "L2_bygroup"))

# both f. inputs using by-group distance -> 2 length-scale parameters
m1 <- fgpm(sIn = sIn, fIn = fIn, sOut = sOut, f_pdims = c(0,5), f_disType = "L2_bygroup")

# both f. inputs using by-index distance -> (10+5) = 15 length-scale parameters
m1 <- fgpm(sIn = sIn, fIn = fIn, sOut = sOut, f_pdims = c(0,5), f_disType = "L2_byindex")

```

4.3 Heuristic model selection

In the previous section, we covered the base functionalities of **funGp**. Now, we present a boosting feature that takes **funGp** models one step further: the **funGp** model factory.

4.3.1 Concept

The **fgpm** function, explored in [Section 4.1.1](#), allows to specify through its arguments a number of characteristics of the model, oriented to make it adaptive to the particularities of the regression problem at hand. In [Section 4.2](#) we catalogued those features under the name of structural parameters of the model, and we illustrated through examples the way of specifying the required configuration of them to the **fgpm** function. In its current version, **funGp** includes the kernel family, the projection basis, the projection dimension and the distance for functions as structural parameters modifiable by the user. But, which combination of structural parameters should you use? If you have strong evidence to think that some level of one of these features will perform better than the others, then you are good to go. Otherwise, it would be better to make some tests in order to make such a decision. As shown by us through a set of computer experiments in [\[7\]](#), the ideal model configuration might likely depend on the particular regression task. Through the **fgpm_factory** function we enable the user to conduct a smart exploration of the solution space composed of all the possible structural parameter configurations. Variable selection is embedded in the optimization through the definition of structural parameters related to the state of each scalar and functional input in the model (active or inactive).

At this point, **funGp** performs heuristic optimization of structural parameters (model selection) by means of the ant colony based algorithm introduced by us in [\[19\]](#) ([find online ↗](#)). For a set of ds scalar inputs and df functional inputs, the optimization problem addressed by our algorithm consists in making the following decisions:

- State of the i -th scalar input (inactive, active);
- State of the j -th functional input (inactive, active);
- Projection basis for the j -th functional input ($B_1, \dots, B_z$);
- Projection dimension for the j -th functional input ($0, \dots, k_j$);
- Distance for the j -th functional input ($D_1, \dots, D_w$);
- Kernel type ($K_1, \dots, K_x$),

with $i \in \{1, \dots, ds\}$, $j \in \{1, \dots, df\}$ and k_j the original dimension of input j . The sets $\{B_1, \dots, B_z\}$, $\{D_1, \dots, D_w\}$ and $\{K_1, \dots, K_x\}$ correspond to the basis, distance and kernel families to be considered, in that order. The projection dimension 0 denotes no projection. In order to find a suitable combination of the parameters listed above, we let our artificial ants to move through a network with a structure similar to the one depicted in Figure 4.1. Such a structure prevents the constitution of senseless solutions (e.g., an input being both, inactive and active) and helps to keep the network data structures considerably simple by only defining strictly necessary links.

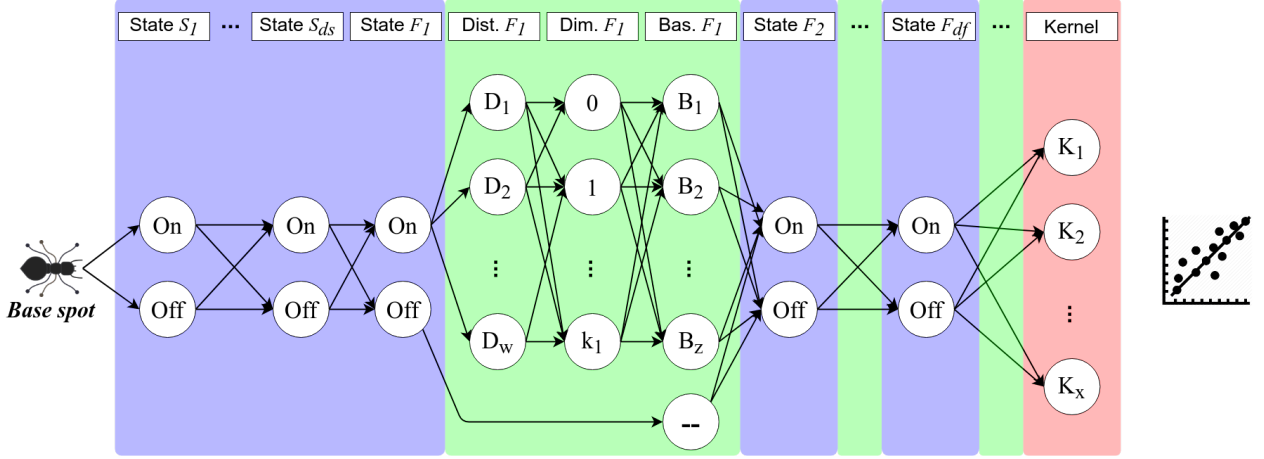


Figure 4.1 – Decision network used by the ant colony based heuristic for model selection. One end-to-end path over the network provides a feasible set of structural parameters.

The implementation of the algorithm in **funGp** strictly considers the levels of kernel function, projection basis and distance type, listed in Sections 4.2.1, 4.2.2 and 4.2.4, respectively. However, both the foundations of the approach and the code implementations are general enough to be easily extended to other levels in future versions of the package. This manual does not go further into the methodological details of the algorithm, however, a detailed explanation of it is offered in [19] for the interested reader.

4.3.2 Using the model factory in **funGp**

In this section we explain how to manipulate the **fgpm_factory** function in order to get optimized model structures. The examples in this section are based on the analytic black-box function

$$\begin{aligned} \mathcal{G}_2 : \quad [0, 1]^5 \times \mathcal{F}^2 &\rightarrow \mathbb{R}, \\ (\mathbf{x}, \mathbf{f}) &\mapsto \left(x^{(2)} + 4x^{(3)} - \frac{5}{4\pi^2} (x^{(1)})^2 + \frac{5}{\pi} x^{(1)} - 6 \right)^2 \\ &\quad + 10 \left(1 - \frac{1}{8\pi} \right) \cos(x^{(1)}) (x^{(2)})^2 (x^{(5)})^3 + 10 \\ &\quad + \frac{4}{3}\pi \left(42 \sin(x^{(4)}) \int_0^1 15 f^{(1)}(t) (1-t) - 5 dt \right. \\ &\quad \left. + \pi \left(\frac{x^{(1)}x^{(5)} + 5}{5} + 15 \right) \int_0^1 15 t f^{(2)}(t) dt \right), \end{aligned}$$

with $\mathbf{x} = (x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}, x^{(5)})$ the scalar inputs, $\mathbf{f} = (f^{(1)}, f^{(2)})$ the functional inputs, and $\mathcal{F}$ the set of continuous functions from $[0, 1]$ to $\mathbb{R}$. This function is inspired by the second analytic example studied in [29], with three additional scalar inputs allocated over the different terms of the equations to increase a bit its complexity. This function is accessible in **funGp** through the black-box function **fgp_BB7**. Here we generate the scalar and functional input values in a similar way to how we did in the previous sections.

• Getting started

Let us open this section with a basic call to the factory using its default attribute values.

```
# generating input and output data
set.seed(100)
n.tr <- 32
sIn <- expand.grid(x1 = seq(0,1,length = n.tr^(1/5)), x2 = seq(0,1,length = n.tr^(1/5)),
                  x3 = seq(0,1,length = n.tr^(1/5)), x4 = seq(0,1,length = n.tr^(1/5)),
                  x5 = seq(0,1,length = n.tr^(1/5)))
fIn <- list(f1 = matrix(runif(n.tr*10), ncol = 10), f2 = matrix(runif(n.tr*22), ncol = 22))
sOut <- fgp_BB7(sIn, fIn, n.tr)

# calling the funGp factory
xm <- fgpm_factory(sIn = sIn, fIn = fIn, sOut = sOut) # (~10 seconds)
```

The output of **fgpm_factory** is an object of class **Xfgpm**. It includes a variety of information on it that we will be explored later in detail, towards the end of this section. For now, let us concentrate on the **@model** slot, which contains the selected regression model. This is an object of type **fgpm**, which can be plotted using the **plotL00** function. Just to illustrate, let us compare our optimized model with that obtained if we arbitrarily use the default argument values in the **fgpm** function, i.e., using **fgpm(sIn = sIn, fIn = fIn, sOut = sOut)**.

```
# plotting the optimized model
plotL00(xm@model)

# plotting the model of default fgpm structural configuration
m1 <- fgpm(sIn = sIn, fIn = fIn, sOut = sOut)
plotL00(m1)
```

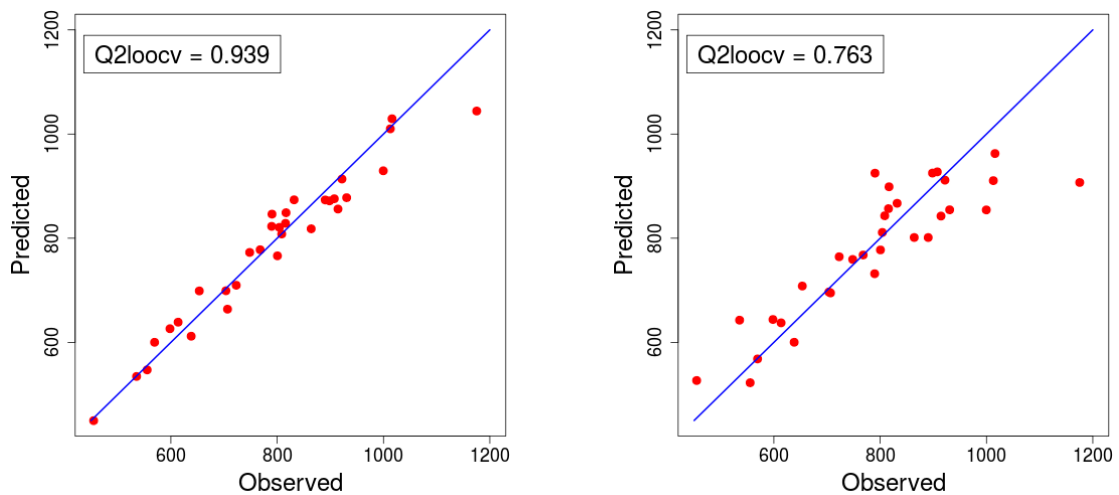


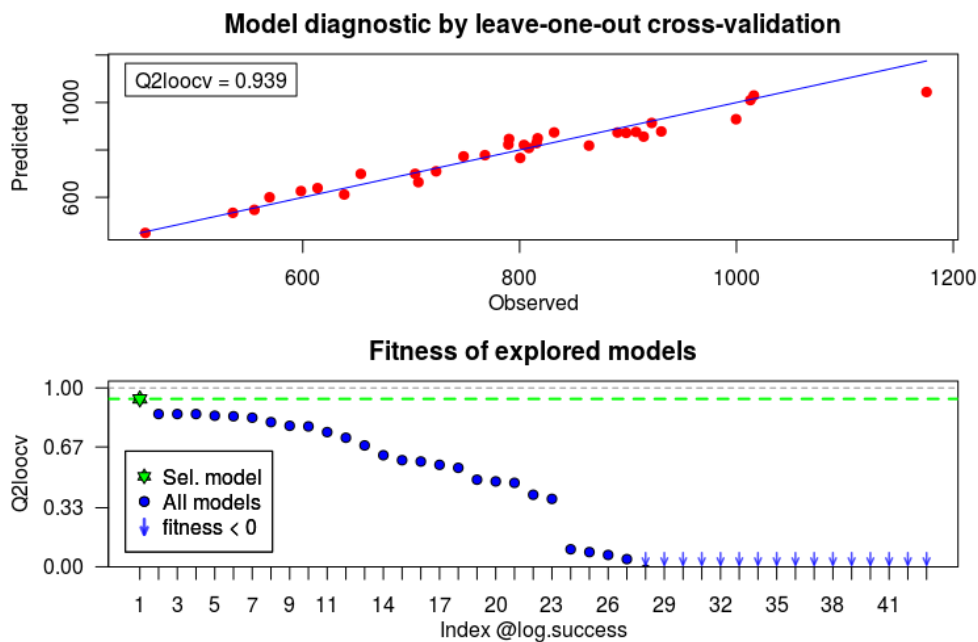
Figure 4.2 – Calibration plot of two structural configurations for the same input and output data. Left panel: optimized configuration. Right panel: unoptimized, arbitrary configuration.

Right away, just by calling `fgpm_factory` with its default arguments, we were able to find a model of greater quality. Some key points in the light of this first result are:

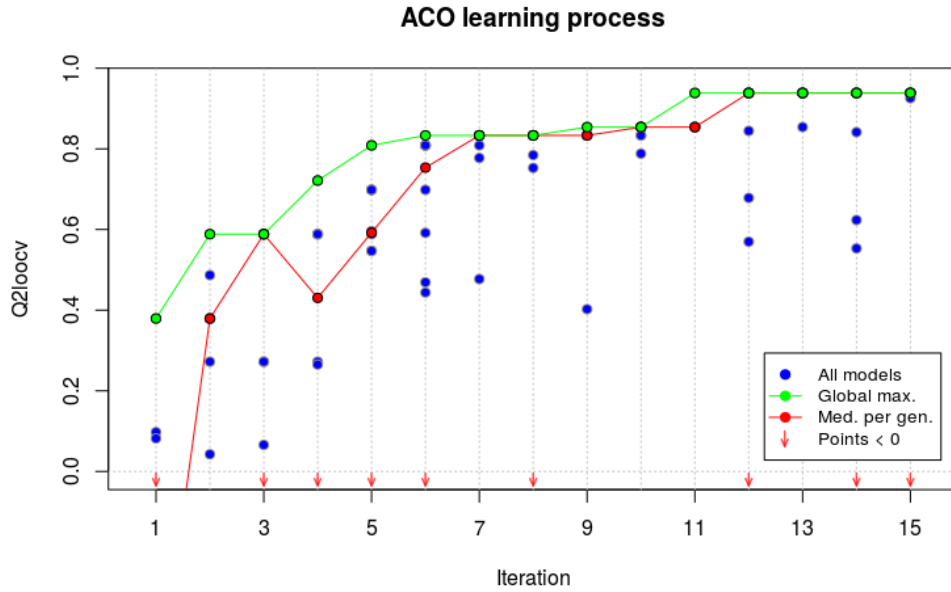
- Firstly, the superiority of the optimized model does not imply that the default argument values of the `fgpm` function are bad. They are just not tailored to this specific regression instance, contrarily to the structural parameters selected by `fgpm_factory`. That is the purpose of having `fgpm_factory` in the package, to be able to find good structural parameters for any regression instance that `fgpm` could handle.
- Secondly, the result does not mean that `funGp` models should always be made through `fgpm_factory`. In this example we see how the unoptimized model still presents a high Q^2_{loocv} . However, if there is time, we strongly recommend to perform the optimization.
- Finally, the superiority of the model delivered by `fgpm_factory` is exclusively fostered by the optimization of the structural parameter configuration, and has nothing to do with the mechanism for the optimization of the hyperparameters. Each model evaluated by `fgpm_factory` is internally created by a call to `fgpm`. Thus, the same mechanism of hyperparameter optimization is used by both functions.

Let us move on with the explanation of the usage of `fgpm_factory`. The outputs of this function can be plotted by either the `plotX` or the `plotEvol` function. The former one provides a notion of the absolute and relative quality of the selected model, and the second one illustrates the evolution of the quality of the explored models along the iterations.

```
# displaying plots on the quality of the selected model
plotX(xm)
```

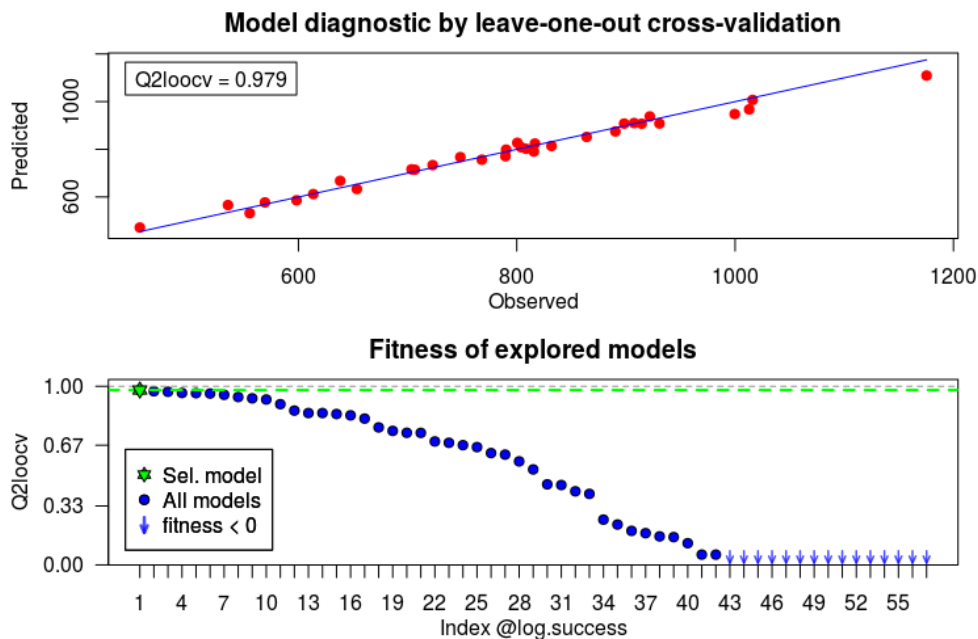


```
# plotting the evolution of the objective function
plotEvol(xm)
```



Even after multiple iterations, some points still fall relatively far from the maximum. This happens mainly because we have multiple categorical features, whose alteration might change the performance statistic in a nonsmooth way. Nonetheless, the median stays close to the maximum, which confirms that the exploration is converging towards the best known solutions. On the other hand, the points that fall below zero usually correspond to models whose hyperparameters were hard to optimize. This occurs sporadically during the log-likelihood optimization for Gaussian processes, due to the non-linearity of the objective function. An easy way to improve the quality of the selected model is just to let the algorithm complete more iterations. This can be done through the argument **setup**, as below.

```
# calling the funGp factory
set.seed(100)
xm <- fgpm_factory(sIn = sIn, fIn = fIn, sOut = sOut, setup = list(n.iter = 30)) # (~6.5 seconds)
```



In the examples above, `fgpm_factory` optimized the model structure for Q_{loocv}^2 . Optimizing for Q_{hout}^2 (i.e., validating against external observations) is also possible. This type of optimization can be requested by specifying the indices that should be used for training and validation. For instance, assume that we have the same data as in the previous example, but now we want to use about 85% of the points for training and the remaining ones for validation. This can be specified to `fgpm_factory` through the `ind.v1` argument as follows.

```
# generating validation indices
ind.v1 <- sample(seq_len(n.tr), 5) # about 15% of points for validation

# calling the funGp factory
xm <- fgpm_factory(sIn = sIn, fIn = fIn, sOut = sOut, ind.v1 = ind.v1) # (~2 seconds)
```

With this call, the factory trains each model using all the data except for the points specified by `ind.v1`. Once built, each model is used to predict the output at the points ignored during training, and the predictive squared correlation coefficient Q_{hout}^2 [77] is computed. This procedure ensures fairness in the comparison, since all the models use the same training and validation sets. In order to account for the sampling noise, the user may want to use multiple training-validation pairs of sets. This option is easily requested to the factory by passing a **matrix** instead of an **array** through the argument `ind.v1`. Such a **matrix** should have the indices for one training set on each column. This means, that the **matrix** should have as many rows as validation points, and as many columns as replicates.

```
# generating validation indices
ind.v1 <- replicate(30, sample(seq_len(n.tr), 5)) # about 15% of points for validation, 30 replicates

# calling the funGp factory
xm <- fgpm_factory(sIn = sIn, fIn = fIn, sOut = sOut, ind.v1 = ind.v1) # (~4 minutes)
```

The larger the number of replicates, the longer the optimization will be, but also the less noise will appear on the statistics used to compare the models. [Section 4.4](#) addresses the reduction of processing time through parallelization.

Note that the calibration plot produced by `plotX` will always report the Q_{loocv}^2 statistic, regardless of whether this or the Q_{hout}^2 was used for the optimization of the structural parameters. In contrast, the bottom frame will always display the statistic used during the optimization. When validation indices are provided, the model stored in the `@model` slot of the `Xfgpm` object will be one trained with as many points as remain once the specified validation points are removed. When multiple validation sets are specified, the model stored in the `@model` slot of the `Xfgpm` object will be selected in two steps by: (i) identifying the structural configuration of higher average Q_{hout}^2 ; and (ii) pick the replicate of best structural configuration with higher Q_{hout}^2 .

• Setting up the parameters of the heuristic

Our model selection algorithm relies on a set of parameters typical of any ant colony based method. Roughly speaking, those parameters control the number of individuals and iterations, the degree of exploration and rate of convergence, along with the learning-reinforcement mechanism in the algorithm. The default values of those parameters in `funGp` were selected based on the values used by Dorigo et al. in the introductory paper of the Ant Colony System [100]. We validated the suitability of that setting for our model selection problem through a large set of trials involving different black-box functions like the one defined at

the beginning of this manual ([Section 4.1](#)), and more than 10 others that raised in the frame of the RISCOPE research project [50] (see [19] for more details). Here we explain how to modify the parameters of the heuristic in case the user wants to experiment with them. Our algorithm performs based on the following list of parameters:

Initial pheromone load

- **tao0**: initial pheromone load on links pointing out to the selection of a distance type, a projection basis or a kernel type. **Default: 0.1**.
- **dop.s**: factor to control how likely it is to activate a scalar input. It operates on a relation of the type $\mathbf{A} = \mathbf{dop.s} * \mathbf{I}$, where $\mathbf{A}$ is the initial pheromone load of links pointing out to the activation of scalar inputs and $\mathbf{I}$ is the initial pheromone load of links pointing out to their inactivation. **Default: 1**.
- **dop.f**: analogous to **dop.s** for functional inputs. **Default: 1**.
- **delta.f** and **dispr.f**: shape parameters for the regularization function that determines the initial pheromone values on the links connecting the **L2_byindex** distance (see [Section 4.2.4](#)) with the projection dimension*. **Default: 2** and **1.4**, respectively.

Local pheromone update

- **rho.l**: pheromone evaporation rate*. **Default: 0.1**.

Global pheromone update

- **u.gbest**: the algorithm works in an iterative fashion; should the pheromone load on the links of the best ant so far over all the iterations be reinforced? **Default: FALSE**.
- **n.ibest**: the algorithm always reinforces the links of the best **n.ibest** ants of each iteration; how many ants should be considered for reinforcement? **Default: 1**.
- **rho.g**: learning reinforcement rate*. **Default: 0.1**.

Population factors

- **n.iter**: number of iterations. Each iteration involves the exploration of the solution space, constitution of a set of model configurations, evaluation of their performance in prediction and system feedback. **Default: 15**.
- **n.pop**: number of ants per iteration; each ant corresponds to one solution to the problem, in this case, a structural configuration for the model. **Default: 10**.

Bias strength

- **q0**: ants use one of two rules to select their next node at each step. The first rule leads the ant through the link with higher pheromone load; the second rule works based on probabilities which are proportional to the pheromone load on the feasible links. The ants will randomly chose one of the two rules at each time. They will opt for rule 1 with probability **q0** *. **Default: 0.95**. For larger number of input variables, we recommend to slightly reduce it to e.g., 0.90. This might facilitate the testing of each input in at least a few models.

The parameters marked with an asterisk (*) are explained more thoroughly in [19]. All the parameters listed above can be accessed in a **fgpm_factory** call through the argument **setup**, which should be a **list**. Below an example using arbitrary setup values.

```
# calling the funGp factory with an arbitrary setup
mysup <- list(tao0 = .15, dop.s = 1.2, dop.f = 1.3, delta.f = 4, dispr.f = 1.1, rho.l = .2,
             u.gbest = TRUE, n.ibest = 2, rho.g = .08, n.iter = 30, n.pop = 12, q0 = .85)
xm <- fgpm_factory(sIn = sIn, fIn = fIn, sOut = sOut, setup = mysup) # (~18 seconds)
```

• Defining the solution space

By default, `fgpm_factory` considers feasible all possible combinations of: inputs state, distance type, projection dimension, basis family, and kernel family. However, the user is allowed to modify the solution space by imposing a system of constraints. This is achieved through the `ctrains` argument, which should be provided as a `list`. Below an example.

```
# generating input and output data
set.seed(100)
n.tr <- 32
sIn <- expand.grid(x1 = seq(0,1,length = n.tr^(1/5)), x2 = seq(0,1,length = n.tr^(1/5)),
                  x3 = seq(0,1,length = n.tr^(1/5)), x4 = seq(0,1,length = n.tr^(1/5)),
                  x5 = seq(0,1,length = n.tr^(1/5)))
fIn <- list(f1 = matrix(runif(n.tr*10), ncol = 10), f2 = matrix(runif(n.tr*22), ncol = 22))
sOut <- fgpm_BB7(sIn, fIn, n.tr)

# setting up the constraints
myctr <- list(s_keepOn = c(1,2), # keep both scalar inputs always on
             f_keepOn = c(2), # keep f2 always active
             f_disTypes = list("2" = c("L2_byindex")), # only use L2_byindex distance for f2
             f_fixDims = matrix(c(2,4), ncol = 1), # f2 should be projected onto a space of dimension 4
             f_maxDims = matrix(c(1,5), ncol = 1), # f1 should be projected onto a space of dimension max 5
             f_basTypes = list("1" = c("B-splines")), # only use B-splines projection for f1
             kerTypes = c("matern5_2", "gauss")) # test only Matern 5/2 and Gaussian kernels

# calling the funGp factory with specific constraints
xm <- fgpm_factory(sIn = sIn, fIn = fIn, sOut = sOut, ctrains = myctr) # (~15 seconds)
```

This call to the factory will exclusively explore models that fulfill the constraints passed through `ctrains`. This can be verified by inspecting the `@log.success@sols` slot of the `Xfgpm` object returned by `fgpm_factory`.

```
# checking log of some successfully built models
cbind(xm@log.success@sols, "Q2" = xm@log.success@fitness)
```

R output:

	State_X1	State_X2	State_X3	State_X4	State_X5	State_F1	Distance_F1	Dim_F1	Prj_basis_F1	State_F2	Distance_F2	Dim_F2	Prj_basis_F2	Kernel	Q2
1	On	On	Off	On	Off	Off	--	--	--	On	L2_byindex	4	B-splines	matern5_2	0.77
2	On	On	Off	On	Off	On	L2_byindex	3	B-splines	On	L2_byindex	4	B-splines	matern5_2	0.74
3	On	On	Off	On	On	Off	--	--	--	On	L2_byindex	4	B-splines	matern5_2	0.64
4	On	On	Off	On	On	On	L2_byindex	3	B-splines	On	L2_byindex	4	B-splines	matern5_2	0.47
5	On	On	On	On	Off	Off	--	--	--	On	L2_byindex	4	B-splines	matern5_2	0.43
6	On	On	Off	Off	On	Off	--	--	--	On	L2_byindex	4	B-splines	gauss	0.43
7	On	On	Off	Off	Off	Off	--	--	--	On	L2_byindex	4	B-splines	matern5_2	0.42
8	On	On	On	On	On	On	L2_byindex	1	B-splines	On	L2_byindex	4	B-splines	matern5_2	0.38
9	On	On	On	On	On	Off	--	--	--	On	L2_byindex	4	B-splines	matern5_2	0.27
10	On	On	On	On	On	On	Off	--	--	On	L2_byindex	4	PCA	gauss	0.26
11	On	On	On	On	On	Off	--	--	--	On	L2_byindex	4	PCA	matern5_2	0.12
12	On	On	On	Off	On	Off	--	--	--	On	L2_byindex	4	B-splines	matern5_2	0.10
13	On	On	Off	Off	On	On	L2_byindex	1	B-splines	On	L2_byindex	4	B-splines	gauss	0.02
14	On	On	Off	Off	On	On	L2_byindex	2	B-splines	On	L2_byindex	4	B-splines	matern5_2	-0.05
15	On	On	Off	On	Off	Off	--	--	--	On	L2_byindex	4	PCA	matern5_2	-0.07
16	On	On	Off	On	Off	Off	--	--	--	On	L2_byindex	4	PCA	gauss	-0.10
17	On	On	Off	Off	On	On	L2_byindex	3	B-splines	On	L2_byindex	4	B-splines	matern5_2	-0.16
18	On	On	On	Off	On	On	L2_bygroup	3	B-splines	On	L2_byindex	4	PCA	gauss	-0.27

• Time based stopping condition

The basic stopping condition for any ant colony based algorithm is the number of iterations. This type of stopping condition is often useful during the development stage of the algorithm.

However, in the wild it is hard to know in advance which number of iterations will be suitable for the problem at hand, and even if one had an idea, it would still be difficult to estimate how much processing time that would suppose. In practice we recommend to use instead a time based stopping condition. It works by defining a time budget for structural optimization, and then letting the heuristic run until the budget is exhausted. This possibility has been implemented in `fgpm_factory`, and is accessible through the `time.lim` argument.

```
# setting up a sufficiently large number of iterations
mysup <- list(n.iter = 2000)

# defining time limit
mytlim <- 60

# calling the funGp factory with time based stopping condition
xm <- fgpm_factory(sIn = sIn, fIn = fIn, sOut = sOut, setup = mysup, time.lim = mytlim)

R output:
** Time limit reached, exploration stopped after 60.01 seconds. # 163 iterations done
```

When using the time based stopping condition, the number of iterations should be set sufficiently large so that it does not cause a premature stop of the exploration. The argument `time.lim` should always be provided in seconds. Once the time limit is reached, the algorithm will attempt to stop as soon as possible, however, the ongoing training process of a model will never be interrupted. Thus, the actual processing time will normally exceed the specified time budget for a bit. This discrepancy might be more noticeable for problems involving heavier model configurations with larger number of inputs or a larger amount of data.

• Further exploring the `Xfgpm` object

After checking different things that can be done through a `fgpm_factory` call, it is good time to dedicate some attention to the information contained in the object delivered by the function. The object is of class `Xfgpm`, which includes diverse information about the selected model and also about the model selection process carried on. Below a list of the slots of the object with a short description of each.

Selected model

- **@model**: selected model delivered by the `fgpm` function.
- **@structure**: `data.frame` with the selected structural configuration.
- **@stat** and **@fitness**: type and value of the performance statistic used for the optimization of the structural parameters. Currently, the type of performance statistic can be either Q_{loocv}^2 or Q_{hout}^2 (see Sections 4.1.1 and 4.1.2 for details on these measures).

Record of explored models

- **@log.success**: object of class `antsLog` with the structure, function calls and performance statistic of all models successfully made during the optimization, organized in decreasing order of performance.
- **@log.crashes**: object of class `antsLog` with the structure and function calls of all models whose `fgpm` function call crashed.

Exploration extent

- **@n.solspace**: total number of structural configurations that could be made, based on the specified solution space.
- **@n.explored**: total number of structural configurations successfully built and evaluated during the exploration.

Further information

- **@details**: a **list** containing: (i) the set of heuristic parameters used; and (ii) the series of fitness vectors over the iterations of the heuristic.
- **@factoryCall**: a reminder of the expression used in the **fgpm_factory** call.

By conducting the structural optimization through **fgpm_factory**, one obtains not only one but a set of high quality models. Those are accessible through the **@log.success@sols** and **@log.success@args** slots. The former contains a data frame with all the levels of structural parameters selected for each explored model. This data structure could be useful to make a posterior analysis on patterns that lead to high quality models. The **@log.success@args** slot contains exactly the same information, but in a format that allows the easy reconstruction of any of the explored models. We illustrate this possibility with the following example. We start by performing a structural optimization.

```
# generating input and output data
set.seed(100)
n.tr <- 32
sIn <- expand.grid(x1 = seq(0,1,length = n.tr^(1/5)), x2 = seq(0,1,length = n.tr^(1/5)),
                  x3 = seq(0,1,length = n.tr^(1/5)), x4 = seq(0,1,length = n.tr^(1/5)),
                  x5 = seq(0,1,length = n.tr^(1/5)))
fIn <- list(f1 = matrix(runif(n.tr*10), ncol = 10), f2 = matrix(runif(n.tr*22), ncol = 22))
sOut <- fgpm_BB7(sIn, fIn, n.tr)

# calling the funGp factory
xm <- fgpm_factory(sIn = sIn, fIn = fIn, sOut = sOut) # (~10 seconds)
```

After some time we update our dataset. Now we have 243 points instead of 32. Then, we rebuild the best three models using the new data.

```
# generating new data
n.tr <- 243 # more points!
sIn <- expand.grid(x1 = seq(0,1,length = n.tr^(1/5)), x2 = seq(0,1,length = n.tr^(1/5)),
                  x3 = seq(0,1,length = n.tr^(1/5)), x4 = seq(0,1,length = n.tr^(1/5)),
                  x5 = seq(0,1,length = n.tr^(1/5)))
fIn <- list(f1 = matrix(runif(n.tr*10), ncol = 10), f2 = matrix(runif(n.tr*22), ncol = 22))
sOut <- fgpm_BB7(sIn, fIn, n.tr)

# re-building the three best models based on the new data (one model at a time)
# m1 <- eval(parse(text = xm$log.success@args[[1]]@string)[[1]])
# m2 <- eval(parse(text = xm$log.success@args[[2]]@string)[[1]])
# m3 <- eval(parse(text = xm$log.success@args[[3]]@string)[[1]])

# re-building the three best models based on the new data (compact code with all 3 calls)
modStack <- lapply(1:3, function(i) eval(parse(text = xm$log.success@args[[i]]@string)[[1]]))
```

Finally, we use each model for prediction. Here, the **format4pred** function will generate a list with the scalar and functional inputs to use for each model. If any of the two types of inputs is not present in the model, **format4pred** will set it to **NULL**, which will be properly interpreted by the **fgpm** function.

```

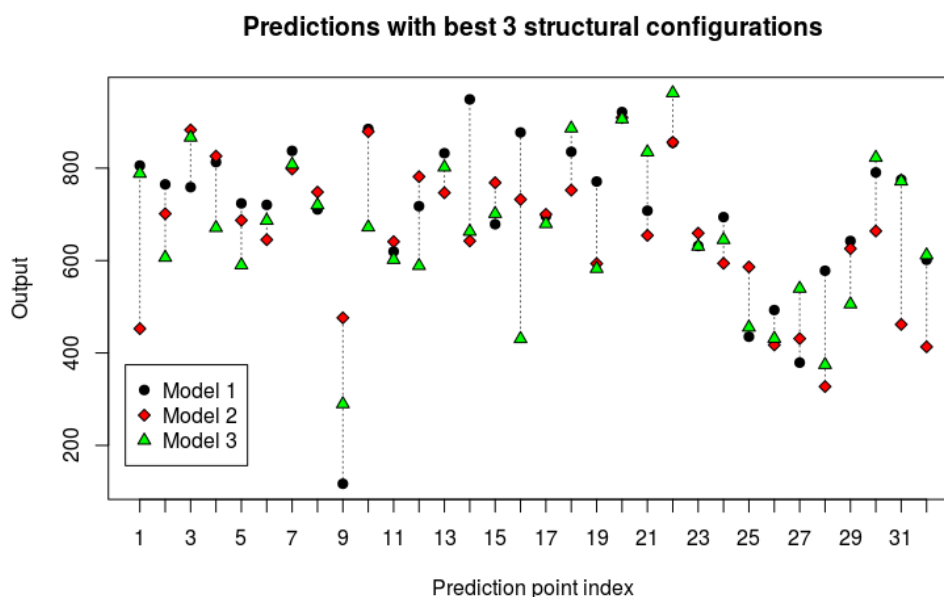
# extracting the fgpm arguments of the three best models
argStack <- xm$log.success@args[1:3]

# generating input data for prediction
n.pr <- 32
sIn.pr <- expand.grid(x1 = seq(0,1,length = n.pr^(1/5)), x2 = seq(0,1,length = n.pr^(1/5)),
                     x3 = seq(0,1,length = n.pr^(1/5)), x4 = seq(0,1,length = n.pr^(1/5)),
                     x5 = seq(0,1,length = n.pr^(1/5)))
fIn.pr <- list(f1 = matrix(runif(n.pr*10), ncol = 10), matrix(runif(n.pr*22), ncol = 22))

# making predictions based on the three best models (compact code with all 3 calls)
preds <- do.call(cbind, Map(function(model, args) {
  in4matted <- format4pred(sIn.pr = sIn.pr, fIn.pr = fIn.pr, args)
  predict(model, sIn.pr = in4matted$sIn.pr, fIn.pr = in4matted$fIn.pr)$mean
}, modStack, argStack))

# plotting predictions made by the three models
require(plyr) # for conciseness
plot(1, xlim = c(1,nrow(preds)), ylim = range(preds), xaxt = "n",
     xlab = "Prediction point index", ylab = "Output",
     main = "Predictions with best 3 structural configurations")
axis(1, 1:nrow(preds))
l_ply(seq_len(n.pr), function(i) lines(rep(i,2), range(preds[i,1:3]), col = "grey35", lty = 3))
points(preds[,1], pch = 21, bg = "black")
points(preds[,2], pch = 23, bg = "red")
points(preds[,3], pch = 24, bg = "green")
legend("bottomleft", legend = c("Model 1", "Model 2", "Model 3"),
      pch = c(21, 23, 24), pt.bg = c("black", "red", "green"), inset = c(.02,.08))

```



4.4 Parallelization in funGp

Sections 4.1, 4.2 and 4.3 made a good description of **funGp** from the perspective of functionality. This last section focuses on efficiency. Both, the **fgpm** and **fgpm_factory** functions have been equipped with the ability to exploit the existence of parallel environments. Below we explain how to use this feature.

4.4.1 Parallelized hyperparameters optimization

Let us start with the `fgpm` function, used to create regression models (see [Section 4.1.1](#)). In `funGp`, the selection of the hyperparameters of the model is made by likelihood maximization. For Gaussian processes, this corresponds to a nonlinear optimization problem, sometimes strongly affected by the selection of the starting points. A common way to deal with this issue is to start the optimization multiple times from different points, which prevents the stagnation in local optima. This can be requested to `fgpm` through the argument `n.starts`, which should be assigned an integer value corresponding to the number of starting points to use. Below an example using 10 starting points.

```
# generating input data for training
set.seed(100)
n.tr <- 243
sIn <- expand.grid(x1 = seq(0,1,length = n.tr^(1/5)), x2 = seq(0,1,length = n.tr^(1/5)),
                  x3 = seq(0,1,length = n.tr^(1/5)), x4 = seq(0,1,length = n.tr^(1/5)),
                  x5 = seq(0,1,length = n.tr^(1/5)))
fIn <- list(f1 = matrix(runif(n.tr*10), ncol = 10), f2 = matrix(runif(n.tr*22), ncol = 22))

# generating output data for training
sOut <- fgpm_BB7(sIn, fIn, n.tr)

# calling fgpm with multistart in sequence
m1 <- fgpm(sIn = sIn, fIn = fIn, sOut = sOut, n.starts = 10) # (~22 seconds)
```

Since each starting point triggers an independent optimization process, the requested task can be performed in parallel. To do so, the user must define a parallel processing cluster and then pass it to `fgpm` through the `par.clust` argument.

```
# calling fgpm with multistart in parallel
cl <- parallel::makeCluster(3)
m1 <- fgpm(sIn = sIn, fIn = fIn, sOut = sOut, n.starts = 10, par.clust = cl) # (~14 seconds)
parallel::stopCluster(cl)
```

As a good practice, the cluster must be stopped right after finishing the requested task in order to prevent memory issues.

Remark: in order to provide progress bars for the monitoring of time consuming processes ran in parallel, `funGp` relies on the `doFuture` [108] and `future` [109] R packages. Unfortunately, under this setting, parallel processes suddenly interrupted tend to leave corrupt connections that will show up as an error next time you try to perform the parallelized task. To make it clear, if you launch `fgpm` in parallel and you stop the process by hand, before it ends, and then you try to repeat the call in parallel, you may likely find an error indicating that ... **the connection to the worker is corrupt....** If that happens to you, the following workaround will help to regain control of parallel processing. Once you get the error, repeat the function call using a different number of nodes. For instance, let us assume that you had run with 3 nodes in the call that produced the error. We can make the new function call, for instance with 2 nodes.

```
# repeating the call with different number of nodes
cl <- parallel::makeCluster(2)
m1 <- fgpm(sIn = sIn, fIn = fIn, sOut = sOut, n.starts = 10, par.clust = cl)
parallel::stopCluster(cl)
```

There is no need to let this process become complete, you can stop it by hand a couple seconds after making the function call. That is it. Now you can launch again the process

in parallel with the number of nodes that you were originally using. We acknowledge that this is more a trick than an ideal way to resolve this types of issues. However, this problem is originated outside `funGp`, which limits our control over it. We find the approach shown above a pragmatic solution for most users. We will remain attentive in case it appears a more elegant solution to this problem. All this discussion also applies for parallelized calls to `fgpm_factory`, which will be discussed in the next section.

4.4.2 Parallelized model selection

Parallelization is also present in the model factory. Each ant in our heuristic algorithm represents a structural configuration, and eventually translates into a regression model. Each ant influence on the decisions made by the others since they share a common decision network and all of them affect the pheromone load in the links. Nonetheless, each time all the ants of one iteration complete a model structure, each of the models is built and evaluated for performance in an independent fashion. Thus, once all the structural configurations of one iteration are complete, the construction of the corresponding models is a task that can be performed in parallel. The way to do that is identical to how it is done in the `fgpm` function. For this, the user must define a parallel processing cluster and then pass it to `fgpm_factory` through the `par.clust` argument, as below.

```
# generating input and output data
set.seed(100)
n.tr <- 243
sIn <- expand.grid(x1 = seq(0,1,length = n.tr^(1/5)), x2 = seq(0,1,length = n.tr^(1/5)),
                  x3 = seq(0,1,length = n.tr^(1/5)), x4 = seq(0,1,length = n.tr^(1/5)),
                  x5 = seq(0,1,length = n.tr^(1/5)))
fIn <- list(f1 = matrix(runif(n.tr*10), ncol = 10), f2 = matrix(runif(n.tr*22), ncol = 22))
sOut <- fgpm_BB7(sIn, fIn, n.tr)

# calling fgpm_factory in parallel
cl <- parallel::makeCluster(3)
xm.par <- fgpm_factory(sIn = sIn, fIn = fIn, sOut = sOut, par.clust = cl) # (~200 seconds)
parallel::stopCluster(cl)
```

The advice given when explaining the parallelization in the `fgpm` function applies here; the cluster must be stopped right after finishing the requested task in order to prevent memory issues. In addition, we clarify that parallelized processing should be reserved for cases where each individual process (call to `fgpm`) takes a significant amount of time. If the call in sequence is already long due to the large number of processes it involves, but each process runs almost immediately, the benefit of parallelization might become null. Thus, we prescribe the use of this feature for problems where the evaluation of a single model takes several seconds or more. In such a context, parallelization will allow the evaluation of a larger number of structural configurations in the same amount of time.

Closing discussion

`funGp` started as a set of scripts enabling to include functional inputs in a regression model. What we present in this tutorial is that and much more. We have done our best to provide a powerful regression tool for all-level users. No impositions on the type of relationship between inputs and outputs, no need of pre-processing of the functional inputs, no need for complex data structures. You put the data and we put the power of the Gaussian process models in order to efficiently extract the underlying information from it. The more expertise

the user has in statistics and also in the usage of the package, the more it will be able to discover new possibilities and features. We make strong emphasis on the model selection functionality, which takes models' construction to a whole new level. Any regression package gives you a model in response for your data. Some packages return different types of models depending on your specifications. Not very often a package helps you to choose the good model, and this is what **funGp** does. During the implementation, we kept present at all time the need for efficiency, and we made an effort to make everything run fast and smooth. Parallelization in the **fgpm** and **fgpm_factory** functions is a valuable commodity in this regard. We envisage the extension of the package in multiple different aspects, and therefore, we made the implementations with scalability in mind. All the structural parameters are modifiable in order to include additional levels or even other structural parameters than those currently available. The heuristic model selection algorithm was also designed to be easily adaptable to this type of extension. Going further, the **fgpm_factory** function was structured in such a way that other model selection methods could be added later. Being **funGp** a piece of open source, we encourage the community to make contributions in any line found pertinent.

Acknowledgements

This study was conducted in the frame of the RISCOPE project, funded by the French Agence Nationale de la Recherche (ANR). We thank the ANR for this support. We are also grateful to Yves Deville from Alpestat for his advice on documentation of R packages and to Juliette Garcia from ENAC for her assistance on the stabilization of the Ant Colony algorithm for structural parameter optimization.

Chapter 5

Structural parameter optimization in the coastal flooding RISCOPE case

The chapter in brief

The previous three chapters addressed the introduction to the RISCOPE coastal flooding application, the development of an Ant Colony algorithm for the efficient optimization of the structural parameters of the metamodel, and the consolidation of the R package `funGp` [20], respectively. [Chapter 5](#) is a follow up to the RISCOPE case. The description given in [Chapter 2](#) for this application remains mostly valid, except for the three following aspects:

- 1) **Updated hydrodynamic code:** at the time of writing [Chapter 2](#), the target hydrodynamic code was in calibration. Thus, we used a simplified version of it which was quicker but less precise and detailed. At this time the target hydrodynamic code is complete.
- 2) **Significantly less observations:** the new hydrodynamic code takes much more time per computation (several hours to days) than the simplified version used in [Chapter 2](#) (around 20 seconds). Thus, current developments are much more limited in the amount of simulations that we can conduct. Up to now, we have completed a total of 135 simulations. The (functional) input coordinates for those simulations were selected by means of an adaption of Extreme Value Analysis (see e.g., [110]) to functional data. Such a methodology is intended to produce input conditions leading (more often) to significative flooding events, while preserving the main statistical properties of each input variable. We reserve the details of this methodology for future dissemination to the scientific community.
- 3) **Several output variables:** the new hydrodynamic code enables the extraction of diverse types of information of interest such as the maximum flooded area, the water height at surveillance points and coefficients of trafficability of critical roads. In this chapter we consider a total of 13 scalar outputs and we build a metamodel for each of them.

The Ant Colony algorithm proved its pertinency and effectiveness by finding high quality structural configurations for all the 13 outputs under analysis. In all cases, the selected configuration outperformed several others, including the default choices of using: (i) only a scalar representation of each functional input; (ii) the full set of scalar and functional representations of the inputs; and (iii) a scalar representation of each functional input, plus the functional representation of some key input variables. Even for the variables the most difficult to fit, our algorithm was able to find a configuration superior to the aforementioned alternatives. All the analysis was conducted using our R package `funGp` [20].

Contents

5.1	Introduction	103
5.2	The new hydrodynamic code	103
5.3	Structural parameter optimization	104
5.4	Conclusions	106

5.1 Introduction

This chapter is a brief update of the RISCOPE application based on the developments made in the previous chapters, and also on the evolution of the hydrodynamic computer code. The new hydrodynamic code allows us to retrieve a variety of useful information. Here we use our Ant Colony algorithm (see [Chapter 3](#)) for the structural parameter optimization of the metamodel built for 13 scalar output variables. Rather than constituting a document in the form of a research paper (as [Chapter 2](#) and [Chapter 6](#), for instance), this chapter is intended to concisely show for the first time in the manuscript the performance of our algorithm in the RISCOPE application.

5.2 The new hydrodynamic code

The hydrodynamic code used in [Chapter 2](#) received four variables with physical interpretability as inputs. Those were the tide (Td), atmospheric storm surge (Sg), significant wave height (Hs) and peak wave period (Tp). In addition to those, the new code receives the peak wave direction (Dp), the wind speed (U) and the wind direction (Du). Moreover, in [Chapter 2](#) we were considering a version of (Td) which had integrated the mean sea level (Msl). In the new analysis we consider both inputs separately. This makes a total of eight functional inputs for the new code. Each of them should be provided to the system in a time series format, similarly as in [Chapter 2](#). All the input time series should be of dimension 37. The code delivers several output variables, also in the form of time series and spatial maps. For now, we focus on scalar quantities representing those outputs. In particular, we consider the following list of scalar output variables:

- **Ysurf_max**: largest area flooded during the event, in m^2 ;
- **Ysurf_fin**: extension of land flooded at the end of the event, in m^2 ;
- **Yvol_fin**: total amount of water entered in land during the event, in m^3 ;
- **Ytr_ft1 and Ytr_fh1**: indices of trafficability for main road 1;
- **Ytr_ft2 and Ytr_fh2**: indices of trafficability for main road 2;
- **Ytr_ft3 and Ytr_fh3**: indices of trafficability for main road 3;
- **Yhcb_max1 and Yhcb_max2**: maximum water height at surveillance points 1 and 2;
- **Yhcb_fin1 and Yhcb_fin2**: water height at the end of the event at points 1 and 2.

The three main roads and the two surveillance points are shown in the aerial views of Gâvres displayed in [Figure 5.1](#).



Figure 5.1 – Aerial view of Gâvres: main roads and surveillance points.

5.3 Structural parameter optimization

For the purpose of metamodeling we are decomposing each functional input into a scalar and a functional representation, similarly to what we did in Sections 2.4.2.2 and 2.5.1 of Chapter 2. A shifted functional input is one whose scalar representation has been subtracted. This, with the aim of letting the algorithm the possibility to decide separately if either the scalar representation, the shifted functional representation, or both parts of an input should be kept active in the model. For Td we used the maximum value of the series as the scalar representation. For all the other inputs except Msl , we used the average value of the series. The Msl is the only input variable for which only the scalar representation is considered, and it comes from the fact that this input is almost always a constant function during a six hour event like the ones we are considering. This setup results in a total of 8 scalar inputs and 7 shifted functional inputs with the potential of being included in the metamodel. **In addition to the active scalar and functional inputs**, in this section we also use our Ant Colony algorithm to set:

- 1) the dimension reduction method to use for each input;
- 2) the projection dimension for each input;
- 3) the kernel function of the model;
- 4) the distance used to measure similarity between functional input coordinates within the kernel function.

For the sake of comparison, we also build the metamodel with some arbitrary structural configurations that one may choose in the absence of an algorithm like ours. Specifically, we consider metamodels with:

- a) all the scalar inputs, but no functional inputs active;
- b) all the scalar and functional inputs active;
- c) all the scalar inputs, plus the main functional inputs from the physical perspective. For this experiment we keep active the functional representations of Td , Sg , Hs and Tp .

For the remaining structural parameters of the benchmark configurations, we keep fixed:

- 1) the dimension reduction method for all functional inputs at B-splines;
- 2) the projection dimension for each input at 3;
- 3) the Matérn 5/2 kernel;
- 4) the $\|\cdot\|_{D,\theta_f}$ distance defined in (2.11) (Chapter 2), referred to as the **L2_bygroup** distance in funGp. We recall that this corresponds to the distance that uses a single length-scale coefficient per functional input, in contrast to the $\|\cdot\|_{S,\hat{\theta}_f}$ defined in (2.13), which uses as many length-scale coefficients as effective dimensions the input has.

For every output, we fitted a metamodel using each of the benchmark structural configurations, and we also ran our Ant Colony algorithm to pick an optimized one. In each case, we looked at the calibration plot and the Q^2_{hout} statistic, corresponding to the classical coefficient of determination R^2 for a test sample, i.e., for prediction residuals [77]. An example for the output variable Ysurf_max is given in Figure 5.2.

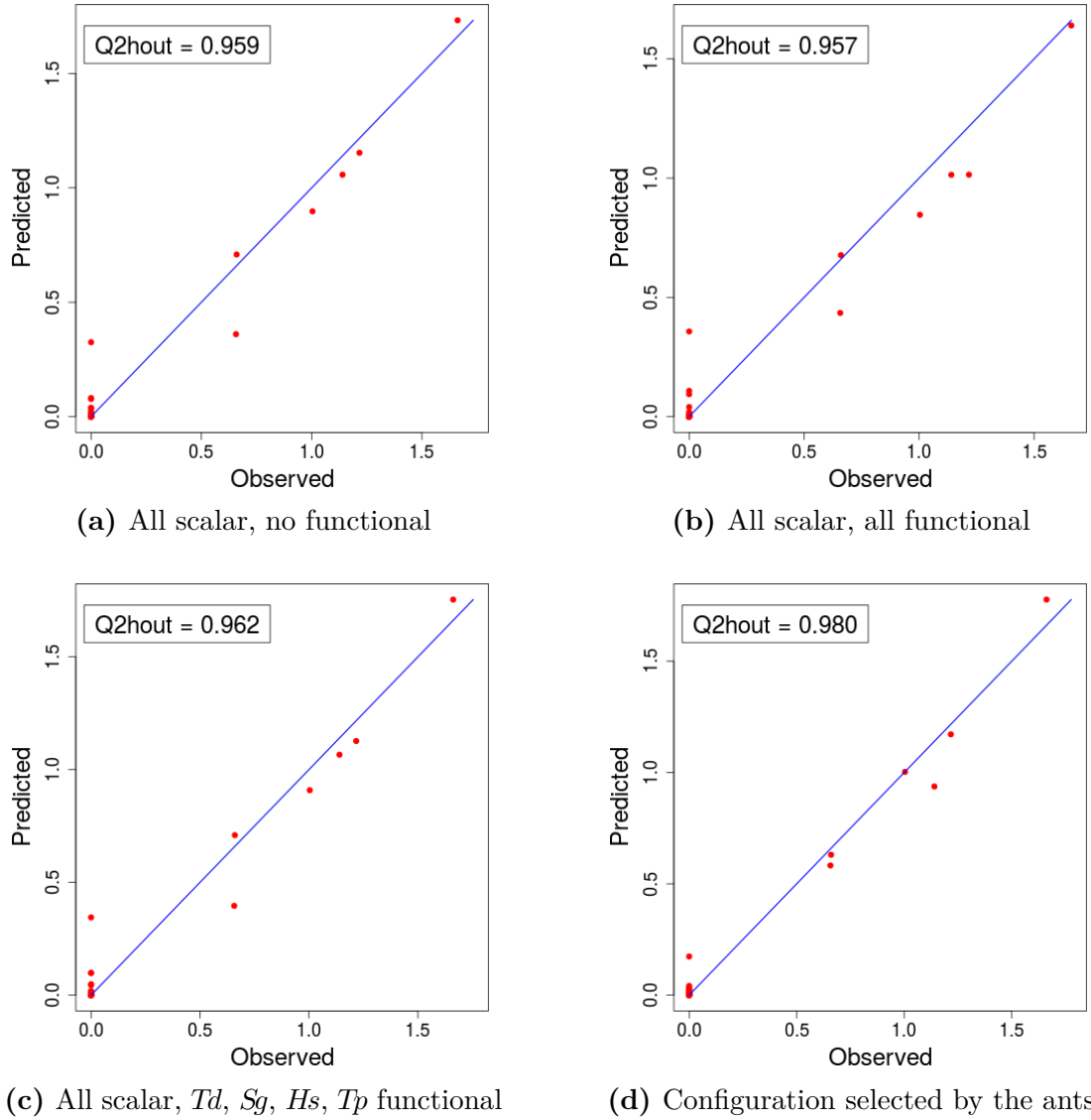


Figure 5.2 – Ants vs. 3 benchmark configurations for one output of the RISCOPE case.

Table 5.1 summarizes our experiment. In all cases we ran the Ant Colony algorithm with the default parameters loaded in funGp, except for the exploration coefficient q_0 (see Section 4.3.2, Chapter 4), which we set to 0.9 instead of 0.95 to allow more diversification in the solutions built by the ants. We let the algorithm run for 10 minutes in each instance.

Output ID	All scalar	All scalar + all functional	All scalar + some functional	Ants selection	Ants active scalar	Ants active functional
Ysurf_max	0,959	0,957	0,962	0,980	{1,2,3,5,6,8}	{2,4}
Ysurf_fin	0,942	0,916	0,946	0,981	{1,2,3,5,6,7}	{7}
Yvol_fin	0,929	0,927	0,938	0,961	{1,2,3,5,6}	{3,4,5,7}
Ytr_ft1	0,927	0,929	0,928	0,963	{1,2,3,5,7,8}	{2}
Ytr_ft2	0,950	0,958	0,957	0,977	{1,3,5,7,8}	{2,5,6,7}
Ytr_ft3	0,533	0,655	0,469	0,764	{1,2,3,4,6,8}	{3,5,6}
Ytr_fh1	0,961	0,954	0,960	0,965	{1,2,3,5,7,8}	{2,4,5,6,7}
Ytr_fh2	0,941	0,862	0,959	0,974	{1,2,3,4,5,6,7,8}	{2}
Ytr_fh3	0,791	0,806	0,828	0,887	{1,3,4,5,7,8}	{2,4,5}
Yhcb_max1	0,576	0,499	0,443	0,900	{1,3,4,5,6}	{2,3,6,7,8}
Yhcb_max2	0,897	0,895	0,880	0,939	{1,3,4,6,7,8}	{2}
Yhcb_fin1	0,424	0,274	0,320	0,625	{1,2,4,6,7,8}	{3}
Yhcb_fin2	0,893	0,883	0,890	0,926	{1,3,5,6,7}	{2,4}

Table 5.1 – Comparison of Ant Colony algorithm to three default benchmark structural configurations for 13 metamodels required in the RISCOPE coastal flooding application. The Q_{hout}^2 obtained by each benchmark configuration, as well as that for the configuration delivered by our algorithm are reported in the table. The darker colored cells are those related to a better prediction quality. The active scalar and functional inputs kept active in the configuration delivered by the Ant Colony algorithm are displayed as well in the two columns at the right side of the table. The numbers 1 to 8 correspond to the inputs Msl , Td , Sg , Hs , Tp , Dp , U , Du , in that order.

The configuration found by our Ant Colony algorithm outperformed the three benchmark configurations in all the instances. Even in the cases where the output was relatively hard to fit (e.g., for Ytr_ft3), the algorithm was able to build a model of considerably higher Q_{hout}^2 . The improvement should not be undervalued in the cases where the other configurations give similar Q_{hout}^2 values (e.g., for Ytr_fh2). On the one hand, the algorithm is already coded and made available in an R package. The only price we paid in order to obtain a better configuration was just 10 minutes of computation, which in most metamodeling contexts (involving numerical simulations lasting for hours and days) is negligible. On the other hand, RISCOPE deals with an application involving human lives at risk. Any improvement we could have in prediction, is therefore highly valuable.

5.4 Conclusions

This chapter presented the application of the Ant Colony algorithm for the calibration of the structural configuration of 13 different metamodels required in the RISCOPE case study. We let the algorithm run for a relatively short period of just 10 minutes. This time was enough for it to find a structural configuration of relatively high prediction quality in all cases. The selected configuration outperformed 3 benchmark structural configurations that one may choose in the absence of an automated selection tool like this. The results obtained prove the robustness of the algorithm to changes in the input-output relationship. Moreover, the instances resolved in this chapter involved 8 scalar and 7 functional inputs, which shows the suitability of the algorithm for problems of larger size than those used for illustration in Chapters 3 and 4.

Chapter 6

Asymptotic properties of the maximum likelihood and cross validation estimators for transformed Gaussian processes

The chapter in brief

This chapter presents a theoretical study which is independent from the RISCOPE coastal flooding application. The chapter is an exact copy of our article [23], published in *Electronic Journal of Statistics*. There, we study the asymptotics of the maximum likelihood (ML) and cross validation (CV) estimators for the covariance parameters of a non-Gaussian process. We are motivated by the fact that many real applications involve output variables that present certain markedly non-Gaussian characteristics such as nonnegativity (e.g., [24]) and monotonicity (e.g., [25]), but they are still modeled as Gaussian without further considerations. This gives rise to the questions of: what can we expect from the estimation of the covariance parameters if we model a non-Gaussian process as Gaussian? In case of suspicion of non-Gaussianity would it be beneficial to apply some transformation to the output before implementing the Gaussian process model? Chapter 6 addresses the first question and gives the bases to undertake the second one.

In particular, we consider the case where the non-Gaussian process results from an unknown non-linear transformation of a Gaussian process. We further assume that the transformation is not modeled or estimated. We show that the ML and CV estimators are consistent and asymptotically normal, although they are defined as if the process was Gaussian. Our results can thus be interpreted as a robustness of (Gaussian) ML and CV towards non-Gaussianity. This study could be extended to the case where the transformation parameters are estimated along with the covariance parameters. We expect such an approach to be beneficial at least for taking into account non-Gaussian characteristics of the process of interest such as the aforementioned nonnegativity or more generally speaking, boundary constraints.

Contents

6.1	Introduction	109
6.2	General properties of transformed Gaussian processes	111
6.3	Two main technical results	113
6.3.1	Transformed Gaussian process framework	113
6.3.2	Bounds on the elements of inverse covariance matrices	113
6.3.3	CLT for quadratic forms of transformed Gaussian processes	114
6.4	Estimation of a single variance parameter	115
6.5	General covariance	116
6.5.1	Framework	116
6.5.2	Maximum Likelihood	117
6.5.3	Cross Validation	120
6.5.4	Joint asymptotic normality	122
6.6	Discussions of extensions	123
6.6.1	Nugget effect	123
6.6.2	Unknown mean	124
6.7	Illustration	124
6.8	Conclusion	128
Appendix 6.A	Proofs	129
6.A.1	Technical results	129
6.A.2	Proofs of the main results	135
	Proof of Lemma 1	135
	Proof of Lemma 2	136
	Proof of Theorem 1	136
	Proof of Theorem 2	138
	Proof of Theorem 3	140
	Proof of Theorem 4	141
	Proof of Theorem 5	142
	Proof of Theorem 6	142
	Proof of Lemma 4	143
	Proof of Theorem 7	145
	Proof of Proposition 1	146
	Proof of Corollary 3	146

Abstract

The asymptotic analysis of covariance parameter estimation of Gaussian processes has been subject to intensive investigation. However, this asymptotic analysis is very scarce for non-Gaussian processes. In this paper, we study a class of non-Gaussian processes obtained by regular non-linear transformations of Gaussian processes. We provide the increasing-domain asymptotic properties of the (Gaussian) maximum likelihood and cross validation estimators of the covariance parameters of a non-Gaussian process of this class. We show that these estimators are consistent and asymptotically normal, although they are defined as if the process was Gaussian. They do not need to model or estimate the non-linear transformation. Our results can thus be interpreted as a robustness of (Gaussian) maximum likelihood and cross validation towards non-Gaussianity. Our proofs rely on two technical results that are of independent interest for the increasing-domain asymptotic literature of spatial processes. First, we show that, under mild assumptions, coefficients of inverses of large covariance matrices decay at an inverse polynomial rate as a function of the corresponding observation location distances. Second, we provide a general central limit theorem for quadratic forms obtained from transformed Gaussian processes. Finally, our asymptotic results are illustrated by numerical simulations.

Keywords: covariance parameters, asymptotic normality, consistency, weak dependence, random fields, increasing-domain asymptotics

6.1 Introduction

Kriging [111, 112] consists of inferring the values of a (Gaussian) random field given observations at a finite set of points. It has become a popular method for a large range of applications, such as geostatistics [113], numerical code approximation [114, 115, 116], calibration [117, 118], global optimization [119], and machine learning [112].

When considering a Gaussian process, one has to deal with the estimation of its covariance function. Usually, it is assumed that the covariance function belongs to a given parametric family (see [120] for a review of classical families). In this case, the estimation boils down to estimating the corresponding covariance parameters. Nowadays, the main estimation techniques are based on maximum likelihood [111, 112], cross-validation [121, 60, 122, 123] and variation estimators [124, 125, 126].

The asymptotic properties of estimators of the covariance parameters have been widely studied in the two following frameworks. The fixed-domain asymptotic framework, sometimes called infill asymptotics [111, 127], corresponds to the case where more and more data

are observed in some fixed bounded sampling domain. The increasing-domain asymptotic framework corresponds to the case where the sampling domain increases with the number of observed data.

Under fixed-domain asymptotics, and particularly in low dimensional settings, not all covariance parameters can be estimated consistently (see [128, 111]). Hence, the distinction is made between microergodic and non-microergodic covariance parameters [128, 111]. Although non-microergodic parameters cannot be estimated consistently, they have an asymptotically negligible impact on prediction [129, 130, 131, 132]. There is, however, a fair amount of literature on the consistent estimation of microergodic parameters (see for instance [132, 133, 134, 135, 136, 137]).

This paper focuses on the increasing-domain asymptotic framework. Indeed, generally speaking, increasing-domain asymptotic results hold for significantly more general families of covariance functions than fixed-domain ones. Under increasing-domain asymptotics, the maximum likelihood and cross validation estimators of the covariance parameters are consistent and asymptotically normal under mild regularity conditions [138, 139, 122, 140].

All the asymptotic results discussed above are based on the assumption that the data come from a Gaussian random field. This assumption is indeed theoretically convenient but might be unrealistic for real applications. When the data stem from a non-Gaussian random field, it is still relevant to estimate the covariance function of this random field. Hence, it would be valuable to extend the asymptotic results discussed above to the problem of estimating the covariance parameters of a non-Gaussian random field.

In this paper, we provide such an extension, in the special case where the non-Gaussian random field is a deterministic (unknown) transformation of a Gaussian random field. Models of transformed Gaussian random fields have been used extensively in practice (for example in [141, 142, 143, 144]).

We provide various asymptotic results, under reasonable regularity assumptions. In particular, our assumptions on the transformation function are mild. For most of our results, only sub-exponentiality is required.

We prove that applying the (Gaussian) maximum likelihood estimator to data from a transformed Gaussian random field yields a consistent and asymptotically normal estimator of the covariance parameters of the transformed random field. This (Gaussian) maximum likelihood estimator corresponds to what would typically be done in practice when applying a Gaussian process model to a non-Gaussian spatial process. This estimator does not need to know the existence of the non-linear transformation function and is not based on the exact density of the non-Gaussian data. We refer to Remark 5 for further details and discussion on this point.

We then obtain the same consistency and asymptotic normality result when considering a cross validation estimator. In addition, we establish the joint asymptotic normality of both these estimators, which provides the asymptotic distribution of a large family of aggregated estimators. Our asymptotic results on maximum likelihood and cross validation are illustrated by numerical simulations.

To the best of our knowledge, our results (Theorems 3, 4, 5, 6 and 7) provide the first increasing-domain asymptotic analysis of Gaussian maximum likelihood and cross validation for non-Gaussian random fields. Our proofs intensively rely on Theorems 1 and 2. Theorem 1 shows that the components of inverse covariance matrices are bounded by inverse polynomial functions of the corresponding distance between observation locations. Theorem 2 provides a generic central limit theorem for quadratic forms constructed from transformed Gaussian processes. These two theorems have an interest in themselves.

The rest of the paper is organized as follows. In Section 6.2, general properties of transformed Gaussian processes are provided. In Section 6.3, Theorems 1 and 2 are stated. In Section 6.4, an application of these two theorems is given to the case of estimating a single variance parameter. In Section 6.5, the consistency and asymptotic normality results for general covariance parameters are given. The joint asymptotic normality result is also given in this section. Section 6.6 discusses some extensions of the results of the previous sections. The simulation results are provided in Section 6.7. All the proofs are provided in the appendix.

6.2 General properties of transformed Gaussian processes

In applications, the use of Gaussian process models may be too restrictive. One possibility for obtaining larger and more flexible classes of random fields is to consider transformations of Gaussian processes. In this section, we now introduce the family of transformed Gaussian processes that we will study asymptotically in this paper. This family is determined by regularity conditions on the covariance function of the original Gaussian process and on the transformation function.

Let us first introduce some notation. Throughout the paper, $C_{\inf} > 0$ (resp. $C_{\sup} < \infty$) denotes a generic strictly positive (resp. finite) constant. This constant never depends on the number of observations n , or on the covariance parameters (see Section 6.5), but is allowed to depend on other variables. We mention these dependences explicitly in cases of ambiguity. The values of $C_{\inf}$ and $C_{\sup}$ may change across different occurrences.

For a vector x of dimension d we let $|x| = \max_{i=1,\dots,d} |x_i|$. Further, the Euclidean and operator norms are denoted by $\|x\|$ and by $\|M\|_{op} = \sup\{\|Mx\| : \|x\| \leq 1\}$, for any matrix M . We let $\lambda_1(B) \geq \dots \geq \lambda_r(B)$ be the r eigenvalues of a $r \times r$ symmetric matrix B . We let $\rho_1(B) \geq \dots \geq \rho_r(B) \geq 0$ be the r singular values of a $r \times r$ matrix B . We let $\mathbb{N}$ be the set of non-zero natural numbers.

Further, we define the Fourier transform of a function $h : \mathbb{R}^d \rightarrow \mathbb{R}$ by

$$\hat{h}(f) = (2\pi)^{-d} \int_{\mathbb{R}^d} h(t) e^{-if^\top t} dt,$$

where $i^2 = -1$, for $f \in \mathbb{R}^d$. When mentioning the Fourier transform of a function h , we implicitly assume that Fourier inversion holds, that is $\hat{h}$ is summable and, for $s \in \mathbb{R}^d$, $h(s) = \int_{\mathbb{R}^d} \hat{h}(f) e^{is^\top f} df$.

For a sequence of observation locations, the next condition ensures that a fixed distance between any two observation locations exists. This condition is classical [122, 145].

Condition 1. *We say that a sequence of observation locations, $(x_i)_{i \in \mathbb{N}}$, $x_i \in \mathbb{R}^d$, is asymptotically well-separated if we have $\inf_{i,j \in \mathbb{N}, i \neq j} |x_i - x_j| > 0$.*

The next condition on a stationary covariance function is classical under increasing-domain asymptotics. This condition provides asymptotic decorrelation for pairs of distant observation locations and implies that covariance matrices are asymptotically well-conditioned when a minimal distance between any two distinct observation locations exists [138, 122].

Condition 2. *We say that a stationary covariance function k on $\mathbb{R}^d$ is sub-exponential and asymptotically positive if:*

i) C_{sup} and C_{inf} exist such that, for all $s \in \mathbb{R}^d$, we have

$$|k(s)| \leq C_{\text{sup}} \exp(-C_{\text{inf}}|s|); \quad (6.1)$$

ii) For any sequence $(x_i)_{i \in \mathbb{N}}$ satisfying Condition 1, we have $\inf_{n \in \mathbb{N}} \lambda_n(\Sigma) > 0$, where Σ is the $n \times n$ matrix $(k(x_i - x_j))_{i,j=1,\dots,n}$.

In Condition 2, we remark that $k : \mathbb{R}^d \rightarrow \mathbb{R}$ is called a stationary covariance function in the sense that $(x_1, x_2) \rightarrow k(x_1 - x_2)$ is a covariance function. We use this slight language abuse for convenience.

We also remark that, when non-transformed Gaussian processes are considered, a polynomial decay of the covariance function in Condition 2 i) is sufficient to obtain asymptotic results [122, 146]. Here an exponential decay is needed in the proofs to deal with the non-Gaussian case. Nevertheless, most classical covariance functions satisfy inequality (6.1). In particular, the Matérn covariance functions [111], the generalized Wendland covariance functions [147] and some of the power exponential covariance functions [115] satisfy inequality (6.1). On the other hand, for instance, the covariance functions in the Cauchy class [148] do not satisfy inequality (6.1).

When considering a transformed Gaussian process, we will consider a transformation satisfying the following regularity condition, which enables us to subsequently obtain regularity conditions on the covariance function of the transformed Gaussian process.

Condition 3. Let $F : \mathbb{R} \rightarrow \mathbb{R}$ be a fixed non-constant continuously differentiable function, with derivative F' . We say that F is sub-exponential and non-decreasing if:

i) For all $t \in \mathbb{R}$, we have $|F(t)| \leq C_{\text{sup}} \exp(C_{\text{sup}}|t|)$ and $|F'(t)| \leq C_{\text{sup}} \exp(C_{\text{sup}}|t|)$;

ii) The function F is non-decreasing on $\mathbb{R}$.

Regarding Condition 3 ii), we point out that many transformations of Gaussian processes considered in the literature are indeed non-decreasing, for instance the Tukey's g -and- h transformation in [142] and the exponential transformation for log-Gaussian processes. Furthermore, Condition 3 ii) need not always be assumed for the results of the present paper to hold, see Remarks 4 and 9.

In the following lemma, we show that the covariance function of a transformed Gaussian process satisfies Condition 2, when Conditions 2 and 3 are satisfied, for the original process and for the transformation.

Lemma 1. Assume that the stationary covariance function k satisfies Condition 2 and that the transformation F satisfies Condition 3. Let X be a zero-mean Gaussian process with covariance function k and let k' be the stationary covariance function of $F(X(\cdot))$. Then, k' satisfies Condition 2.

In the next lemma, we show that we can replace the condition of an increasing transformation by the condition of a monomial transformation of even degree (with an additive constant).

Lemma 2. If a covariance function k satisfies Condition 2 i) and if the Fourier transform $\hat{k}$ of k is strictly positive on $\mathbb{R}^d$, then k satisfies Condition 2 ii). Furthermore, in this case, in Lemma 1, Condition 3 ii) can be replaced by the condition $F(x) = x^{2r} + u$ for $r \in \mathbb{N}$, $u \in \mathbb{R}$ and $x \in \mathbb{R}$.

6.3 Two main technical results

6.3.1 Transformed Gaussian process framework

In the rest of the paper, we will consider an unobserved Gaussian process Z on $\mathbb{R}^d$ with $d \in \mathbb{N}$ fixed. Assume that Z has zero-mean and stationary covariance function k_Z . We assume throughout that k_Z satisfies Condition 2.

We consider a fixed transformation function T satisfying Condition 3. We assume that we observe the transformed Gaussian process Y , defined by $Y(s) = T(Z(s))$ for any $s \in \mathbb{R}^d$.

We assume throughout that the random field Y has zero-mean. We remark that, for a non-linear transformation $F : \mathbb{R} \rightarrow \mathbb{R}$, for $s \in \mathbb{R}^d$, the fact that $Z(s)$ has zero-mean does not imply that the random variable $F(Z(s))$ has zero-mean. Hence, we implicitly assume that T is of the form $F - \mathbb{E}[F(Z(x))]$, where F satisfies Condition 3 and $x \in \mathbb{R}^d$ is arbitrary. Note that $\mathbb{E}[F(Z(x))]$ is constant in x by stationarity and that, if F satisfies Condition 3 or the condition specified in Lemma 2, then $F - \mathbb{E}[F(Z(x))]$ also satisfies these conditions. We remark that the assumption of zero-mean (or equivalently of a known mean function for Y) is common in the literature. We provide a further discussion of this assumption in Section 6.6.2.

We let k_Y be the covariance function of Y . We remark that, from Lemma 1 (applied with $F = T$ and $X = Z$ and thus $k' = k_Y$), k_Y also satisfies Condition 2.

We let $(s_i)_{i \in \mathbb{N}}$ be the sequence of observation locations, with $s_i \in \mathbb{R}^d$ for $i \in \mathbb{N}$. We assume that $(s_i)_{i \in \mathbb{N}}$ satisfies Condition 1.

For $n \in \mathbb{N}$, we let $y = (y_1, \dots, y_n)^\top = (Y(s_1), \dots, Y(s_n))^\top$ be the (non-Gaussian) observation vector and $R = (k_Y(s_i - s_j))_{i,j=1,\dots,n}$ be its covariance matrix.

The problem of estimating the covariance function k_Y from the observation vector y is crucial and has been extensively studied in the Gaussian case (when T is a linear function). Classically, we assume that k_Y belongs to a parametric family of covariance functions. We will provide the asymptotic properties of two of the most popular estimators of the covariance parameters: the one based on the (Gaussian) maximum likelihood [112, 111] and the one based on cross validation [123, 60, 121]. To our knowledge, such properties are currently known only for Gaussian processes, and we will provide analogous properties in the transformed Gaussian framework.

6.3.2 Bounds on the elements of inverse covariance matrices

In the case of (non-transformed) Gaussian processes, one important argument for establishing the asymptotic properties of the maximum likelihood and cross validation estimators is to bound the largest eigenvalue of the inverse covariance matrix R^{-1} . Unfortunately, due to the non-linearity of the transformation T , such a bound on the largest eigenvalue is no longer sufficient in our setting.

To circumvent this issue, we obtain in the following theorem stronger control over the matrix R^{-1} : we show that its coefficients decrease polynomially quickly with respect to the corresponding distance between observation locations. This theorem may have an interest in itself.

Theorem 1. *Consider the setting of Section 6.3.1. For all fixed $0 < \tau < \infty$, we have, for all $n \in \mathbb{N}$ and $i, j = 1, \dots, n$*

$$\left| (R^{-1})_{i,j} \right| \leq \frac{C_{\sup}}{1 + |s_i - s_j|^{d+\tau}},$$

where $C_{\sup}$ depends on τ but does not depend on n, i, j .

6.3.3 Central limit theorem for quadratic forms of transformed Gaussian processes

In the proofs on covariance parameter estimation of Gaussian processes, a central step is to show the asymptotic normality of quadratic forms of large Gaussian vectors. This asymptotic normality is established by diagonalizing the matrices of the quadratic forms. This diagonalization provides sums of squares of decorrelated Gaussian variables and thus sums of independent variables [124, 122].

In the transformed Gaussian case, one has to deal with quadratic forms involving transformations of Gaussian vectors. Hence, the previous arguments are not longer valid. To overcome this issue, we provide below a general central limit theorem for quadratic forms of transformed Gaussian vectors. This theorem may have an interest in itself.

This asymptotic normality result is established by considering a metric d_w generating the topology of weak convergence on the set of Borel probability measures on Euclidean spaces (see, e.g., [149] p. 393). We prove that the distance between the sequence of the standardized distributions of the quadratic forms and Gaussian distributions decreases to zero when n increases. The introduction of the metric d_w enables us to formulate asymptotic normality results in cases when the sequence of standardized variances of the quadratic forms does not necessarily converge as $n \rightarrow \infty$.

Theorem 2. *Consider the setting of Section 6.3.1. Let $(A_n)_{n \in \mathbb{N}}$ be a sequence of matrices such that A_n has dimension $n \times n$ for any $n \in \mathbb{N}$. Let $A = A_n$ for concision. Assume that for all $n \in \mathbb{N}$ and for all $i, j = 1, \dots, n$,*

$$|A_{i,j}| \leq \frac{C_{\sup}}{1 + |s_i - s_j|^{d+C_{\inf}}},$$

where $C_{\sup}$ does not depend on i, j . Let

$$V_n = \frac{1}{n} y^\top A y. \tag{6.2}$$

Let $\mathcal{L}_n$ be the distribution of $\sqrt{n}(V_n - \mathbb{E}[V_n])$. Then, as $n \rightarrow \infty$,

$$d_w(\mathcal{L}_n, \mathcal{N}[0, n \text{Var}(V_n)]) \rightarrow 0.$$

In addition, the sequence $(n \text{Var}(V_n))_{n \in \mathbb{N}}$ is bounded.

Remark 1. In the case where limits E_∞ and σ_∞ exist, such that

$$\mathbb{E}[V_n] - E_\infty = o(n^{-1/2})$$

and the sequence $(n \text{Var}(V_n))_{n \in \mathbb{N}}$ converges to a fixed variance σ_∞^2 , the result of Theorem 2 can be written in the classical form

$$\sqrt{n}(V_n - E_\infty) \xrightarrow{\mathcal{L}} \mathcal{N}[0, \sigma_\infty^2],$$

as $n \rightarrow \infty$.

Remark 2. Throughout Section 6.3, we assume that k_Z satisfies Condition 2. Note that this condition does not impose k_Z to be continuous at zero. Hence, Theorems 1 and 2 allow for a decomposition $k_Z(s) = \bar{k}_Z(s) + \eta 1_{s=0}$, for $s \in \mathbb{R}^d$, where $\bar{k}_Z : \mathbb{R}^d \rightarrow \mathbb{R}$ is a continuous stationary covariance function and $\eta \geq 0$. Hence, Theorems 1 and 2 allow for an additive nugget effect, with variance η , on the unobserved Gaussian process Z . Furthermore, we remark that in this case, when $\eta > 0$, k_Z automatically satisfies Condition 2 ii), as the eigenvalues of covariance matrices obtained from k_Z are larger than η .

Remark 3. One can check that in the proofs of Theorems 1 and 2, it is not necessary that k_Z satisfies Condition 2 ii). Hence, these two theorems hold whenever T satisfies Condition 3, k_Y satisfies Condition 2 i) and ii) and k_Z satisfies Condition 2 i). There are various assumptions that can be made, in order to guarantee that k_Y satisfies Condition 2 ii).

First, if a nugget effect as described in Remark 2 is assumed on the covariance function k_Z , then k_Z automatically satisfies Condition 2 ii) and thus, from Lemma 1, k_Y satisfies Condition 2 ii).

Second, it would be possible to assume that for $s \in \mathbb{R}^d$, we have $Y(s) = T(Z(s)) + \check{N}(s)$, where Y and Z still have zero-mean and where $\check{N}$ is a centered stationary random field on $\mathbb{R}^d$, independent of Z , with $\text{Cov}(\check{N}(u), \check{N}(v)) = \eta 1_{u=v}$, for $u, v \in \mathbb{R}^d$, with $\eta > 0$. In this case, the covariance function k_Y of Y satisfies Condition 2 ii), similarly as in Remark 2. Furthermore, consider the case where there exists $\check{T} : \mathbb{R} \rightarrow \mathbb{R}$ satisfying Condition 3, such that, for $s \in \mathbb{R}^d$, $\check{N}(s) = \check{T}(\zeta(s))$, where ζ is a centered Gaussian random field, independent of Z , with $\text{Cov}(\zeta(u), \zeta(v)) = \kappa 1_{u=v}$ for $u, v \in \mathbb{R}^d$, with $\kappa > 0$. In this case, one can check that Theorems 1 and 2 hold in this modified setting for Y , with proofs that are minor but straightforward modifications of those given in the appendix (see also the proof of Proposition 1 in the appendix). Note that this modified setting is equivalent to letting the observation vector y be defined by $y_i = T(Z(s_i)) + \xi_i$, $i = 1, \dots, n$, where $\xi_1, \dots, \xi_n$ are i.i.d, independent of Z , with zero-mean and variance η , since $s_1, \dots, s_n$ are two-by-two distinct.

Third, in the case where k_Y and k_Z are continuous, we have shown in Lemma 1 that if k_Z satisfies Condition 2 ii), then k_Y satisfies Condition 2 ii). In Sections 6.3 and 6.4, we thus simply assume that k_Z satisfies Condition 2 ii). A classical assumption that guarantees k_Z to satisfy Condition 2 ii) is that k_Z has a strictly positive Fourier transform [122, 145] (see also the proof of Lemma 2).

Finally, when k_Y is continuous, one could also assume that its Fourier transform is strictly positive to guarantee that k_Y satisfies Condition 2 ii). In fact, this is what we do in Section 6.5, see Condition 5.

6.4 Estimation of a single variance parameter

We let σ_0^2 be the marginal variance of Y , that is $\text{Var}(Y(s)) = \sigma_0^2$ for any $s \in \mathbb{R}^d$. We let $k_Y = \sigma_0^2 c_Y$ be the stationary covariance function of Y , where c_Y is a correlation function. We assume that the same conditions as in Section 6.3 hold. Then, the standard Gaussian maximum likelihood estimator of the variance parameter is

$$\hat{\sigma}_{\text{ML}}^2 = \frac{1}{n} y^\top C^{-1} y,$$

where $C = (c_Y(s_i - s_j))_{1 \leq i, j \leq n}$. One can simply show that $\mathbb{E}[\hat{\sigma}_{\text{ML}}^2] = \sigma_0^2$ even though y is not a Gaussian vector, since y has mean vector 0 and covariance matrix $\sigma_0^2 C$. Hence, a direct consequence of Theorems 1 and 2 is then that the maximum likelihood estimator

is asymptotically Gaussian, with a $n^{1/2}$ rate of convergence, even though the transformed process Y is not a Gaussian process.

Corollary 1. *Let $\mathcal{L}_n$ be the distribution of $\sqrt{n}(\hat{\sigma}_{ML}^2 - \sigma_0^2)$.*

Then, as $n \rightarrow \infty$,

$$d_w(\mathcal{L}_n, \mathcal{N}[0, n \text{Var}(\hat{\sigma}_{ML}^2 - \sigma_0^2)]) \rightarrow 0.$$

In addition, the sequence $(n \text{Var}(\hat{\sigma}_{ML}^2 - \sigma_0^2))_{n \in \mathbb{N}}$ is bounded.

The proof of Theorem 2 actually allows us to study another estimator of the variance σ_0^2 of the form

$$\hat{\sigma}_{ML,K}^2 = \frac{1}{n} y^\top C_K^{-1} y,$$

where $(C_K^{-1})_{i,j} = (C^{-1})_{i,j} 1_{|s_i - s_j| \leq K}$. The proof of Theorem 2 then directly implies the following.

Corollary 2. *Let K_n be any sequence of positive numbers tending to infinity. Let $\mathcal{L}_{K_n,n}$ be the distribution of $\sqrt{n}(\hat{\sigma}_{ML,K_n}^2 - \sigma_0^2)$.*

Then, as $n \rightarrow \infty$,

$$d_w(\mathcal{L}_{K_n,n}, \mathcal{N}[0, n \text{Var}(\hat{\sigma}_{ML,K_n}^2 - \sigma_0^2)]) \rightarrow 0.$$

In addition, we have

$$n \text{Var}(\hat{\sigma}_{ML,K_n}^2 - \sigma_0^2) - n \text{Var}(\hat{\sigma}_{ML}^2 - \sigma_0^2) \rightarrow 0.$$

The above corollary shows that one can taper the elements of C^{-1} when estimating the variance parameter, and obtain the same asymptotic distribution of the error, as long as the taper range goes to infinity, with no rate assumption. This result may have an interest in itself, in view of the existing literature on covariance tapering for Gaussian processes under increasing-domain asymptotics [140, 139]. We also remark that the computation costs of $\hat{\sigma}_{ML,K}^2$ and $\hat{\sigma}_{ML}^2$ have the same orders of magnitude because C^{-1} needs to be computed in both cases.

Remark 4. *In the results of this section, the condition that F is non-decreasing in Condition 3 ii) can be replaced by the condition that C has its smallest eigenvalue bounded away from zero. This can be checked in the corresponding proofs.*

6.5 General covariance

6.5.1 Framework

As in Section 6.3.1, we consider a zero-mean Gaussian process Z defined on $\mathbb{R}^d$ with covariance function k_Z satisfying Condition 2. Let Y be the random field defined for any $s \in \mathbb{R}^d$ by $Y(s) = T(Z(s))$, where T is a fixed function satisfying Condition 3. Furthermore we assume that Y has zero-mean function and we recall that from Lemma 1, its covariance function k_Y also satisfies Condition 2. Finally, the sequence of observation locations $(s_i)_{i \in \mathbb{N}}$ satisfies Condition 1.

Let $\{k_{Y,\theta}; \theta \in \Theta\}$ be a parametric set of stationary covariance functions on $\mathbb{R}^d$, with Θ a compact set of $\mathbb{R}^p$. We consider the following condition on this parametric set of covariance functions.

Condition 4. *For all $s \in \mathbb{R}^d$, $k_{Y,\theta}(s)$ is three times continuously differentiable with respect to θ , and we have*

$$\sup_{\theta \in \Theta} |k_{Y,\theta}(s)| \leq C_{\sup} \exp(-C_{\inf}|s|), \quad (6.3)$$

$$\sup_{\substack{\theta \in \Theta \\ \ell=1,2,3 \\ i_1, \dots, i_\ell=1, \dots, p}} \left| \frac{\partial k_{Y,\theta}(s)}{\partial \theta_{i_1}, \dots, \partial \theta_{i_\ell}} \right| \leq \frac{C_{\sup}}{1 + |s|^{d+C_{\inf}}}. \quad (6.4)$$

The smoothness condition in (6.4) is classical and is assumed for instance in [122]. As discussed after Condition 2, milder versions of (6.3) can be assumed for non-transformed Gaussian processes, but (6.3) is satisfied by most classical families of covariance functions nonetheless.

The next condition, on the Fourier transforms of the covariance functions in the model, is standard.

Condition 5. *We let $\hat{k}_{Y,\theta}$ be the Fourier transform of $k_{Y,\theta}$. Then $\hat{k}_{Y,\theta}(f)$ is jointly continuous with respect to θ and f and is strictly positive on $\Theta \times \mathbb{R}^d$.*

We remark that Condition 5 does not automatically follow from Condition 4 (nor from Conditions 6 to 10 below). For instance, in the case $d = 1$, the family of triangular covariance functions $\{\sigma^2 c_Y; \sigma^2 \in [\sigma_{\inf}^2, \sigma_{\sup}^2]\}$, with $0 < \sigma_{\inf}^2 < \sigma_{\sup}^2 < \infty$ and with $c_Y(x) = (1 - |x|)^+$ for $x \in \mathbb{R}$, satisfies Condition 4 but not Condition 5 [145].

Finally, the next condition means that we address the well-specified case [60, 146], where the family of covariance functions does contain the true covariance function of Y . The well-specified case is considered in the majority of the literature on Gaussian processes.

Condition 6. *There exists θ_0 in the interior of Θ such that $k_Y = k_{Y,\theta_0}$.*

In the next two subsections, we study the asymptotic properties of two classical estimators (maximum likelihood and cross validation) for the covariance parameter θ_0 . The asymptotic properties of these estimators are already known for Gaussian processes and we extend them to the non-Gaussian process Y .

6.5.2 Maximum Likelihood

For $n \in \mathbb{N}$, let R_θ be the $n \times n$ matrix $(k_{Y,\theta}(s_i - s_j))_{i,j=1, \dots, n}$, and let

$$\hat{\theta}_{\text{ML}} \in \operatorname{argmin}_{\theta \in \Theta} L_\theta \quad (6.5)$$

with

$$L_\theta = \frac{1}{n} \left(\log(\det(R_\theta)) + y^\top R_\theta^{-1} y \right)$$

be a maximum likelihood estimator. We will provide its consistency under the following condition.

Condition 7. For all $\chi > 0$ we have

$$\liminf_{n \rightarrow \infty} \inf_{\|\theta - \theta_0\| \geq \chi} \frac{1}{n} \sum_{i,j=1}^n \left(k_{Y,\theta}(s_i - s_j) - k_{Y,\theta_0}(s_i - s_j) \right)^2 > 0.$$

Condition 7 can be interpreted as a global indentifiability condition. It implies in particular that two different covariance parameters yield two different distributions for the observation vector. It is used in several studies, for instance [122].

Theorem 3. Consider the setting of Section 6.5.1 for Z , T , Y and $(s_i)_{i \in \mathbb{N}}$.

Assume that Conditions 4, 5, 6 and 7 hold. Then, as $n \rightarrow \infty$,

$$\hat{\theta}_{ML} \xrightarrow{p} \theta_0.$$

Remark 5. Let us discuss and interpret Theorem 3. In Theorem 3, there is a Gaussian process Z with zero-mean (see Section 6.6.2 for a discussion of the case where Z has non-zero mean) and covariance function k_Z . Consider for the sake of interpretation that the covariance function k_Z belongs to a set of covariance functions $\{k_{Z,\alpha}, \alpha \in \mathcal{A}\}$, with $k_Z = k_{Z,\alpha_0}$, $\alpha_0 \in \mathcal{A}$.

In Theorem 3 there is a transformation $T : \mathbb{R} \rightarrow \mathbb{R}$ that is fixed and unknown. The aim of this paper is not to estimate T , and in particular we do not assume that T belongs to a known parametric set of transformation functions. This transformation T then defines the non-Gaussian process $Y = T(Z)$. We also assume that Y has zero-mean, the case of a non-zero mean also being discussed in Section 6.6.2. In Theorem 3, we assume that the covariance function k_Y of Y belongs to the set of covariance functions $\{k_{Y,\theta}; \theta \in \Theta\}$, with $k_Y = k_{Y,\theta_0}$ for $\theta_0 \in \Theta$.

Assume for the sake of interpretation that, for $\alpha \in \mathcal{A}$, if the covariance function of Z was $k_{Z,\alpha}$, then the covariance function of Y would belong to $\{k_{Y,\theta}; \theta \in \Theta\}$. Assume also for the sake of discussion that $\theta \mapsto k_{Y,\theta}$ is injective (which is an extension of the requirement in Condition 7). Then, there exists a fixed mapping $\tau : \mathcal{A} \rightarrow \Theta$ such that, if the covariance function of Z was $k_{Z,\alpha}$, then the covariance function of Y would be $k_{Y,\tau(\alpha)}$. The mapping τ is uniquely defined because $\theta \mapsto k_{Y,\theta}$ is assumed to be injective.

The process Y is observed at $s_1, \dots, s_n$, yielding the observation vector y and then the (Gaussian) maximum likelihood estimator $\hat{\theta}_{ML}$ in Theorem 3. This estimator is based on the false assumption that Y is a Gaussian process but on the correct assumption that the covariance function of Y belongs to $\{k_{Y,\theta}; \theta \in \Theta\}$. This estimator is thus misspecified from a likelihood point of view but well-specified from a covariance function point of view. Theorem 3 shows that this estimator is consistent and Theorem 4 below will show that this estimator is asymptotically unbiased with an asymptotic covariance matrix of sandwich form, see (6.6). We remark that this form is typical in misspecified models [150, 151, 152].

If the transformation T was known and injective, one could recover the values $Z(s_1), \dots, Z(s_n)$ from $Y(s_1), \dots, Y(s_n)$ and thus compute the (well-specified) Gaussian maximum likelihood estimator $\hat{\alpha}_{ML}$ of α_0 . Note that we have $\hat{\theta}_{ML} \neq \tau(\hat{\alpha}_{ML})$ in general. We remark that one can expect $\hat{\alpha}_{ML}$ to satisfy some type of statistical efficiency principle, as an estimator of α_0 , since it is a well-specified maximum likelihood estimator. Nevertheless, while there is a fair amount of work on efficiency of estimators in the case of i.i.d. data [153], almost no counterparts exist, to the best of our knowledge, for the efficiency of estimators of covariance parameters of Gaussian processes. One of the few examples is [154], in the special case of the exponential covariance functions. If an efficiency property could be proved for $\hat{\alpha}_{ML}$, then one could also expect that $\tau(\hat{\alpha}_{ML})$ satisfies a similar efficiency property, as an estimator of θ_0 (for instance,

in the spirit of Theorem 25.47 in [153]). In particular, one can expect $\tau(\hat{\alpha}_{ML})$ to be asymptotically a better estimator of θ_0 than $\hat{\theta}_{ML}$. Nevertheless, we emphasize that when T is unknown, $\tau(\hat{\alpha}_{ML})$ can not be computed but $\hat{\theta}_{ML}$ can.

Finally, let us explicit the mapping τ in a special case. Consider that the transformation function is given by $T(x) = x^2 - \mathbb{E}[Z(0)^2]$ for $x \in \mathbb{R}$. Consider also that the covariance models $\{k_{Y,\theta}; \theta \in \Theta\}$ and $\{k_{Z,\alpha}; \alpha \in \mathcal{A}\}$ are $\{\sigma^2 e^{-\rho \|\cdot\|}; (\sigma^2, \rho) \in \Theta\}$ and $\{\sigma^2 e^{-\rho \|\cdot\|}; (\sigma^2, \rho) \in \mathcal{A}\}$, with $\Theta, \mathcal{A} \subset (0, \infty)^2$. Then if $\alpha_0 = (\sigma_0^2, \rho_0)$, by Mehler's formula [155], for $u, v \in \mathbb{R}^d$,

$$\text{Cov}(Y(u), Y(v)) = 2 \text{Cov}(Z(u), Z(v))^2 = 2\sigma_0^4 e^{-2\rho_0 \|u-v\|}$$

and thus we have, for $\alpha = (\alpha_1, \alpha_2)$, $\tau(\alpha) = (2\alpha_1^2, 2\alpha_2)$.

Condition 8. For any $(\chi_1, \dots, \chi_p) \neq (0, \dots, 0)$, we have

$$\liminf_{n \rightarrow \infty} \frac{1}{n} \sum_{i,j=1}^n \left(\sum_{\ell=1}^p \chi_\ell \frac{\partial k_{Y,\theta_0}(s_i - s_j)}{\partial \theta_\ell} \right)^2 > 0.$$

Condition 8 can be interpreted as a regularity condition and as a local indentifiability condition around θ_0 . In the next theorem, we provide the asymptotic normality of the maximum likelihood estimator. In this theorem, the matrices M_{θ_0} and Σ_{θ_0} depend on the number of observation locations.

Theorem 4. Consider the setting of Section 6.5.1 for Z , T , Y and $(s_i)_{i \in \mathbb{N}}$. Assume that Conditions 4, 5, 6, 7 and 8 hold. Let M_{θ_0} be the $p \times p$ matrix defined by

$$(M_{\theta_0})_{i,j} = \frac{1}{n} \text{tr} \left(R_{\theta_0}^{-1} \frac{\partial R_{\theta_0}}{\partial \theta_i} R_{\theta_0}^{-1} \frac{\partial R_{\theta_0}}{\partial \theta_j} \right).$$

Let Σ_{θ_0} be the $p \times p$ covariance matrix defined by $(\Sigma_{\theta_0})_{i,j} = \text{Cov}(n^{1/2} \partial L_{\theta_0} / \partial \theta_i, n^{1/2} \partial L_{\theta_0} / \partial \theta_j)$.

Let $\mathcal{L}_{\theta_0,n}$ be the distribution of $\sqrt{n}(\hat{\theta}_{ML} - \theta_0)$.

Then, as $n \rightarrow \infty$,

$$d_w \left(\mathcal{L}_{\theta_0,n}, \mathcal{N}[0, M_{\theta_0}^{-1} \Sigma_{\theta_0} M_{\theta_0}^{-1}] \right) \rightarrow 0. \quad (6.6)$$

In addition,

$$\limsup_{n \rightarrow \infty} \lambda_1(M_{\theta_0}^{-1} \Sigma_{\theta_0} M_{\theta_0}^{-1}) < +\infty. \quad (6.7)$$

Remark 6. If the sequences of matrices M_{θ_0} and Σ_{θ_0} converge as $n \rightarrow \infty$, then $\sqrt{n}(\hat{\theta}_{ML} - \theta_0)$ converges in distribution to a fixed centered Gaussian distribution where the limiting covariance matrix is given by

$$\lim_{n \rightarrow \infty} M_{\theta_0}^{-1} \Sigma_{\theta_0} M_{\theta_0}^{-1}.$$

Conditions 7 and 8 involve the model of covariance functions $\{k_{Y,\theta}; \theta \in \Theta\}$ and the sequence of observation locations $(s_i)_{i \in \mathbb{N}}$ but not the transformation T . They are further discussed, in a different context, in [156]. We believe that these conditions are mild. For instance, Conditions 7 and 8 hold when the sequence of observation locations $(s_i)_{i \in \mathbb{N}}$ is a randomly perturbed regular grid, as in [122].

Lemma 3 (see [122]). For $i \in \mathbb{N}$, let $s_i = g_i + \epsilon_i$, where $(g_i)_{i \in \mathbb{N}}$ is a sequence with, for $N \in \mathbb{N}$, $\{g_1, \dots, g_{Nd}\} = \{(i_1, \dots, i_d); i_1 = 1, \dots, N, \dots, i_d = 1, \dots, N\}$ and where $(\epsilon_i)_{i \in \mathbb{N}}$ is a sequence of i.i.d. random variables with distribution on $[-1/2 + \delta, 1/2 - \delta]^d$ with $0 < \delta < 1/2$. Then, Condition 7 holds almost surely, provided that, for $\theta \neq \theta_0$, there exists $i \in \mathbb{Z}^d \setminus \{0\}$ for which $k_{Y,\theta}(i + \epsilon_1 - \epsilon_2)$ and $k_{Y,\theta_0}(i + \epsilon_1 - \epsilon_2)$ are not almost surely equal.

Furthermore, Condition 8 holds almost surely, provided that for $(\chi_1, \dots, \chi_p) \neq (0, \dots, 0)$, there exists $i \in \mathbb{Z}^d \setminus \{0\}$ for which $\sum_{\ell=1}^p \chi_\ell \partial k_{Y,\theta_0}(i + \epsilon_1 - \epsilon_2) / \partial \theta_\ell$ is not almost surely equal to zero.

6.5.3 Cross Validation

We consider the cross validation estimator consisting of minimizing the average of the leave-one-out square errors.

Since the leave-one-out errors do not depend on the variance $k_{Y,\theta}(0)$, we introduce some additional notation. In Sections 6.5.3 and 6.5.4, we let $\Theta = [\sigma_{\inf}^2, \sigma_{\sup}^2] \times \mathcal{S}$ where $0 < \sigma_{\inf}^2 < \sigma_{\sup}^2 < \infty$ are fixed and where $\mathcal{S}$ is compact in $\mathbb{R}^{p-1}$. We let $\theta = (\sigma^2, \psi)$ with $\sigma_{\inf}^2 \leq \sigma^2 \leq \sigma_{\sup}^2$ and $\psi \in \mathcal{S}$. We assume that for $\theta \in \Theta$, $k_{Y,\theta} = \sigma^2 c_{Y,\psi}$, with $c_{Y,\psi}$ a stationary correlation function. Similarly, we let $\theta_0 = (\sigma_0^2, \psi_0)$. For $\psi \in \mathcal{S}$, we let $C_\psi = (c_{Y,\psi}(s_i - s_j))_{i,j=1,\dots,n}$. Cross validation is defined for $n \in \mathbb{N}$ by

$$\hat{\psi}_{\text{CV}} \in \underset{\psi \in \mathcal{S}}{\operatorname{argmin}} CV_\psi, \quad (6.8)$$

with

$$CV_\psi = \frac{1}{n} y^\top C_\psi^{-1} \operatorname{diag}(C_\psi^{-1})^{-2} C_\psi^{-1} y,$$

where $\operatorname{diag}(M)$ is obtained by setting the off-diagonal elements of a square matrix M to zero. The criterion CV_ψ is the average of the leave-one-out square errors, as is shown for instance in [157, 60]. More precisely, if we let $\hat{y}_{i,\psi}$ be the best linear predictor of y_i based on $(y_1, \dots, y_{i-1}, y_{i+1}, \dots, y_n)$, assuming that Y has covariance function $\sigma^2 c_{Y,\psi}$, for any choice of $\sigma^2 \in [\sigma_{\inf}^2, \sigma_{\sup}^2]$, then we have

$$CV_\psi = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_{i,\psi})^2.$$

In particular, we point out that $\hat{y}_{i,\psi}$ does not depend on σ^2 , because the best linear predictors from the two covariance functions $\sigma_1^2 c_{Y,\psi}$ and $\sigma_2^2 c_{Y,\psi}$ are the same, for any $\sigma_1^2, \sigma_2^2 \in [\sigma_{\inf}^2, \sigma_{\sup}^2]$.

Let us insist on the fact that we consider here a cross validation estimator of the correlation parameter ψ_0 but that we do not consider a cross validation estimator of σ_0^2 . Indeed, we consider a cross validation estimator obtained by minimizing the average of the leave-one-out square errors, that depends only on ψ , not on σ^2 .

The asymptotic behaviour of $\hat{\psi}_{\text{CV}}$ was studied in the Gaussian framework in [122] and under increasing-domain asymptotics.

The next identifiability condition is also made in [122].

Condition 9. For all $\chi > 0$, we have

$$\liminf_{n \rightarrow \infty} \inf_{\|\psi - \psi_0\| \geq \chi} \frac{1}{n} \sum_{i,j=1}^n (c_{Y,\psi}(s_i - s_j) - c_{Y,\psi_0}(s_i - s_j))^2 > 0.$$

The next theorem provides the consistency of the cross validation estimator.

Theorem 5. *Consider the setting of Section 6.5.1 for Z, T, Y and $(s_i)_{i \in \mathbb{N}}$. Assume that Conditions 4, 5, 6 and 9 hold. Then, as $n \rightarrow \infty$,*

$$\hat{\psi}_{CV} \xrightarrow{P} \psi_0.$$

The next condition is a local identifiability condition.

Condition 10. *For any $(\chi_1, \dots, \chi_{p-1}) \neq (0, \dots, 0)$, we have*

$$\liminf_{n \rightarrow \infty} \frac{1}{n} \sum_{i,j=1}^n \left(\sum_{\ell=1}^{p-1} \chi_\ell \frac{\partial}{\partial \psi_\ell} (c_{Y,\psi_0}(s_i - s_j)) \right)^2 > 0.$$

In the next theorem, we provide the asymptotic normality of the cross validation estimator. In this theorem, the matrices N_{ψ_0} and Γ_{ψ_0} depend on the number of observation locations.

Theorem 6. *Consider the setting of Section 6.5.1 for Z, T, Y and $(s_i)_{i \in \mathbb{N}}$. Assume that Conditions 4, 5, 6, 9 and 10 hold. Let N_{ψ_0} be the $(p-1) \times (p-1)$ matrix defined by*

$$\begin{aligned} (N_{\psi_0})_{i,j} = & -\frac{8}{n} \text{tr} \left(\frac{\partial C_{\psi_0}}{\partial \psi_j} C_{\psi_0}^{-1} \text{diag}(C_{\psi_0}^{-1})^{-3} \text{diag} \left(C_{\psi_0}^{-1} \frac{\partial C_{\psi_0}}{\partial \psi_i} C_{\psi_0}^{-1} \right) C_{\psi_0}^{-1} \right) \\ & + \frac{2}{n} \text{tr} \left(\frac{\partial C_{\psi_0}}{\partial \psi_j} C_{\psi_0}^{-1} \text{diag}(C_{\psi_0}^{-1})^{-2} C_{\psi_0}^{-1} \frac{\partial C_{\psi_0}}{\partial \psi_i} C_{\psi_0}^{-1} \right) \\ & + \frac{6}{n} \text{tr} \left(\text{diag}(C_{\psi_0}^{-1})^{-4} \text{diag} \left(C_{\psi_0}^{-1} \frac{\partial C_{\psi_0}}{\partial \psi_i} C_{\psi_0}^{-1} \right) \text{diag} \left(C_{\psi_0}^{-1} \frac{\partial C_{\psi_0}}{\partial \psi_j} C_{\psi_0}^{-1} \right) C_{\psi_0}^{-1} \right). \end{aligned}$$

Let Γ_{ψ_0} be the $(p-1) \times (p-1)$ covariance matrix defined by $(\Gamma_{\psi_0})_{i,j} = \text{Cov}(n^{1/2} \partial C V_{\psi_0} / \partial \psi_i, n^{1/2} \partial C V_{\psi_0} / \partial \psi_j)$. Let $\mathcal{Q}_{\psi_0,n}$ be the distribution of $\sqrt{n}(\hat{\psi}_{CV} - \psi_0)$.

Then, as $n \rightarrow \infty$,

$$d_w \left(\mathcal{Q}_{\psi_0,n}, \mathcal{N}[0, N_{\psi_0}^{-1} \Gamma_{\psi_0} N_{\psi_0}^{-1}] \right) \rightarrow 0. \quad (6.9)$$

In addition,

$$\limsup_{n \rightarrow \infty} \lambda_1(N_{\psi_0}^{-1} \Gamma_{\psi_0} N_{\psi_0}^{-1}) < +\infty. \quad (6.10)$$

Similarly as for maximum likelihood, Condition 9 is a global identifiability condition for the correlation function. In the same way, Condition 10 is a local identifiability condition for the correlation function around ψ_0 .

We remark that the conditions for cross validation imply those for maximum likelihood.

Lemma 4. *Condition 9 implies Condition 7 and Condition 10 implies Condition 8.*

Finally, similarly as for maximum likelihood, we point out that Conditions 9 and 10 hold in the case of a randomly perturbed regular grid, as in [122].

Lemma 5 (see [122]). *Let $(s_i)_{i \in \mathbb{N}}$ and $(\epsilon_i)_{i \in \mathbb{N}}$ be as in Lemma 3. Then, Condition 9 holds almost surely, provided that, for $\psi \neq \psi_0$, there exists $i \in \mathbb{Z}^d \setminus \{0\}$ for which $c_{Y,\psi}(i + \epsilon_1 - \epsilon_2)$ and $c_{Y,\psi_0}(i + \epsilon_1 - \epsilon_2)$ are not almost surely equal. Furthermore, Condition 10 holds almost surely, provided that for $(\chi_1, \dots, \chi_{p-1}) \neq (0, \dots, 0)$, there exists $i \in \mathbb{Z}^d \setminus \{0\}$ for which $\sum_{\ell=1}^{p-1} \chi_\ell \partial c_{Y,\psi_0}(i + \epsilon_1 - \epsilon_2) / \partial \psi_\ell$ is not almost surely equal to zero.*

Remark 7. Here we have provided the asymptotic properties of cross validation by minimization of the average of the square leave-one-out errors for estimating ψ_0 . This corresponds to an extension of [122] to the transformed Gaussian framework. We remark that there exist other cross validation methods that provide estimators of both ψ_0 and σ_0^2 [60, 123, 112, 121] but their asymptotic properties are not established under increasing-domain asymptotics, even for Gaussian processes (note that [123] provides the fixed-domain asymptotic properties of one of these other cross validation methods for the exponential covariance function in dimension one). It would be interesting to extend the results of Section 6.5.3 to these other cross validation methods in future work.

6.5.4 Joint asymptotic normality

From Theorems 4 and 6, both the maximum likelihood and cross validation estimators converge at the standard parametric rate $n^{1/2}$. Let us write $\hat{\theta}_{\text{ML}} = (\hat{\sigma}_{\text{ML}}^2, \hat{\psi}_{\text{ML}})$. In the case where T is the identity function (that is, where we observe Gaussian processes instead of transformed Gaussian processes), numerical experiments tend to show that $\hat{\psi}_{\text{ML}}$ is more accurate than $\hat{\psi}_{\text{CV}}$ [122]. Indeed, when T is the identity function, maximum likelihood is based on the Gaussian probability density function of the observation vector.

In contrast, when T is not the identity function, $\hat{\psi}_{\text{ML}}$ is an M-estimator based on a criterion which does not coincide with the observation probability density function anymore. Hence, it is conceivable that $\hat{\psi}_{\text{CV}}$ could become more accurate than $\hat{\psi}_{\text{ML}}$. Furthermore, it is possible that using linear combinations of these two estimators could result in a third one with improved accuracy [158, 159].

Motivated by this discussion, we now provide a joint central limit theorem for the maximum likelihood and cross validation estimators.

Theorem 7. Consider the setting of Section 6.5.1 for Z , T , Y and $(s_i)_{i \in \mathbb{N}}$. Assume that Conditions 4, 5, 6, 9 and 10 hold. Let D_{θ_0} be the $(2p-1) \times (2p-1)$ block diagonal matrix with first $p \times p$ block equal to M_{θ_0} and second $(p-1) \times (p-1)$ block equal to N_{ψ_0} , with the notation of Theorems 4 and 6. Also let Ψ_{θ_0} be the $(2p-1) \times (2p-1)$ covariance matrix of the vector $n^{1/2}(\partial L_{\theta_0}/\partial \theta, \partial CV_{\psi_0}/\partial \psi)$.

Let $\mathcal{Q}_{\theta_0, n}$ be the distribution of

$$\sqrt{n} \begin{pmatrix} \hat{\theta}_{\text{ML}} - \theta_0 \\ \hat{\psi}_{\text{CV}} - \psi_0 \end{pmatrix}.$$

Then, as $n \rightarrow \infty$,

$$d_w \left(\mathcal{Q}_{\theta_0, n}, \mathcal{N}[0, D_{\theta_0}^{-1} \Psi_{\theta_0} D_{\theta_0}^{-1}] \right) \rightarrow 0. \quad (6.11)$$

In addition,

$$\limsup_{n \rightarrow \infty} \lambda_1(D_{\theta_0}^{-1} \Psi_{\theta_0} D_{\theta_0}^{-1}) < +\infty. \quad (6.12)$$

Remark 8. From Theorem 7, considering any C^1 function f from $\mathcal{S}^2 \rightarrow \mathcal{S}$ such that $f(\psi, \psi) = \psi$ for any $\psi \in \mathcal{S}$, and applying the classical delta method, we obtain the asymptotic normality of the new estimator $f(\hat{\psi}_{\text{ML}}, \hat{\psi}_{\text{CV}})$. A classical choice for f is $f(\psi_1, \psi_2) = \lambda\psi_1 + (1-\lambda)\psi_2$, which leads to linear aggregation [158, 159]. We remark that selecting an optimal λ leading to the smallest asymptotic covariance matrix necessitates an estimation of the asymptotic covariance matrix in Theorem 7. We leave this as an open problem for further research.

Remark 9. In all the results of Section 6.5, the condition that F is non-decreasing in Condition 3 ii) is actually not needed. It can indeed be verified that all the proofs for Section 6.5 do not use this condition. In fact, Condition 3 ii) is used in Lemma 1 to show that covariance matrices have their smallest eigenvalues bounded away from zero. This is already the case in Section 6.5, thanks to Condition 5.

Remark 10. The results and proofs in Section 6.5 rely significantly on the assumption that the set of covariance parameters Θ is compact. This is a common assumption in the literature [122, 133, 139]. Nevertheless, this assumption is, in some aspects, unnatural. For instance, the domain of allowed values of estimators of the (constant) variance of a stationary random field is often $[0, \infty)$.

An extension of the results of this paper to non-compact sets Θ of covariance parameters, while valuable, could turn out to be very challenging, and would definitely require new proof arguments. For instance, we remark that letting correlation length parameters go to zero or infinity may yield asymptotic situations and techniques that are qualitatively different from the ones tackled in this paper, see for instance [160, 161].

6.6 Discussions of extensions

6.6.1 Nugget effect

In Section 6.5, Condition 5 implies that the covariance function $k_{Y,\theta}$ is continuous for any $\theta \in \Theta$. Furthermore the true covariance function $k_Y = k_{Y,\theta_0}$ of Y is also continuous. Hence, the results of Section 6.5 hold for models of continuous covariance functions and with error-free observations of continuous transformed Gaussian processes.

Here, we provide an extension of the results of Section 6.5 to a nugget effect on the covariance function $k_{Y,\theta}$. We assume that for $\theta \in \Theta$, the covariance function $k_{Y,\theta}$ is of the form $k_{Y,\theta}(s) = \bar{k}_{Y,\theta}(s) + \eta_\theta 1_{s=0}$, $s \in \mathbb{R}^d$, where $\bar{k}_{Y,\theta}$ is a continuous stationary covariance function on $\mathbb{R}^d$ and $\eta_\theta \geq 0$. In this case, Condition 5 does not hold, but it can be replaced by the following condition.

Condition 11. We have $\inf_{\theta \in \Theta} \eta_\theta > 0$.

We also assume that, for $s \in \mathbb{R}^d$, $Y(s) = T(Z(s)) + \check{N}(s)$, where Y and Z still have zero-mean and where there exists $\check{T} : \mathbb{R} \rightarrow \mathbb{R}$ satisfying Condition 3, such that, for $s \in \mathbb{R}^d$, $\check{N}(s) = \check{T}(\zeta(s))$, where ζ is a centered Gaussian random field, independent of Z , with $\text{Cov}(\zeta(u), \zeta(v)) = \kappa 1_{u=v}$ for $u, v \in \mathbb{R}^d$, with $\kappa > 0$. We also assume that $\check{N}$ has zero-mean. We still assume that k_Z satisfies Condition 2. Note that k_Z is thus not necessarily continuous and can also include a nugget effect as described in Remark 2.

In this setting, the results of Section 6.5 can be extended in a relatively straightforward way.

Proposition 1. Consider the setting of Section 6.6.1. Let Condition 5 be replaced by Condition 11. Let Conditions 4, 6, 7, 8, 9 and 10 be unchanged and applied to the new form of $k_{Y,\theta}$ given in this section. Then Theorems 3, 4, 5, 6 and 7 still hold.

We remark that in Proposition 1 we still address the well-specified case. In particular, there exists $\theta_0 \in \Theta$ such that $k_{Y,\theta_0}(0) = \bar{k}_{Y,\theta_0}(0) + \eta_{\theta_0} = \text{Var}(T(Z(s))) + \text{Var}(\check{T}(\zeta(s)))$, for all $s \in \mathbb{R}^d$. We also remark that the setting considered here is equivalent to additive i.i.d. observation errors on a transformed Gaussian process, since the observation points are two-by-two distinct.

6.6.2 Unknown mean

Throughout Section 6.5, we assume that the mean function of Y is constant, known and equal to zero (in particular, the maximum likelihood estimator in (6.5) is based on the density of a centered Gaussian vector). The results of Section 6.5 also apply to the case of any constant known mean $\nu_0 \in \mathbb{R}$ for Y , since one can then also observe the centered process $Y - \nu_0$. However, the results of Section 6.5 do not apply to the case of a constant unknown mean ν_0 for Y .

In fact, even for (non-transformed) Gaussian processes, there exist very few results on the joint estimation of a constant mean and of covariance parameters by maximum likelihood or cross validation. For instance, the reference [122], that we extend from the Gaussian to transformed Gaussian case, considers a constant known mean. Among these few results, one can mention [162, 138], but the conditions there are more difficult to check and to interpret than those given in [122].

In future work, it would be interesting to extend the results of Section 6.5 to transformed Gaussian processes with constant unknown mean. For instance, one may consider a likelihood criterion to minimize of the form

$$\frac{1}{n} \left(\log(\det(R_\theta)) + (y - v_\nu)^\top R_\theta^{-1} (y - v_\nu) \right),$$

where θ is still the covariance parameter, ν is the mean parameter and $v_\nu = (\nu, \dots, \nu)^\top$ is of size n . In this case a first task is to extend the results of [122] to show the consistency and asymptotic normality of the maximum likelihood estimators $\hat{\theta}_{\text{ML}}$ and $\hat{\nu}_{\text{ML}}$ when Y is Gaussian. A second task is to further extend these results to the case where Y is transformed Gaussian.

Finally, in the case where Z or Y has non-zero constant mean, Theorem 2 can be extended as follows.

Corollary 3. *Consider the setting of Theorem 2, with the only modification that the Gaussian process Z has a constant mean function $\mu_0 \in \mathbb{R}$ and that the transformed process $Y = T(Z)$ has a constant mean function $\nu_0 \in \mathbb{R}$. Then the conclusion of Theorem 2 holds, with y replaced by $y - v_{\nu_0}$.*

This extension of Theorem 2 can be relevant and useful to the question of extending the results of Section 6.5, as discussed above.

6.7 Illustration

In this section we numerically illustrate the convergence of different estimators as stated in Corollary 1 and Theorems 4, 6 and 7. We use the following simulation setup in dimension $d = 2$. For $n = 4(m + 1/2)^2$ we define the grid $\{-m, \dots, m\}^2$ and add i.i.d. variables with uniform distribution on $[-0.4, 0.4]^2$ to obtain n observation points. Thus, we have a distance of at least $\Delta = 0.2$ between the individual observation locations. The zero-mean Gaussian process Z has stationary covariance function $k_Z(s) = \sigma_0^2 \exp(-\|s\|/\rho_0)$, $s \in \mathbb{R}^2$ and we will denote this reference case as the Gaussian case throughout. We define the zero-mean process $Y = T(Z) = Z^2 - \sigma_0^2$ and we will denote this case as the non-Gaussian case. Recall that $k_Y(s) = 2k_Z(2s) = 2\sigma_0^4 \exp(-2\|s\|/\rho_0)$ (see Remark 5). We set the marginal variance to $\sigma_0^2 = 1.5$ and the range to $\rho_0 = 2$. Hence, in the non-Gaussian case, the marginal variance of Y is $2\sigma_0^4 = 4.5$ with a range of $\rho_0/2$ which is equal to a half of that of Z .

To start, we consider the maximum likelihood estimates of the marginal variance parameters, when the range of Y or Z is assumed to be known, i.e., Corollary 1. Figure 6.1 illustrates the empirical densities of $\mathcal{L}_n$ in Corollary 1 for $n = 100, 400$ and 900 observation locations based on $N = 2500$ replicates. For moderate n sizes and in the non-Gaussian case, the asymptotic variance $\sigma_\infty^2 := n \text{Var}(\hat{\sigma}_{\text{ML}}^2 - 2\sigma_0^4)$ (Corollary 1) can be calculated based on (6.26) and using

$$\begin{aligned} \text{Cov}(y_i y_j, y_k y_l) &= 4(k_{i,k}^2 k_{j,l}^2 + k_{i,l}^2 k_{j,k}^2) \\ &\quad + 16(k_{i,k} k_{i,l} k_{j,k} k_{j,l} + k_{i,j} k_{i,l} k_{j,k} k_{k,l} + k_{i,j} k_{i,k} k_{j,l} k_{k,l}), \end{aligned}$$

with $k_{i,j} = k_Z(s_i - s_j)$. The above display follows from tedious computations based on Isserlis' theorem. The densities in red in Figure 6.1 are based on the asymptotic distribution $\mathcal{N}[0, \sigma_\infty^2]$. In the non-Gaussian case and for $n \geq 400$, the calculation of σ_∞^2 is computationally prohibitive, so σ_∞^2 has instead been approximated by the empirical variance with the corresponding densities indicated in green. As expected, the convergence for the Gaussian case is faster than for the non Gaussian case. But in both situations, the results behave nicely.

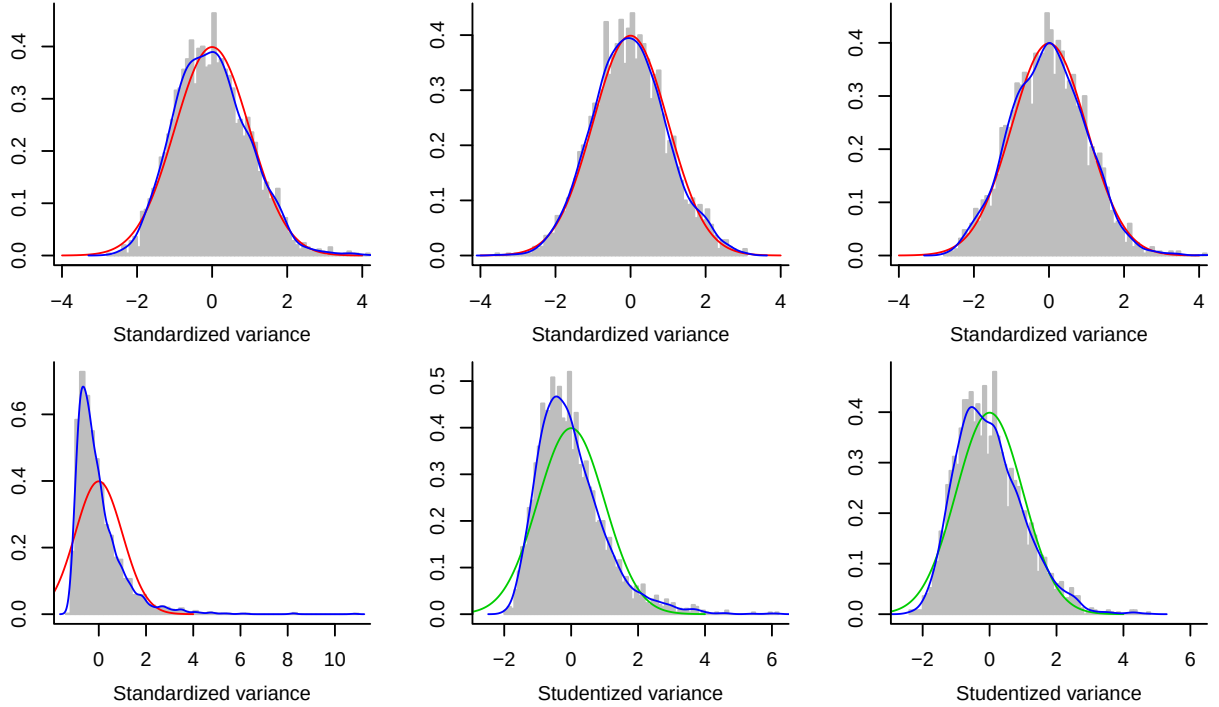


Figure 6.1 – Histograms of standardized and studentized variance maximum likelihood estimates. Blue: empirical density (kernel density estimates), red: asymptotic density, green: asymptotic density based on the empirical variance. The top row shows results for the process Z (Gaussian case), the bottom row for the transformed process $Y = Z^2 - \sigma_0^2$ (non-Gaussian case). The columns are for $n = 100, 400, 900$. All panels are based on $N = 2500$ replicates.

We now turn to Theorem 4 and consider the bivariate variance and range maximum likelihood estimation. That is, we consider the two-dimensional maximum likelihood estimates of (σ_0^2, ρ_0) in the Gaussian case and of $(2\sigma_0^4, \rho_0/2)$ in the non-Gaussian case. Again, we do not observe many surprises. Skewness of the empirical distribution is slightly higher compared to the single variance parameter estimation, and convergence is slightly slower, as is illustrated in Figure 6.2.

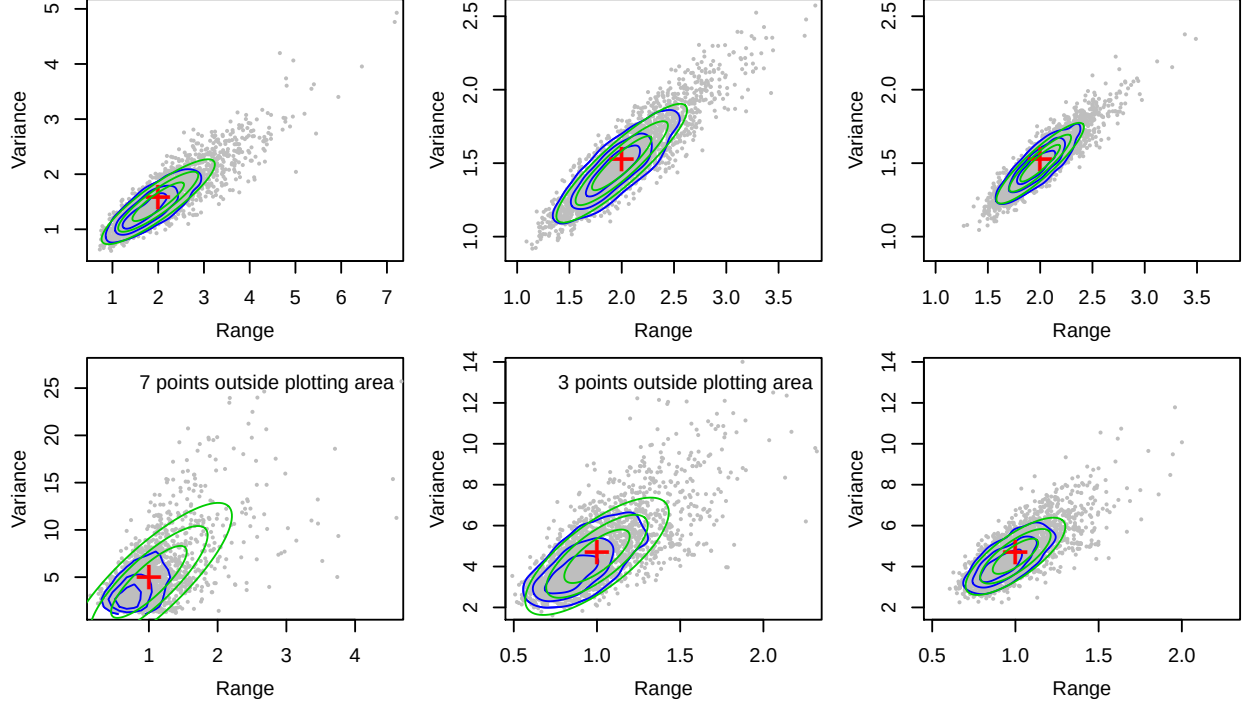


Figure 6.2 – Scatter plots of variance and range maximum likelihood estimates. Blue: contour lines of kernel density estimates; green: contour lines of asymptotic density based on the empirical bivariate covariance matrix; red cross: true mean (true parameter values). The top row shows results for the process Z (Gaussian case), the bottom row for the transformed process $Y = Z^2 - \sigma_0^2$ (non-Gaussian case). The columns are for $n = 100, 400, 900$. All panels are based on $N = 2500$ replicates.

For the general setting, when estimating jointly the variance and range parameter, the asymptotic bivariate covariance matrix is challenging to compute (see Theorem 4) and thus Figure 6.2 illustrates the empirical densities and densities based on the empirical bivariate covariance matrix.

We now consider not only maximum likelihood estimation of the variance and range, but also cross validation estimation of the range (see the beginning of Section 6.5.3). We have observed that the range estimates based on cross validation are much more variable, and in many situations the maximum was attained at the (imposed) boundary. Here we used the bound $[2/15, 12]$, i.e., smaller than the minimal distance between two observation locations and 6 (resp. 12) times the diameter of the observation points of Z (resp. Y). Estimates at or close to the boundary indicate convergence issues and would imply a second, possibly manual, inspection. For the reported results, we eliminated all cross validation cases that yielded estimates outside $[0.14, 11.4]$.

Figure 6.3 shows the mean squared error, squared bias and variance of $\hat{\sigma}_{\text{ML}}^2$, $\hat{\rho}_{\text{ML}}$ and $\hat{\rho}_{\text{CV}}$ under different settings. For maximum likelihood, we consider the univariate case (one parameter is estimated while the other is known) and the bivariate case (both parameters are jointly estimated). For cross validation, only the range parameter is estimated (see the beginning of Section 6.5.3), and thus only the univariate case is considered. The mean squared error is dominated by the variance component. Univariate maximum likelihood estimation for Gaussian cases have low bias and the lowest variance (top left and right panel). Joint maximum likelihood estimation has a somewhat larger variance than individual estimation. Surprisingly, cross validation for Gaussian cases has a higher variance compared to cross validation for non-Gaussian cases.

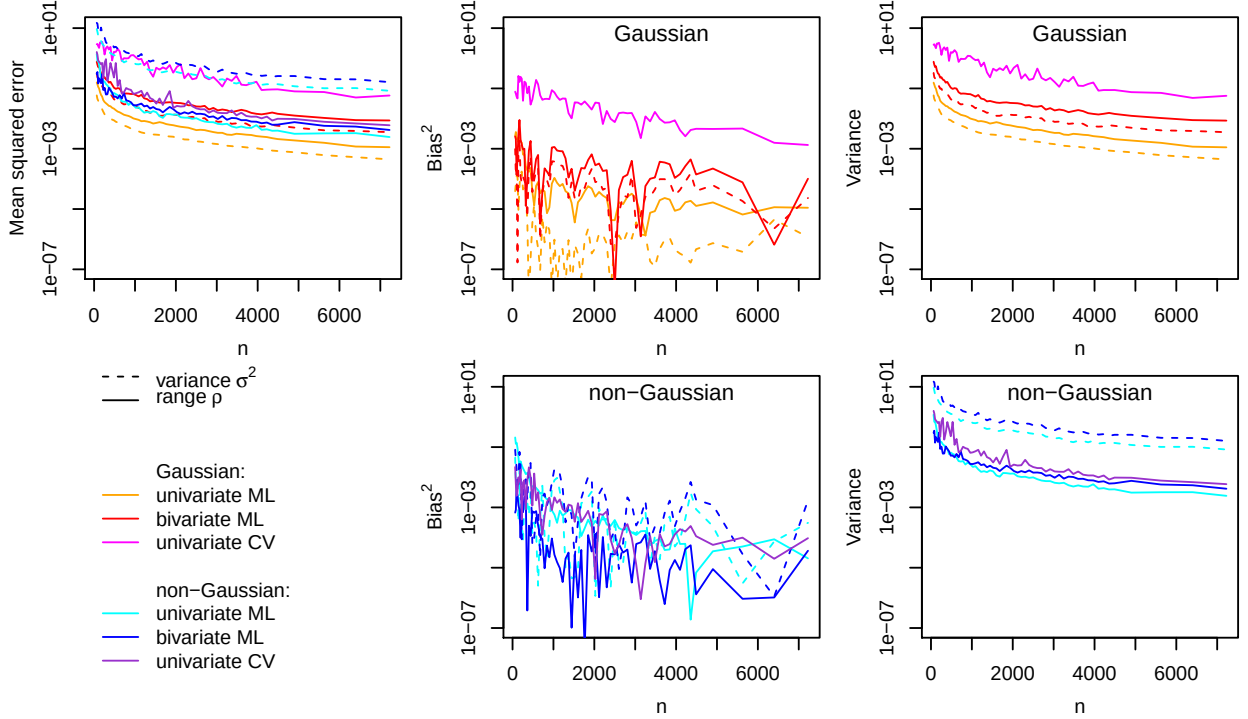


Figure 6.3 – Mean squared error (top left), squared bias (middle column) and variance (right column) as a function of n for different settings in log-scale. The variance parameter is represented in dashed lines, the range parameter in solid lines. Process Z is shown by reddish colors (Gaussian case), transformed process $Y = Z^2 - \sigma_0^2$ in blueish colors (non-Gaussian case). The panels are based on $N = 250$ replicates that did not reveal any convergence issues.

Recall that Theorems 4, 6 and 7 show that, as n increases, the distribution of the standardized estimation error is close to a Gaussian distribution in terms of the metric d_w . In Figure 6.4, we illustrate this by computing one-dimensional Wasserstein distances between the empirical distribution of the standardized estimation errors and Gaussian distributions. The figure shows the Wasserstein distance ($p = 1$) as a function of the number of observation locations n for individual parameters and for specific bivariate settings (similarly as for Figure 6.3). In each case, the samples have been centered around the true mean (true parameter values) and standardized by an empirical standard deviation (n -weighted average over all the samples). Their empirical distribution is compared to the standardized Gaussian distribution. The top left panel shows that the densities of the cross validation parameters are converging slowest whereas their mean squared error is comparable (see Figure 6.3); the densities are highly skewed and thus lead to much larger Wasserstein distances compared to the distributions of the maximum likelihood derived parameters. For the bivariate maximum likelihood estimation the marginal distributions have very similar Wasserstein distances; in the center panels: the dashed and solid colored lines are visually hardly separable. As suggested by the individual panels of Figures 6.1 and 6.2, convergence in the Gaussian case is much faster compared to the non-Gaussian case. The right column of Figure 6.4 illustrates the joint asymptotic normality of the range parameter estimators by maximum likelihood and cross validation. The gray lines there illustrate Theorem 7 and are Wasserstein distances for linear combinations of the range estimates by maximum likelihood and cross validation, i.e., $\lambda \hat{\rho}_{\text{ML}} + (1 - \lambda) \hat{\rho}_{\text{CV}}$ for $\lambda = j/10$, $j = 1, \dots, 9$. The highly skewed distribution of the cross

validation-estimated range parameter for Gaussian processes is clearly visible. In the non Gaussian case, the effect of the skewness is less pronounced since the maximum likelihood is skewed as well.

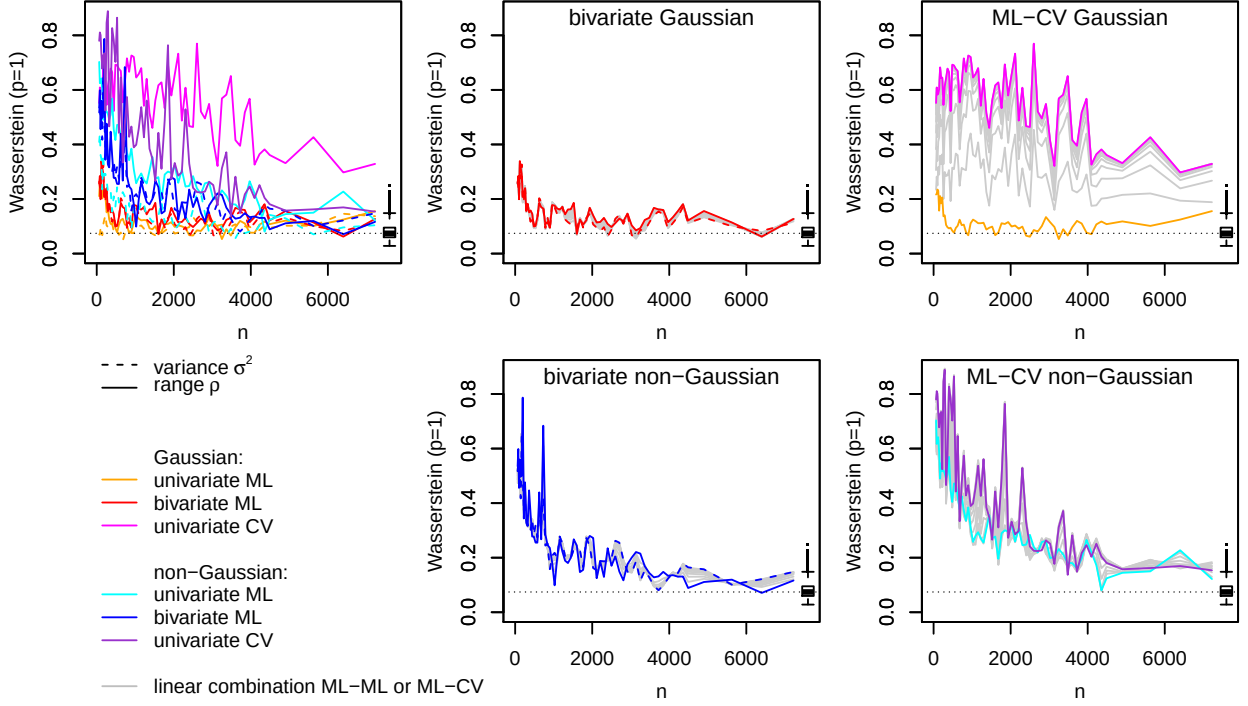


Figure 6.4 – Wasserstein distance ($p = 1$). Top left: marginal for each parameter. Center panels: bivariate estimation of range and variance by maximum likelihood as in Theorem 4 and linear combinations thereof; right panels: univariate estimation of the range parameter by maximum likelihood and cross validation and with linear combinations of these two estimators as in Theorem 7. Top middle and right panels: Gaussian cases. Lower row panels: non-Gaussian case. The colors and line styles follow those in Figure 6.3. The gray lines are Wasserstein distances for estimates based on linear combinations of maximum likelihood (center column) and of maximum likelihood and cross validation (right column). The panels are based on $N = 250$ replicates that did not reveal any convergence issues. The small boxplot on the right in each panel shows the Wasserstein distance for sample size $n = 250$ of 10000 realizations of $\mathcal{N}[0, 1]$; the horizontal dotted line shows the median thereof.

6.8 Conclusion

We have shown that the covariance parameters of transformed Gaussian processes can be estimated by cross validation and Gaussian maximum likelihood, with the same rate of convergence as in the case of non-transformed Gaussian processes. In particular, Gaussian maximum likelihood works well asymptotically, despite the fact that the observations do not have a Gaussian distribution. Hence Gaussian maximum likelihood is here robust with respect to non-Gaussian data. This provides the first step of a theoretical validation of the use of Gaussian maximum likelihood in frequent cases where the data are non-Gaussian.

In future research, it would be interesting to extend the results of this paper to other classes of non-Gaussian random fields rather than only transformed Gaussian processes. In addition, the asymptotic analysis of estimators of the transformation of transformed Gaussian processes is of great interest.

Acknowledgments

RF acknowledges the support of the Swiss National Science Foundation SNSF-175529. FB acknowledges the support of a PEPS from the French Centre national de la recherche scientifique. This research was partly undertaken within the RISCOPE ANR project. The authors are indebted to two anonymous referees, whose comments led to an improved exposition and more generality of the results.

Appendix 6.A Proofs

6.A.1 Technical results

Lemma 6. *Let $q \in \mathbb{N}$ be fixed. Let $g : \mathbb{R}^q \rightarrow \mathbb{R}^+$ be fixed and satisfy $g(x) \leq C_{\sup} \exp(C_{\sup}|x|)$. Let W be a Gaussian vector of dimension q . Then $\mathbb{E}[g(W)] < \infty$.*

Proof. Without loss of generality, we can assume that W_i has variance 1 for $i = 1, \dots, q$. We let w_i be the mean of W_i for $i = 1, \dots, q$. We have, for $t > 1$,

$$\begin{aligned} \mathbb{P}(g(W) \geq t) &\leq \mathbb{P}(C_{\sup} \exp(C_{\sup} \max_{i=1, \dots, q} |W_i|) \geq t) \\ &\leq \sum_{i=1}^q \mathbb{P}(C_{\sup} \exp(C_{\sup} |W_i|) \geq t) \\ &= \sum_{i=1}^q \mathbb{P}(|W_i| \geq (1/C_{\sup}) \log(t/C_{\sup})) \\ &\leq 2 \sum_{i=1}^q \mathbb{P}(W \geq (1/C_{\sup}) \log(t/C_{\sup}) - |w_i|), \end{aligned}$$

where $W \sim \mathcal{N}[0, 1]$. From the Gaussian tail inequality, we obtain, for

$$t \geq C_{\sup} \exp(C_{\sup} (\max_{i=1, \dots, q} |w_i| + 1)),$$

that

$$\mathbb{P}(g(W) \geq t) \leq \frac{2q}{\sqrt{2\pi}} \exp\left(-(1/2)\left((1/C_{\sup}) \log(t/C_{\sup}) - \max_{i=1, \dots, q} |w_i|\right)^2\right).$$

The function of t above is clearly summable as $t \rightarrow +\infty$. Hence, we have $\mathbb{E}[g(W)] = \int_0^\infty \mathbb{P}(g(W) \geq t) < +\infty$. $\square$

Lemma 7. *Let X be a centered Gaussian process with covariance function k_X satisfying Condition 2. Let F satisfy Condition 3. Let W be the spatial process $F(X(\cdot))$ and assume that W is centered. Let $(x_i)_{i \in \mathbb{N}}$ satisfy Condition 1.*

Then, we have, for any $r_1, r_2 \in \mathbb{N}$ and $\Delta \geq 0$,

$$\sup_{\substack{i_1, \dots, i_{r_1} \in \mathbb{N} \\ j_1, \dots, j_{r_2} \in \mathbb{N} \\ \min_{a=1, \dots, r_1, b=1, \dots, r_2} |x_{i_a} - x_{j_b}| \geq \Delta}} \left| \text{Cov}(W(x_{i_1}) \dots W(x_{i_{r_1}}), W(x_{j_1}) \dots W(x_{j_{r_2}})) \right| \leq C_{\sup} e^{-C_{\inf} \Delta},$$

where $C_{\sup}$ and $C_{\inf}$ depend on r_1, r_2 but not on Δ .

Proof. Let $\Delta \geq 0$, $i_1, \dots, i_{r_1} \in \mathbb{N}$ and $j_1, \dots, j_{r_2} \in \mathbb{N}$ such that $\min_{a=1, \dots, r_1, b=1, \dots, r_2} |x_{i_a} - x_{j_b}| \geq \Delta$.

Let $\epsilon_3 \sim \mathcal{N}[0, I_{r_1}]$ and $\epsilon_4 \sim \mathcal{N}[0, I_{r_2}]$, ϵ_3 and ϵ_4 being independent. Let R be the $r_2 \times r_1$ matrix $(k_X(x_{j_a}, x_{i_b}))_{a,b}$, let C_1 be the $r_1 \times r_1$ matrix $(k_X(x_{i_a}, x_{i_b}))_{a,b}$ and let C_2 be the $r_2 \times r_2$ matrix $(k_X(x_{j_a}, x_{j_b}))_{a,b}$. Let $M = RC_1^{-1}$. Let K be a matrix square root of $C_2 - RC_1^{-1}R^\top$. Let K_1 be the unique symmetric matrix square root of C_1 .

Then the vector $((K_1\epsilon_3)^\top, (MK_1\epsilon_3 + K\epsilon_4)^\top)$ has the same distribution as

$$(X(x_{i_1}), \dots, X(x_{i_{r_1}}), X(x_{j_1}), \dots, X(x_{j_{r_2}})).$$

For $i = 1, 2$, for $x = (x_1, \dots, x_{r_i}) \in \mathbb{R}^{r_i}$, let $f_i(x) = F(x_1) \cdots F(x_{r_i}) \in \mathbb{R}$.

Then we have

$$\text{Cov}(W(x_{i_1}) \cdots W(x_{i_{r_1}}), W(x_{j_1}) \cdots W(x_{j_{r_2}})) = \text{Cov}(f_1(K_1\epsilon_3), f_2(MK_1\epsilon_3 + K\epsilon_4)).$$

By a Taylor expansion, there exists a random vector ϵ_5 belonging to the segment with endpoints $K\epsilon_4$ and $MK_1\epsilon_3 + K\epsilon_4$ such that, with $G_2(\epsilon_5)$ the gradient column vector of f_2 at ϵ_5 , we have

$$f_2(MK_1\epsilon_3 + K\epsilon_4) = f_2(K\epsilon_4) + (MK_1\epsilon_3)^\top G_2(\epsilon_5).$$

This yields

$$\begin{aligned} |\text{Cov}(W(x_{i_1}) \cdots W(x_{i_{r_1}}), W(x_{j_1}) \cdots W(x_{j_{r_2}}))| &= |\text{Cov}(f_1(K_1\epsilon_3), \epsilon_3^\top K_1^\top M^\top G_2(\epsilon_5))| \\ &\leq \sqrt{\mathbb{E}[f_1^2(K_1\epsilon_3)]} \sqrt{\mathbb{E}[(\epsilon_3^\top K_1^\top M^\top G_2(\epsilon_5))^2]}. \end{aligned}$$

From Condition 2 ii) and from the equivalence of norms, we obtain $\|M\|_{op} \leq C_{\text{sup}} e^{-C_{\text{inf}}\Delta}$ and $\|K_1\|_{op} \leq C_{\text{sup}}$, where C_{sup} and C_{inf} do not depend on $i_1, \dots, i_{r_1}, j_1, \dots, j_{r_2}, \Delta$. By equivalence of norms, we then obtain, with C_{sup} and C_{inf} not depending on $i_1, \dots, i_{r_1}, j_1, \dots, j_{r_2}, \Delta$,

$$|\text{Cov}(W(x_{i_1}) \cdots W(x_{i_{r_1}}), W(x_{j_1}) \cdots W(x_{j_{r_2}}))| \leq C_{\text{sup}} e^{-C_{\text{inf}}\Delta} \sqrt{\mathbb{E}[f_1^2(K_1\epsilon_3)] \mathbb{E}[(\|\epsilon_3\| \|G_2(\epsilon_5)\|)^2]}.$$

Now,

$$\|\epsilon_5\| \leq \|MK_1\epsilon_3\| + \|K\epsilon_4\| \leq C_{\text{sup}} (\|\epsilon_3\| + \|\epsilon_4\|),$$

from Condition 2 ii), where, again, C_{sup} does not depend on $i_1, \dots, i_{r_1}, j_1, \dots, j_{r_2}, \Delta$. Furthermore, $\|K_1\epsilon_3\| \leq C_{\text{sup}}\|\epsilon_3\|$. Eventually, we have

$$\begin{aligned} |\text{Cov}(W(x_{i_1}) \cdots W(x_{i_{r_1}}), W(x_{j_1}) \cdots W(x_{j_{r_2}}))| \\ \leq C_{\text{sup}} \exp(-C_{\text{inf}}\Delta) \sqrt{\mathbb{E}[f_1^2(K_1\epsilon_3)]} \sqrt{\mathbb{E}\left[\left(\|\epsilon_3\| \sup_{\|x\| \leq C_{\text{sup}}(\|\epsilon_3\| + \|\epsilon_4\|)} \|G_2(x)\|\right)^2\right]}. \end{aligned}$$

From Condition 3 i), we have $|f_1(K_1x)| \leq C_{\text{sup}} e^{C_{\text{sup}}\|x\|}$ and $\|G_2(x)\| \leq C_{\text{sup}} e^{C_{\text{sup}}\|x\|}$ where C_{sup} does not depend on x and $i_1, \dots, i_{r_1}, j_1, \dots, j_{r_2}, \Delta$. Hence the above square roots are finite from Lemma 6 and do not depend on $i_1, \dots, i_{r_1}, j_1, \dots, j_{r_2}$ and Δ . This concludes the proof. $\square$

Lemma 8. Consider the setting of Section 6.3.1, that is k_Z satisfies Condition 2 and T satisfies Condition 3. For $n \in \mathbb{N}$ and $i, j = 1, \dots, n$ we have

$$\sum_{k,l=1,\dots,n} |\text{Cov}(y_i y_j, y_k y_l)| \leq C_{\text{sup}},$$

where C_{sup} does not depend on n, i, j .

Proof. We let $d(a, (b, c)) = \min(|a - b|, |a - c|)$ for $a, b, c \in \mathbb{R}^d$. It is enough to show that

$$|\text{Cov}(y_i y_j, y_k y_l)| \leq C_{\text{sup}} \exp\left(-C_{\text{inf}} \max\left(d(s_k, (s_i, s_j)), d(s_l, (s_i, s_j))\right)\right). \quad (6.13)$$

Indeed, let, for $t \geq 0$, $N_{i,j,t}$ be the number of pairs (k, l) , with $1 \leq k, l \leq n$ such that

$$t \leq \max(d(s_k, (s_i, s_j)), d(s_l, (s_i, s_j))) \leq t + 1.$$

From Condition 1, we can show that we have $\sup_{n \in \mathbb{N}, i, j=1,\dots,n} N_{i,j,t} \leq C_{\text{sup}} t^{2d}$. Hence, we have

$$\sum_{i,j=1}^n \exp\left(-C_{\text{inf}} \max\left(d(s_k, (s_i, s_j)), d(s_l, (s_i, s_j))\right)\right) \leq \sum_{k=0}^{+\infty} C_{\text{sup}} (k+1)^{2d} \exp(-C_{\text{inf}} k) < +\infty.$$

Thus, (6.13) implies the result of the lemma and it suffices to prove (6.13).

Let $i, j, k, l \in \{1, \dots, n\}$ and let $\Delta = \max(d(s_k, (s_i, s_j)), d(s_l, (s_i, s_j)))$. By symmetry, we can consider that $d(s_k, (s_i, s_j)) = \Delta$.

If $|s_k - s_l| \leq |s_i - s_l|$ and $|s_k - s_l| \leq |s_j - s_l|$, then $|s_i - s_l| \geq \Delta/2$ and $|s_j - s_l| \geq \Delta/2$. Hence, we can apply Lemma 7 with distance $\Delta/2$ to obtain $|\text{Cov}(y_k y_l, y_i y_j)| \leq C_{\text{sup}} \exp(-C_{\text{inf}} \Delta/2)$, where C_{sup} and C_{inf} do not depend on n, i, j, k, l, Δ .

If $|s_i - s_l| \leq |s_k - s_l|$ and $|s_j - s_l| \leq |s_k - s_l|$, then $|s_k - s_l| \geq \Delta/2$. We then have

$$\begin{aligned} \text{Cov}(y_i y_j, y_k y_l) &= \mathbb{E}[y_i y_j y_k y_l] - \mathbb{E}[y_i y_j] \mathbb{E}[y_k y_l] \\ &= \text{Cov}(y_i y_j y_l, y_k) - \text{Cov}(y_i, y_j) \text{Cov}(y_k, y_l) \end{aligned} \quad (6.14)$$

since it is assumed that Y has zero-mean. In (6.14), the first and third covariances are bounded in absolute value by $C_{\text{sup}} \exp(-C_{\text{inf}} \Delta/2)$ from Lemma 7, because $|s_k - s_l| \geq \Delta/2$, $|s_k - s_i| \geq \Delta/2$ and $|s_k - s_j| \geq \Delta/2$. Hence we have $|\text{Cov}(y_k y_l, y_i y_j)| \leq C_{\text{sup}} \exp(-C_{\text{inf}} \Delta/2)$, where C_{sup} and C_{inf} do not depend on n, i, j, k, l, Δ .

If $|s_j - s_l| \leq |s_k - s_l|$ and $|s_i - s_l| \leq |s_k - s_l|$, we obtain the same bound by symmetry. We have thus considered all possible cases and the proof of (6.13) is concluded. $\square$

In the context of Theorem 2, the following lemma provides an approximation of V_n , based on replacing A by a sparse matrix. We remark that a similar approximation was shown in a time series context in [163]. Nevertheless, we find that our assumptions on the random field Y are more transparent and interpretable than the assumptions in [163], where cumulants are used. Because of these differences of assumptions, our proof of the following lemma differs from that in [163].

Lemma 9. Let, for $K, n \in \mathbb{N}$, $A^{(K)}$ be the $n \times n$ matrix defined by

$$A_{i,j}^{(K)} = A_{i,j} 1_{|s_i - s_j| \leq K}.$$

Then, under the same assumptions as in Lemma 8, we have

$$\sup_{n \in \mathbb{N}} n \operatorname{Var} \left(\frac{1}{n} y^\top A y - \frac{1}{n} y^\top A^{(K)} y \right) \rightarrow_{K \rightarrow \infty} 0.$$

Proof. For any $K, n \in \mathbb{N}$ we have

$$n \operatorname{Var} \left(\frac{1}{n} y^\top A y - \frac{1}{n} y^\top A^{(K)} y \right) = \frac{1}{n} \sum_{i,j,k,l=1}^n (A - A^{(K)})_{i,j} (A - A^{(K)})_{k,l} \operatorname{Cov}(y_i y_j, y_k y_l)$$

We observe that $|(A - A^{(K)})_{k,l}|$ is equal to 0 or is smaller than $C_{\sup}/(1 + K^{d+C_{\inf}})$ by assumption. Hence we have

$$\begin{aligned} n \operatorname{Var} \left(\frac{1}{n} y^\top A y - \frac{1}{n} y^\top A^{(K)} y \right) &\leq C_{\sup} \frac{1}{1 + K^{d+C_{\inf}}} \frac{1}{n} \sum_{i,j,k,l=1}^n |(A - A^{(K)})_{i,j}| |\operatorname{Cov}(y_i y_j, y_k y_l)| \\ &\leq C_{\sup} \frac{1}{1 + K^{d+C_{\inf}}} \frac{1}{n} \sum_{i,j=1}^n |(A - A^{(K)})_{i,j}| \\ &\leq C_{\sup} \frac{1}{1 + K^{d+C_{\inf}}} \max_{i=1,\dots,n} \sum_{j=1}^n \frac{1}{1 + |s_i - s_j|^{d+C_{\inf}}} \\ &\leq C_{\sup} \frac{1}{1 + K^{d+C_{\inf}}}, \end{aligned}$$

where we have used Lemma 8 and where we have observed that $|(A - A^{(K)})_{i,j}|$ is equal to 0 or is smaller than $C_{\sup}/(1 + |s_i - s_j|^{d+C_{\inf}})$. We have also used Lemma 4 in [140] for the last inequality above. All the above constants $C_{\sup}$ and $C_{\inf}$ naturally do not depend on n , so the lemma is proved. $\square$

Lemma 10. Consider the setting of Section 6.3.1, that is k_Z satisfies Condition 2 and T satisfies Condition 3. Let $a, b \in \mathbb{N}$. For $i \in \{1, \dots, a\}$, let $\alpha_i \in \mathbb{N}$ and let $I(i, 1), \dots, I(i, \alpha_i) \in \{1, \dots, n\}$. For $j \in \{1, \dots, b\}$, let $\beta_j \in \mathbb{N}$ and let $J(j, 1), \dots, J(j, \beta_j) \in \{1, \dots, n\}$. For $i = 1, \dots, a$, let f_i be a function from $\mathbb{R}^{\alpha_i}$ to $\mathbb{R}$. For $j = 1, \dots, b$, let g_j be a function from $\mathbb{R}^{\beta_j}$ to $\mathbb{R}$. For $i = 1, \dots, a$, let $v^{(i)} = f_i(Z(s_{I(i,1)}), \dots, Z(s_{I(i,\alpha_i)}))$. For $j = 1, \dots, b$, let $w^{(j)} = g_j(Z(s_{J(j,1)}), \dots, Z(s_{J(j,\beta_j)}))$. Let

$$\begin{aligned} &\alpha(\{v^{(1)}, \dots, v^{(a)}\}, \{w^{(1)}, \dots, w^{(b)}\}) \\ &= \sup \left\{ \left| \mathbb{P}(A \cap B) - \mathbb{P}(A)\mathbb{P}(B) \right|; A \in \sigma(\{v^{(1)}, \dots, v^{(a)}\}), B \in \sigma(\{w^{(1)}, \dots, w^{(b)}\}) \right\}, \end{aligned} \quad (6.15)$$

where, for any set of random variables $\{\epsilon_1, \dots, \epsilon_r\}$, $\sigma(\{\epsilon_1, \dots, \epsilon_r\})$ is the sigma algebra generated by the random variables $\{\epsilon_1, \dots, \epsilon_r\}$.

Let

$$\Delta = \inf_{\substack{i \in \{1, \dots, a\} \\ j \in \{1, \dots, b\} \\ \tilde{i} \in \{1, \dots, \alpha_i\} \\ \tilde{j} \in \{1, \dots, \beta_j\}}} |s_{I(i, \tilde{i})} - s_{J(j, \tilde{j})}|.$$

Then, we have

$$\alpha(\{v^{(1)}, \dots, v^{(a)}\}, \{w^{(1)}, \dots, w^{(b)}\}) \leq C_{\sup} e^{-C_{\inf} \Delta},$$

where $C_{\sup}$ and $C_{\inf}$ may depend on $a, \alpha_1, \dots, \alpha_a$ but do not depend on $b, (J(j, \tilde{j}))_{j=1, \dots, b, \tilde{j}=1, \dots, \beta_j}$ and Δ .

Proof. Let $\mathcal{I} = \{I(i, \tilde{i}); i = 1, \dots, a, \tilde{i} = 1, \dots, \alpha_i\}$ and let $\mathcal{J} = \{J(j, \tilde{j}); j = 1, \dots, b, \tilde{j} = 1, \dots, \beta_j\}$. In (6.15), any of the events A (resp. B) is an event defined on the set of random variables $\{Z(s_i)\}_{i \in \mathcal{I}}$ (resp. $\{Z(s_j)\}_{j \in \mathcal{J}}$). We thus obtain

$$\begin{aligned} & \alpha(\{v^{(1)}, \dots, v^{(a)}\}, \{w^{(1)}, \dots, w^{(b)}\}) \\ & \leq \alpha(\{Z(s_i)\}_{i \in \mathcal{I}}, \{Z(s_j)\}_{j \in \mathcal{J}}) \\ & := \sup \left\{ |\mathbb{P}(A \cap B) - \mathbb{P}(A)\mathbb{P}(B)|; A \in \sigma(\{Z(s_i)\}_{i \in \mathcal{I}}), B \in \sigma(\{Z(s_j)\}_{j \in \mathcal{J}}) \right\}. \end{aligned}$$

Let $I_1 < \dots < I_{\bar{\alpha}}$ be such that $\{I_1, \dots, I_{\bar{\alpha}}\} = \mathcal{I}$ and let $v_Z = (Z(s_{I_1}), \dots, Z(s_{I_{\bar{\alpha}}}))^\top$. Let $J_1 < \dots < J_{\bar{\beta}}$ be such that $\{J_1, \dots, J_{\bar{\beta}}\} = \mathcal{J}$ and let $w_Z = (Z(s_{J_1}), \dots, Z(s_{J_{\bar{\beta}}}))^\top$. From Lemma 1 in Section 2.1 of [164], we have

$$\begin{aligned} & \alpha(\{Z(s_i)\}_{i \in \mathcal{I}}, \{Z(s_j)\}_{j \in \mathcal{J}}) \\ & \leq \sup \left\{ |\text{Cov}(v_Z^\top v_Z, w_Z^\top w_Z)|; \text{Var}(v_Z^\top v_Z) = 1, \text{Var}(w_Z^\top w_Z) = 1 \right\}. \end{aligned} \quad (6.16)$$

Let v and w be vectors belonging to the set in (6.16). The smallest eigenvalues of the covariance matrices of v_Z and w_Z are larger than a constant $C_{\inf}$, not depending on $\mathcal{I}$ and $\mathcal{J}$, since k_Z satisfies Condition 2. Thus we have

$$1 = \text{Var}(v_Z^\top v_Z) = v_Z^\top \text{Cov}(v_Z) v_Z \geq C_{\inf} \|v_Z\|^2.$$

It follows that $\|v_Z\|^2 \leq C_{\sup}$, where $C_{\sup}$ does not depend on $\mathcal{I}$, $\mathcal{J}$ and Δ . Similarly $\|w_Z\|^2 \leq C_{\sup}$.

We have

$$\begin{aligned} \text{Cov}^2(v_Z^\top v_Z, w_Z^\top w_Z) & \leq \|\text{Cov}(v_Z^\top v_Z, w_Z)\|^2 \|w_Z\|^2 \\ & \leq C_{\sup} \sum_{j=1, \dots, \bar{\beta}} \text{Cov}(v_Z^\top v_Z, Z(s_{J_j}))^2 \\ & \leq C_{\sup} \|v_Z\|^2 \sum_{i=1, \dots, \bar{\alpha}} \sum_{j=1, \dots, \bar{\beta}} \text{Cov}(Z(s_{I_i}), Z(s_{J_j}))^2 \\ & \leq C_{\sup} \sum_{i=1, \dots, \bar{\alpha}} \sum_{\substack{j=1, \dots, n \\ |s_j - s_i| \geq \Delta}} e^{-C_{\inf} |s_{I_i} - s_j|}, \end{aligned}$$

by definition of Δ , since k_Z satisfies Condition 2 and where $C_{\sup}$ and $C_{\inf}$ do not depend on b , $\mathcal{J}$ and Δ . For any $i \in \{1, \dots, n\}$, the number of indices $j \in \{1, \dots, n\}$ such that $\tilde{\Delta} \leq |s_i - s_j| \leq \tilde{\Delta} + 1$ is smaller than $C_{\sup} \tilde{\Delta}^d$, from Condition 1 and where $C_{\sup}$ only depends on d . This yields

$$\begin{aligned}
\text{Cov}^2(v^\top v_Z, w^\top w_Z) &\leq C_{\text{sup}}(\alpha_1 + \dots + \alpha_a) \sum_{k=0}^{+\infty} C_{\text{sup}}(\Delta + k)^d e^{-C_{\text{inf}}|\Delta+k|} \\
&\leq C_{\text{sup}} e^{-C_{\text{inf}}|\Delta|/2} (\alpha_1 + \dots + \alpha_a) \sum_{k=1}^{+\infty} (\Delta + k)^d e^{-C_{\text{inf}}|\Delta+k|/2} \\
&\leq C_{\text{sup}} e^{-C_{\text{inf}}|\Delta|}
\end{aligned}$$

where the different C_{sup} and C_{inf} do not depend on b , $\mathcal{J}$ and Δ . This concludes the proof from (6.16). $\square$

Lemma 11. Consider a sequence $(x_i)_{i \in \mathbb{N}}$ of points in $\mathbb{R}^d$ satisfying Condition 1. Let $\tau > 0$ be fixed. For $n \in \mathbb{N}$, let $(A_\theta)_{\theta \in \Theta}$ and $(B_\theta)_{\theta \in \Theta}$ be families of $n \times n$ matrices. Assume that for all $n \in \mathbb{N}$, $i, j = 1, \dots, n$ and $\theta \in \Theta$,

$$|(A_\theta)_{i,j}| \leq \frac{C_{\text{sup}}}{1 + |x_i - x_j|^{d+\tau}} \quad \text{and} \quad |(B_\theta)_{i,j}| \leq \frac{C_{\text{sup}}}{1 + |x_i - x_j|^{d+\tau}}$$

where C_{sup} does not depend on n, i, j, θ . Then we have for all $n \in \mathbb{N}$, $i, j = 1, \dots, n$ and $\theta \in \Theta$,

$$|(A_\theta B_\theta)_{i,j}| \leq \frac{C_{\text{sup}}}{1 + |x_i - x_j|^{d+\tau}}$$

where C_{sup} does not depend on n, i, j, θ .

Proof. We have,

$$\begin{aligned}
|(A_\theta B_\theta)_{i,j}| &= \left| \sum_{\ell=1}^n (A_\theta)_{i,\ell} (B_\theta)_{\ell,j} \right| \\
&\leq \sum_{\ell=1}^n \frac{C_{\text{sup}}}{1 + |x_i - x_\ell|^{d+\tau}} \frac{C_{\text{sup}}}{1 + |x_j - x_\ell|^{d+\tau}} \\
&\leq \sum_{\substack{\ell=1, \dots, n \\ |x_i - x_\ell| \leq |x_j - x_\ell|}} \frac{C_{\text{sup}}}{1 + |x_i - x_\ell|^{d+\tau}} \frac{C_{\text{sup}}}{1 + (|x_i - x_j|/2)^{d+\tau}} \\
&\quad + \sum_{\substack{\ell=1, \dots, n \\ |x_j - x_\ell| \leq |x_i - x_\ell|}} \frac{C_{\text{sup}}}{1 + (|x_i - x_j|/2)^{d+\tau}} \frac{C_{\text{sup}}}{1 + |x_j - x_\ell|^{d+\tau}} \\
&\leq C_{\text{sup}} \frac{C_{\text{sup}}}{1 + |x_i - x_j|^{d+\tau}} \max_{a=1, \dots, n} \sum_{b=1}^n \frac{1}{1 + |x_a - x_b|^{d+\tau}} \\
&\leq \frac{C_{\text{sup}}}{1 + |x_i - x_j|^{d+\tau}}
\end{aligned}$$

from Lemma 4 in [140]. $\square$

Lemma 12. Consider the setting of Section 6.5.1. Under Conditions 4 and 5, we have

$$\sup_{\theta \in \Theta} \lambda_1(R_\theta^{-1}) \leq C_{\text{sup}}, \quad (6.17)$$

$$\sup_{\theta \in \Theta} \lambda_1(R_\theta) \leq C_{\text{sup}}, \quad (6.18)$$

and

$$\sup_{\substack{\theta \in \Theta \\ \ell=1,2,3 \\ i_1, \dots, i_\ell=1, \dots, p}} \lambda_1\left(\frac{\partial R_\theta}{\partial \theta_{i_1} \dots \partial \theta_{i_\ell}}\right) \leq C_{\text{sup}}. \quad (6.19)$$

Proof. Conditions 1, 4 and 5 imply (6.17) from Theorem 5 in [145]. Conditions 1 and 4 imply (6.18) and (6.19) from Lemma 6 in [140]. $\square$

Lemma 13. *Consider the setting of Section 6.5.1. Under Conditions 4 and 5, we have, for $n \in \mathbb{N}$ and $i, j \in \{1, \dots, n\}$,*

$$\sup_{\theta \in \Theta} |(R_\theta^{-1})_{i,j}| \leq \frac{C_{\text{sup}}}{1 + |s_i - s_j|^{d+C_{\text{inf}}}},$$

where C_{sup} and C_{inf} do not depend on n, i, j, θ .

Proof. One can show that the proof of Theorem 1 can be made uniform over $\theta \in \Theta$, thus yielding Lemma 13. $\square$

Lemma 14. *Consider the setting of Section 6.5.1. Under Conditions 4 and 5, we have,*

$$\inf_{\theta \in \Theta} \lambda_n(R_\theta) \geq C_{\text{inf}}, \quad (6.20)$$

$$\inf_{\theta \in \Theta} \lambda_n(\text{diag}(R_\theta^{-1})) \geq C_{\text{inf}}. \quad (6.21)$$

Proof. Equation (6.20) holds from (6.17). Then, (6.21) follows from (6.20) as in Lemma D.6 in [122]. $\square$

6.A.2 Proofs of the main results

Proof of Lemma 1. As a special case of Lemma 7, k' satisfies Condition 2 i).

Let us now show that k' satisfies Condition 2 ii). Let (x_i) satisfy Condition 1. Let $n \in \mathbb{N}$ be fixed and let R be the $n \times n$ covariance matrix $k(x_i - x_j)_{i,j=1, \dots, n}$. Let $a_1, \dots, a_n \in \mathbb{R}$. We have

$$\sum_{i,j=1}^n a_i a_j R_{i,j} = \text{Var}\left(\sum_{i=1}^n a_i F(X(x_i))\right).$$

We now let $z = (X(x_1), \dots, X(x_n))^\top$ and $g : \mathbb{R}^n \rightarrow \mathbb{R}$ be defined by $g(t) = \sum_{i=1}^n a_i F(t_i)$. The gradient of g at t is $\nabla g(t) = (a_1 F'(t_1), \dots, a_n F'(t_n))^\top$. We use the inequality in Theorem 3.7 in [165]. This yields

$$\sum_{i,j=1}^n a_i a_j R_{i,j} \geq \mathbb{E}[\nabla g(z)]^\top \text{Cov}(z) \mathbb{E}[\nabla g(z)].$$

From Condition 2 ii), we have $\lambda_1(\text{Cov}(z)) \geq C_{\text{inf}}$. This yields

$$\sum_{i,j=1}^n a_i a_j R_{i,j} \geq C_{\text{inf}} \sum_{i=1}^n \mathbb{E}^2[(\nabla g(z))_i]$$

$$\begin{aligned}
&= C_{\inf} \sum_{i=1}^n a_i^2 \mathbb{E}[F'(z_i)]^2 \\
&= C_{\inf} \left(\sum_{i=1}^n a_i^2 \right) \mathbb{E}[F'(z_1)]^2.
\end{aligned}$$

From Condition 3, the above expectation is non-zero, which concludes the proof. $\square$

Proof of Lemma 2. The fact that k satisfies Condition 2 ii) follows from Theorem 4 in [145].

Let us now consider the case where F is defined by $F(x) = x^{2r} + u$. Since we consider a covariance function we can assume that $u = 0$ without loss of generality. Assume also that $k(0) = 1$ without loss of generality. From Lemma 7, k' satisfies Condition 2 i). Let us show that Condition 2 ii) is satisfied. Let $a, b \in \mathbb{R}^d$, let $c = k(a - b)$ and let $\lambda = (1 - c^2)^{1/2}$. With $(A_1, A_2) \sim \mathcal{N}[0, I_2]$, we have

$$\begin{aligned}
k'(a - b) &= \text{Cov}(A_1^{2r}, (cA_1 + \lambda A_2)^{2r}) \\
&= \text{Cov}\left(A_1^{2r}, \sum_{i=0}^{2r} \binom{2r}{i} c^i \lambda^{2r-i} A_1^i A_2^{2r-i}\right) \\
&= \sum_{i=0}^{2r} \binom{2r}{i} c^i \lambda^{2r-i} \text{Cov}(A_1^{2r}, A_1^i A_2^{2r-i}).
\end{aligned}$$

By independence of A_1 and A_2 , we obtain, for $i = 0, \dots, 2r$,

$$\text{Cov}(A_1^{2r}, A_1^i A_2^{2r-i}) = \mathbb{E}[A_2^{2r-i}] (\mathbb{E}[A_1^{2r+i}] - \mathbb{E}[A_1^{2r}] \mathbb{E}[A_1^i]). \quad (6.22)$$

From Isserlis' theorem, one can show that (6.22) is zero if i is odd and is strictly positive if i is even. As a consequence, we have

$$k'(a - b) = \sum_{i=0}^r \alpha_i k(a - b)^{2i}$$

with $\alpha_1, \dots, \alpha_r > 0$. Hence, the Fourier transform of k' is a linear combination of multiple convolutions of the Fourier transform of k , with strictly positive components. Since the Fourier transform of k is strictly positive everywhere, then also the Fourier transform of k' is strictly positive everywhere. Hence, from Theorem 4 in [145], k' satisfies Condition 2 ii). $\square$

Proof of Theorem 1. Condition 1 and Lemma 6 in [140] imply that the spectral norms of R^{-1} and R are bounded functions of n . Let $C_{\sup} = \sup_n \lambda_1(R) < \infty$ and $C_{\inf} = \inf_n \lambda_n(R) > 0$.

We write

$$R^{-1} = \frac{1}{C_{\sup}} \left(I - \left(I - \frac{R}{C_{\sup}} \right) \right)^{-1} = \frac{1}{C_{\sup}} \sum_{\ell=0}^{\infty} \left(I - \frac{R}{C_{\sup}} \right)^{\ell}. \quad (6.23)$$

We remark that the above sum is well-defined because the eigenvalues of $I - R/C_{\sup}$ are between 0 and $1 - C_{\inf}/C_{\sup}$.

We denote $M = I - C_{\sup}^{-1}R$ and $h_{i,j} = |s_i - s_j|$. Let $1 \leq A < \infty$ and $a > 0$ be fixed such that $M_{i,j} \leq Ae^{-2ah_{i,j}}$. Let $\delta = \inf_{i,j \in \mathbb{N}, i \neq j} |s_i - s_j| > 0$ (Condition 1). Let $D < \infty$ be a constant such that $D \geq 1$ and, for any $L \geq \delta$ and $i \in \mathbb{N}$, the set $\{j \in \mathbb{N}; |s_i - s_j| \leq L\}$ has no more than $(D/2)L^d$ elements.

Let $0 < \mu < \infty$ be fixed. We show by induction over $\ell \in \mathbb{N}$ that there exists a constant $1 \leq \varphi < \infty$, depending on μ but not depending on ℓ, i, j , such that for $\ell \leq \mu \log(h_{i,j})$,

$$|(M^\ell)_{i,j}| \leq A^\ell \varphi^\ell D^\ell h_{i,j}^{d\ell-d} e^{-ah_{i,j}}. \quad (6.24)$$

In the case $\log(h_{i,j}) < 0$, there is nothing to prove in (6.24), so we consider i, j such that $\log(h_{i,j}) \geq 0$ when proving (6.24).

For $\ell = 1$, (6.24) holds. Assume that (6.24) holds for some $\ell \in \mathbb{N}$. We have

$$\begin{aligned} |(M^{\ell+1})_{i,j}| &= \sum_{r=1}^n (M^\ell)_{i,r} M_{j,r} \\ &\leq \sum_{r=1}^n A^\ell \varphi^\ell D^\ell h_{i,r}^{d\ell-d} e^{-ah_{i,r}} A e^{-2ah_{r,j}} \\ &= A^{\ell+1} \varphi^\ell D^\ell \sum_{r=1}^n h_{i,r}^{d\ell-d} e^{-ah_{i,r}} e^{-2ah_{r,j}}. \end{aligned}$$

Let now $B_i = \{x \in \mathbb{R}^d; |x - s_i| \leq |s_i - s_j|\}$ and $B_j = \{x \in \mathbb{R}^d; |x - s_j| \leq |s_i - s_j|\}$. From the triangle inequality we obtain

$$\begin{aligned} |(M^{\ell+1})_{i,j}| &\leq A^{\ell+1} \varphi^\ell D^\ell \sum_{r \in \mathbb{N}; s_r \in B_i \cup B_j} h_{i,r}^{d\ell-d} e^{-ah_{i,r}} e^{-2ah_{r,j}} \\ &\quad + A^{\ell+1} \varphi^\ell D^\ell \sum_{r \in \mathbb{N}; s_r \in B_i^c \cap B_j^c} h_{i,r}^{d\ell-d} e^{-ah_{i,r}} e^{-2ah_{r,j}} \\ &\leq A^{\ell+1} \varphi^\ell D^\ell 2(D/2) h_{i,j}^d h_{i,j}^{d\ell-d} e^{-ah_{i,j}} \\ &\quad + A^{\ell+1} \varphi^\ell D^\ell e^{-2ah_{i,j}} \sum_{r \in \mathbb{N}; s_r \in B_i^c} h_{i,r}^{d\ell-d} e^{-ah_{i,r}} \\ &\leq A^{\ell+1} \varphi^\ell D^{\ell+1} h_{i,j}^{d(\ell+1)-d} e^{-ah_{i,j}} \\ &\quad + A^{\ell+1} \varphi^\ell D^\ell e^{-ah_{i,j}} \left(e^{-ah_{i,j}} \sum_{r \in \mathbb{N}} Q r^{d\mu \log(h_{i,j})} e^{-a(r-1)} \right), \end{aligned}$$

where for the last inequality we let Qr^d be an upper bound on the cardinality of $\{b \in \mathbb{N}; |s_b - s_i| \in [r-1, r]\}$ for all $i \in \mathbb{N}$. The constant Q is finite and depends only on d and δ from Condition 1. We also let $\ell \leq \mu \log(h_{i,j})$ to show the last above inequality. Hence, in order to finish the proof of (6.24), it remains to show that the term $(.)$ in the above display is a bounded function of $h_{i,j}$, and to let $\varphi/2 \geq 1$ be a bound for the term $(.)$.

We have, for $h_{i,j}$ large enough, with $\lceil \cdot \rceil$ the integer ceiling,

$$\begin{aligned} \left(e^{-ah_{i,j}} \sum_{r \in \mathbb{N}} Q r^{d\mu \log(h_{i,j})} e^{-a(r-1)} \right) &= Q e^a e^{-ah_{i,j}} \sum_{r \in \mathbb{N}} r^{\lceil d\mu \log(h_{i,j}) \rceil} e^{-ar} \\ &= Q e^a e^{-ah_{i,j}} \left(\frac{2}{a} \right)^{\lceil d\mu \log(h_{i,j}) \rceil} \sum_{r \in \mathbb{N}} \left(\frac{ar}{2} \right)^{\lceil d\mu \log(h_{i,j}) \rceil} e^{-ar} \\ &\leq Q e^a e^{-ah_{i,j}} \left(\frac{2}{a} \right)^{\lceil d\mu \log(h_{i,j}) \rceil} \lceil d\mu \log(h_{i,j}) \rceil! \sum_{r \in \mathbb{N}} e^{ar/2} e^{-ar} \\ &= Q e^a e^{-ah_{i,j}} \left(\frac{2}{a} \right)^{\lceil d\mu \log(h_{i,j}) \rceil} \lceil d\mu \log(h_{i,j}) \rceil! \frac{1}{1 - e^{-ar/2}}. \end{aligned}$$

The above function of $h_{i,j}$ clearly goes to 0 as $h_{i,j}$ goes to ∞ . Thus, the above term $(.)$ is bounded and thus (6.24) is proved.

Coming back to (6.23), using (6.24) and using the triangle inequality, we obtain, letting $\Delta = 1 - C_{\inf}/C_{\sup}$, and for $h_{i,j}$ large enough,

$$\begin{aligned} |(R^{-1})_{i,j}| &\leq \left(\sum_{1 \leq \ell \leq \mu \log(h_{i,j})} A^\ell \varphi^\ell D^\ell h_{i,j}^{d\ell-d} e^{-ah_{i,j}} \right) + \sum_{\mu \log(h_{i,j}) \leq \ell \leq \infty} \Delta^\ell \\ &\leq \mu \log(h_{i,j}) (A\varphi D)^{\mu \log(h_{i,j})} h_{i,j}^{d\mu \log(h_{i,j})} e^{-ah_{i,j}} + \frac{\Delta^{\mu \log(h_{i,j})}}{1 - \Delta} \\ &= \mu \log(h_{i,j}) (A\varphi D)^{\mu \log(h_{i,j})} h_{i,j}^{d\mu \log(h_{i,j})} e^{-ah_{i,j}} + \frac{h_{i,j}^{\mu \log(\Delta)}}{1 - \Delta}. \end{aligned} \quad (6.25)$$

In the above display, for any $\tau < \infty$ in the statement of Theorem 1, we can choose μ such that $\mu \log(\Delta) \leq -d - \tau$. Then, it is clear that the first summand in (6.25) is also smaller than a constant (depending on τ) time $h_{i,j}^{-d-\tau}$. This concludes the proof of Theorem 1, since also $\sup_{n \in \mathbb{N}} \max_{i,j=1,\dots,n} |(R^{-1})_{i,j}|$ is bounded by $C_{\sup}$. $\square$

Proof of Theorem 2. We have

$$n \operatorname{Var}(V_n) = \frac{1}{n} \sum_{i,j,k,l=1}^n A_{i,j} A_{k,l} \operatorname{Cov}(y_i y_j, y_k y_l). \quad (6.26)$$

From Lemma 8, we obtain

$$\begin{aligned} n \operatorname{Var}(V_n) &\leq \frac{1}{n} C_{\sup} \sum_{i,j=1}^n |A_{i,j}| \\ &\leq C_{\sup} \max_{i=1,\dots,n} \sum_{j=1}^n \frac{1}{1 + |s_i - s_j|^{d+C_{\inf}}} \\ &\leq C_{\sup} \end{aligned}$$

from Lemma 4 in [140]. Hence, $n \operatorname{Var}(V_n)$ is bounded as $n \rightarrow \infty$.

Assume now that

$$d_w(\mathcal{L}_n, \mathcal{N}[0, n \operatorname{Var}(V_n)]) \not\rightarrow 0 \quad (6.27)$$

as $n \rightarrow \infty$. Because $n \operatorname{Var}(V_n)$ is bounded, there exists a subsequence $\phi(n)$ such that

$$d_w(\mathcal{L}_{\phi(n)}, \mathcal{N}[0, \phi(n) \operatorname{Var}(V_{\phi(n)})]) \not\rightarrow 0 \quad (6.28)$$

as $n \rightarrow \infty$ and $\phi(n) \operatorname{Var}(V_{\phi(n)}) \rightarrow V \in [0, \infty)$ as $n \rightarrow \infty$. It is then simple to show that this implies

$$d_w(\mathcal{L}_{\phi(n)}, \mathcal{N}[0, V]) \not\rightarrow 0 \quad (6.29)$$

as $n \rightarrow \infty$. If $V = 0$, then, from Chebyshev inequality, $\mathcal{L}_{\phi(n)}$ converges to a Dirac mass at zero and so (6.29) does not hold, yielding a contradiction.

Hence it remains to consider the case $\phi(n) \operatorname{Var}(V_{\phi(n)}) \rightarrow V \in (0, \infty)$ as $n \rightarrow \infty$ and where (6.29) holds.

To reach a contradiction, we will show that

$$\sqrt{\phi(n)} \left(\frac{V_{\phi(n)} - \mathbb{E}[V_{\phi(n)}]}{\sqrt{V}} \right) \rightarrow_{n \rightarrow \infty} \mathcal{N}[0, 1].$$

To simplify notations in the sequel, without loss of generality, we will consider that $\phi(n) = n$ and show that

$$\sqrt{n} \left(\frac{V_n - \mathbb{E}[V_n]}{\sqrt{V}} \right) \rightarrow_{n \rightarrow \infty} \mathcal{N}[0, 1], \quad (6.30)$$

where $n \text{Var}(V_n) \rightarrow V \in (0, \infty)$ as $n \rightarrow \infty$. From Slutsky's lemma it is sufficient to show that

$$\sqrt{n} \left(\frac{V_n - \mathbb{E}[V_n]}{\sqrt{n \text{Var}(V_n)}} \right) \rightarrow_{n \rightarrow \infty} \mathcal{N}[0, 1]. \quad (6.31)$$

For $K \geq 0$, let

$$V_n^{(K)} = \frac{1}{n} y^\top A^{(K)} y,$$

with the notation of Lemma 9. We have

$$\begin{aligned} & \sup_{n \in \mathbb{N}} \mathbb{E} \left[\left(\sqrt{n} \left(\frac{V_n - \mathbb{E}[V_n]}{\sqrt{n \text{Var}(V_n)}} \right) - \sqrt{n} \left(\frac{V_n^{(K)} - \mathbb{E}[V_n^{(K)}]}{\sqrt{n \text{Var}(V_n^{(K)})}} \right) \right)^2 \right] \\ & \leq 2 \sup_{n \in \mathbb{N}} \text{Var} \left(\sqrt{n} \left(\frac{V_n - V_n^{(K)}}{\sqrt{n \text{Var}(V_n)}} \right) \right) + 2 \sup_{n \in \mathbb{N}} \text{Var} \left(\sqrt{n} V_n^{(K)} \left(\frac{1}{\sqrt{n \text{Var}(V_n)}} - \frac{1}{\sqrt{n \text{Var}(V_n^{(K)})}} \right) \right) \\ & \rightarrow_{n \rightarrow \infty} 0 \end{aligned}$$

from Lemma 9 and because $V > 0$. Hence, from Theorem 4.2 in [166] (as in [163]), it is sufficient to show that there exists $L \in (0, \infty)$ such that for any fixed $K \geq L$, we have

$$\sqrt{n} \left(\frac{V_n^{(K)} - \mathbb{E}[V_n^{(K)}]}{\sqrt{n \text{Var}(V_n^{(K)})}} \right) \rightarrow^{\mathcal{L}} \mathcal{N}[0, 1]$$

as $n \rightarrow \infty$, in order to prove (6.31) and thus to conclude the proof. We remark that, because of Lemma 9, we have $\left| \liminf_{n \rightarrow \infty} n \text{Var}(V_n^{(K)}) - \liminf_{n \rightarrow \infty} n \text{Var}(V_n) \right|$ goes to 0 as $K \rightarrow \infty$. Hence, we may take L such that $\liminf_{n \rightarrow \infty} n \text{Var}(V_n^{(K)}) > 0$ for $K \geq L$. Hence, up to extracting a subsequence, it is sufficient to show

$$\sqrt{n} (V_n^{(K)} - \mathbb{E}[V_n^{(K)}]) \rightarrow^{\mathcal{L}} \mathcal{N}[0, V^{(K)}], \quad (6.32)$$

where $n \text{Var}(V_n^{(K)}) \rightarrow V^{(K)} > 0$ as $n \rightarrow \infty$. We have

$$\begin{aligned} \sqrt{n} (V_n^{(K)} - \mathbb{E}[V_n^{(K)}]) &= \frac{1}{\sqrt{n}} \sum_{i,j=1}^n (y_i y_j - \mathbb{E}[y_i y_j]) A_{i,j} 1_{|s_i - s_j| \leq K} \\ &= \frac{1}{\sqrt{n}} \sum_{i=1}^n \left(\sum_{\substack{j=1, \dots, n \\ |s_i - s_j| \leq K}} A_{i,j} (T(Z(s_i)) T(Z(s_j)) - \mathbb{E}[T(Z(s_i)) T(Z(s_j))]) \right) \\ &= \frac{1}{\sqrt{n}} \sum_{i=1}^n X_n(s_i), \end{aligned}$$

say, where X_n can be interpreted as a centered random field defined on $(s_i)_{i \in \mathbb{N}}$. We will now show that the sequence of random fields $(X_n)_{n \in \mathbb{N}}$ satisfies the conditions of Corollary 1 of [167].

We let, for $k, l \in \mathbb{N}$ and $r \geq 0$

$$\begin{aligned} \bar{\alpha}_{k,l}(r) = \sup_{n \in \mathbb{N}} \sup \bigg\{ & |\mathbb{P}(A \cap B) - \mathbb{P}(A)\mathbb{P}(B)|; \\ & A \in \sigma(X_n(s_{I_1}), \dots, X_n(s_{I_{\bar{k}}})) , B \in \sigma(X_n(s_{J_1}), \dots, X_n(s_{J_{\bar{l}}})) , \\ & \bar{k} \leq k, \bar{l} \leq l, I_1, \dots, I_{\bar{k}}, J_1, \dots, J_{\bar{l}} \in \{1, \dots, n\}, \min_{\bar{k}=1, \dots, \bar{k}, \bar{l}=1, \dots, \bar{l}} |s_{I_{\bar{k}}} - s_{J_{\bar{l}}}| \geq r \bigg\}. \end{aligned}$$

Let $N_K = \sup_{n \in \mathbb{N}} \max_{i=1, \dots, n} \sum_{j=1}^n 1_{|s_j - s_i| \leq K}$. Then $N_K \leq C_{\text{sup}}$, where C_{sup} depends only on K , d and δ from Condition 1. We remark that for $i = 1, \dots, n$, $X_n(s_i)$ is a function of the variables $Z(s_{I(n,i,1)}), \dots, Z(s_{I(n,i,\gamma(n,i))})$, with $\gamma(n,i) \leq N_K$ and with $|s_{I(n,i,\gamma)} - s_i| \leq K$ for $\gamma = 1, \dots, \gamma(n,i)$. Furthermore, for $|s_i - s_j| \geq r$ we have for $\gamma_i = 1, \dots, \gamma(n,i)$ and for $\gamma_j = 1, \dots, \gamma(n,j)$ that $|s_{I(n,i,\gamma_i)} - s_{I(n,j,\gamma_j)}| \geq r - 2K$. Hence, from Lemma 10, we have, for any $k \in \mathbb{N}$

$$\sup_{l \in \mathbb{N}} \bar{\alpha}_{k,l}(r) \leq C_{\text{sup}} e^{-C_{\text{inf}} r},$$

where C_{sup} and C_{inf} may depend on k and K .

We now let $D = (s_i)_{i \in \mathbb{N}}$, $D_n = (s_1, \dots, s_n)$, $Z_{i,n} = n^{-1/2} X_n(s_i)$ for $n \in \mathbb{N}$ and $i = 1, \dots, n$. We also let $c_{i,n} = n^{-1/2}$ for $n \in \mathbb{N}$ and $i = 1, \dots, n$. We remark that $Z_{i,n}/c_{i,n}$ can be written as $f(w)$ where w is a Gaussian vector of dimension less than N_K , with variances 1 and where $|f(x)| \leq C_{\text{sup}} e^{C_{\text{sup}} |x|}$, where C_{sup} does not depend on $n \in \mathbb{N}$ and $i = 1, \dots, n$. One can thus show, from the Cauchy-Schwarz inequality and with the same techniques as in Lemma 6, that for any $q > 0$

$$\lim_{M \rightarrow +\infty} \sup_{n \in \mathbb{N}} \max_{i=1, \dots, n} \mathbb{E} \left[|Z_{i,n}/c_{i,n}|^{2+q} 1_{|Z_{i,n}/c_{i,n}| \geq M} \right] = 0. \quad (6.33)$$

With the previous notation and with (6.33), one can show that all the assumptions of Corollary 1 in [167] are satisfied. This shows (6.32) and thus concludes the proof. $\square$

Proof of Theorem 3. Let $\theta \in \Theta$ be fixed. We have

$$\text{Var}(L_\theta) = \frac{1}{n} n \text{Var} \left(\frac{1}{n} (y^\top R_\theta^{-1} y) \right).$$

From Lemma 13 and Theorem 2, applied with $A_n = R_\theta^{-1}$, we obtain $\text{Var}(L_\theta) \rightarrow 0$ as $n \rightarrow \infty$. For $i = 1, \dots, p$,

$$\frac{\partial L_\theta}{\partial \theta_i} = \frac{1}{n} \text{tr} \left(R_\theta^{-1} \frac{\partial R_\theta}{\partial \theta_i} \right) + \frac{1}{n} \left(y^\top \left(-R_\theta^{-1} \frac{\partial R_\theta}{\partial \theta_i} R_\theta^{-1} \right) y \right),$$

which can be rewritten for convenience as

$$\frac{\partial L_\theta}{\partial \theta_i} = \frac{1}{n} \text{tr} (P_{\theta,i}) + \frac{1}{n} (y^\top Q_{\theta,i} y)$$

with

$$P_{\theta,i} = R_\theta^{-1} \frac{\partial R_\theta}{\partial \theta_i} \quad \text{and} \quad Q_{\theta,i} = -R_\theta^{-1} \frac{\partial R_\theta}{\partial \theta_i} R_\theta^{-1}.$$

The matrices R_θ^{-1} and $\partial R_\theta / \partial \theta_i$ are both valid choices for A_θ and B_θ in Lemma 11. From Gerschgorin Circle Theorem (GCT) and Lemma 4 in [140], we obtain $\sup_{\theta \in \Theta} \lambda_1(P_{\theta,i}^\top P_{\theta,i}) \leq C_{\text{sup}}$ and $\sup_{\theta \in \Theta} \lambda_1(Q_{\theta,i}) \leq C_{\text{sup}}$. This, in turn implies that $\sup_{\theta \in \Theta} \rho_1(P_{\theta,i}) \leq C_{\text{sup}}$. It follows that

$$\begin{aligned}
\max_{i=1,\dots,n} \sup_{\theta \in \Theta} \left| \frac{\partial L_\theta}{\partial \theta} \right| &\leq \sup_{\theta \in \Theta} \left| \frac{1}{n} \text{tr}(P_{\theta,i}) + \frac{1}{n} (y^\top Q_{\theta,i} y) \right| \\
&\leq C_{\text{sup}} + C_{\text{sup}} \frac{\|y\|^2}{n} \\
&= O_p(1).
\end{aligned}$$

Hence, Theorem 3 can be proved by proceeding as in the proof of Proposition 3.1 in [122]. $\square$

Proof of Theorem 4. From the proof of Theorem 3, we have for $i = 1, \dots, p$

$$\partial L_\theta / \partial \theta_i = \frac{1}{n} \text{tr}(P_{\theta,i}) + \frac{1}{n} (y^\top Q_{\theta,i} y),$$

where $P_{\theta,i}$ is a $n \times n$ matrix satisfying $\sup_{\theta \in \Theta} |(P_{\theta,i})_{a,b}| \leq C_{\text{sup}} / (1 + |s_a - s_b|^{d+C_{\text{inf}}})$ and $Q_{\theta,i}$ is a $n \times n$ symmetric matrix satisfying $\sup_{\theta \in \Theta} |(Q_{\theta,i})_{a,b}| \leq C_{\text{sup}} / (1 + |s_a - s_b|^{d+C_{\text{inf}}})$.

One can check that $\partial L_{\theta_0} / \partial \theta_i$ has mean zero for $i = 1, \dots, p$, since the mean value of $\partial L_{\theta_0} / \partial \theta_i$ is calculated as if Y were a Gaussian process with zero-mean and covariance function k_{Y,θ_0} . Let $\partial L_{\theta_0} / \partial \theta$ be the gradient column vector of L_θ at θ_0 . From Theorem 2, with $\mathcal{L}_{\Sigma,\theta_0,n}$ the distribution of $\sqrt{n} \partial L_{\theta_0} / \partial \theta$, as $n \rightarrow \infty$,

$$d_w(\mathcal{L}_{\Sigma,\theta_0,n}, \mathcal{N}[0, \Sigma_{\theta_0}]) \rightarrow 0. \quad (6.34)$$

In addition, for $i \in \{1, \dots, p\}$, the sequence $(n \text{Var}(\partial L_{\theta_0} / \partial \theta_i))$ is bounded, which implies that the elements of Σ_{θ_0} are bounded too.

One can check that the mean value of $\partial L_{\theta_0} / \partial \theta_i \partial \theta_j$ is $(M_{\theta_0})_{i,j}$ (also because this mean value is calculated as if Y were a Gaussian process with zero-mean and covariance function k_{Y,θ_0}).

Also, for $i, j = 1, \dots, p$, we have

$$\partial L_\theta / \partial \theta_i \partial \theta_j = \frac{1}{n} \text{tr}(C_{\theta,i,j}) + \frac{1}{n} (y^\top D_{\theta,i,j} y),$$

where $C_{\theta,i,j}$ and $D_{\theta,i,j}$ are sums of products of the matrices R_θ^{-1} , R_θ and the first and second derivative matrices of R_θ (see e.g., [122]). Hence, from Condition 4 and Lemma 13 used inside Lemma 11, we have $\sup_{\theta \in \Theta} |(C_{\theta,i,j})_{a,b}| \leq C_{\text{sup}} / (1 + |s_a - s_b|^{d+C_{\text{inf}}})$ and $\sup_{\theta \in \Theta} |(D_{\theta,i,j})_{a,b}| \leq C_{\text{sup}} / (1 + |s_a - s_b|^{d+C_{\text{inf}}})$.

Thus, the variance of $\partial L_{\theta_0} / \partial \theta_i \partial \theta_j$ goes to zero as $n \rightarrow \infty$ from Theorem 2. Hence

$$\partial L_{\theta_0} / \partial \theta_i \partial \theta_j \rightarrow^p (M_{\theta_0})_{i,j} \quad (6.35)$$

as $n \rightarrow \infty$.

It can be shown, similarly as in the proof of Proposition 3.3 in [122] that

$$\liminf_{n \rightarrow \infty} \lambda_p(M_{\theta_0}) > 0. \quad (6.36)$$

Hence, (6.7) follows.

Then, for $i, j, \ell \in \{1, \dots, p\}$, we have

$$\partial L_\theta / \partial \theta_i \partial \theta_j \partial \theta_\ell = \frac{1}{n} \text{tr}(E_{\theta,i,j,\ell}) + \frac{1}{n} (y^\top F_{\theta,i,j,\ell} y),$$

where $E_{\theta,i,j,\ell}$ and $F_{\theta,i,j,\ell}$ are sums of products of the matrices R_θ^{-1} , R_θ and the first, second and third derivative matrices of R_θ . Hence, from Condition 4 and Lemma 13 used inside Lemma 11, we have $\sup_{\theta \in \Theta} |(E_{\theta,i})_{a,b}| \leq C_{\text{sup}}/(1 + |s_a - s_b|^{d+C_{\text{inf}}})$ and $\sup_{\theta \in \Theta} (F_{\theta,i})_{a,b} \leq C_{\text{sup}}/(1 + |s_a - s_b|^{d+C_{\text{inf}}})$. Then, from GCT and Lemma 4 in [140], we have $\sup_{\theta \in \Theta} \rho_1(E_{\theta,i}) \leq C_{\text{sup}}$ and $\sup_{\theta \in \Theta} \rho_1(F_{\theta,i}) \leq C_{\text{sup}}$. Hence, as in the proof of Theorem 3, we can show

$$\sup_{\theta \in \Theta} \left| \frac{\partial L_\theta}{\partial \theta_i \partial \theta_j \partial \theta_\ell} \right| = O_p(1). \quad (6.37)$$

Also, $\lambda_1(M_{\theta_0})$ is clearly bounded as $n \rightarrow \infty$. From Theorem 2, $\lambda_1(\Sigma_{\theta_0})$ is bounded as $n \rightarrow \infty$. Hence, by considering subsequences along which M_{θ_0} and Σ_{θ_0} converge, and using (6.34), (6.35), (6.36) and (6.37), we can proceed as in the proof of Proposition D.10 in [122] and show (6.6). $\square$

Proof of Theorem 5. Let $\psi \in \mathcal{S}$ be fixed. We have

$$\text{Var}(CV_\psi) = \frac{1}{n} \text{Var} \left(\frac{1}{n} y^\top C_\psi^{-1} \text{diag}(C_\psi^{-1})^{-2} C_\psi^{-1} y \right).$$

From Lemmas 11, 13 and 14 (that can be trivially adapted by replacing θ by ψ), as well as Theorem 2, applied with $A_n = C_\psi^{-1} \text{diag}(C_\psi^{-1})^{-2} C_\psi^{-1}$, we obtain $\text{Var}(CV_\psi) \rightarrow 0$ as $n \rightarrow \infty$.

For $i = 1, \dots, p-1$,

$$\frac{\partial CV_\psi}{\partial \psi_i} = \frac{2}{n} y^\top A_{\psi,i} y$$

with

$$A_{\psi,i} = C_\psi^{-1} \text{diag}(C_\psi^{-1})^{-2} \left(\text{diag} \left(C_\psi^{-1} \frac{\partial C_\psi}{\partial \psi_i} C_\psi^{-1} \right) \text{diag}(C_\psi^{-1})^{-1} - C_\psi^{-1} \frac{\partial C_\psi}{\partial \psi_i} \right) C_\psi^{-1}.$$

As in the proof of Theorem 3, GCT and Lemma 4 in [140] lead us to $\sup_{\psi \in \mathcal{S}} \lambda_1(A_{\psi,i}^\top A_{\psi,i}) \leq C_{\text{sup}}$, which in turn implies $\sup_{\psi \in \mathcal{S}} \rho_1(A_{\psi,i}) \leq C_{\text{sup}}$. It follows that

$$\begin{aligned} \max_{i=1,\dots,p-1} \sup_{\psi \in \mathcal{S}} \left| \frac{\partial CV_\psi}{\partial \psi_i} \right| &\leq \sup_{\psi \in \mathcal{S}} \left| \frac{2}{n} (y^\top A_{\psi,i} y) \right| \\ &\leq C_{\text{sup}} \frac{\|y\|^2}{n} \\ &= O_p(1). \end{aligned}$$

Hence, Theorem 5 can also be proved by proceeding as in the proof of Proposition 3.4 in [122]. $\square$

Proof of Theorem 6. From Condition 4 and Lemma 13 used inside Lemma 11, we have for $i = 1, \dots, p-1$

$$\frac{\partial CV_\psi}{\partial \psi_i} = \frac{2}{n} y^\top A_{\psi,i} y,$$

where $A_{\psi,i}$ is a $n \times n$ matrix satisfying $\sup_{\psi \in \mathcal{S}} |(A_{\psi,i})_{a,b}| \leq C_{\text{sup}}/(1 + |s_a - s_b|^{d+C_{\text{inf}}})$. As in the proof of Theorem 4, one can check that $\partial CV_{\psi_0}/\partial \psi_i$ has mean zero for $i = 1, \dots, p-1$. Let $\partial CV_{\psi_0}/\partial \psi$ be the gradient column vector of CV_ψ at ψ_0 . From Theorem 2, with $\mathcal{L}_{\Gamma, \psi_0, n}$ the distribution of $\sqrt{n} \partial CV_{\psi_0}/\partial \psi$, as $n \rightarrow \infty$,

$$d_w(\mathcal{L}_{\Gamma, \psi_0, n}, \mathcal{N}[0, \Gamma_{\psi_0}]) \rightarrow 0. \quad (6.38)$$

In addition, for $i \in \{1, \dots, p-1\}$, the sequence $(n \text{Var}(\partial CV_{\psi_0}/\partial \psi_i))$ is bounded and thus, the elements of Γ_{ψ_0} are bounded too.

One can check that the mean value of $\partial CV_{\psi_0}/\partial \psi_i \partial \psi_j$ is $(N_{\psi_0})_{i,j}$. Furthermore, from Theorem 2, the variance of $\partial CV_{\psi_0}/\partial \psi_i \partial \psi_j$ goes to zero as $n \rightarrow \infty$. Hence, as $n \rightarrow \infty$,

$$\partial CV_{\psi_0}/\partial \psi_i \partial \psi_j \rightarrow^p (N_{\psi_0})_{i,j}. \quad (6.39)$$

It can be shown, similarly as in the proof of Proposition 3.7 in [122] that

$$\liminf_{n \rightarrow \infty} \lambda_{p-1}(N_{\psi_0}) > 0. \quad (6.40)$$

Hence, (6.10) follows. On the other hand, for $i, j = 1, \dots, p-1$, we have

$$\frac{\partial CV_{\psi}}{\partial \psi_i \partial \psi_j} = \frac{1}{n} y^\top D_{\psi,i,j} y,$$

where $D_{\psi,i,j}$ is computed as a sum of products of the matrices C_{ψ}^{-1} , C_{ψ} , the first and second derivative matrices of C_{ψ} and the diag operator (see e.g., [122]). Hence, from Condition 4 and Lemma 13 used inside Lemma 11, we have $\sup_{\psi \in \mathcal{S}} |(D_{\psi,i})_{a,b}| \leq C_{\text{sup}}/(1 + |s_a - s_b|^{d+C_{\text{inf}}})$.

Similarly, for $i, j, \ell \in \{1, \dots, p-1\}$, we have

$$\frac{\partial CV_{\psi}}{\partial \psi_i \partial \psi_j \partial \psi_{\ell}} = \frac{1}{n} y^\top E_{\psi,i,j,\ell} y,$$

where $E_{\psi,i,j,\ell}$ is a sum of products of the matrices C_{ψ}^{-1} , C_{ψ} , the first, second and third derivative matrices of C_{ψ} and the diag operator. Hence, from Condition 4 and Lemma 13 used inside Lemma 11, we have $\sup_{\psi \in \mathcal{S}} |(E_{\psi,i,j,\ell})_{a,b}| \leq C_{\text{sup}}/(1 + |s_a - s_b|^{d+C_{\text{inf}}})$. Then, from GCT and Lemma 4 in [140], we have $\sup_{\psi \in \mathcal{S}} \rho_1(E_{\psi,i,j,\ell}) \leq C_{\text{sup}}$. Hence, as in the proof of Theorem 3, we can show, for $i, j, \ell \in \{1, \dots, p-1\}$,

$$\sup_{\psi \in \mathcal{S}} \left| \frac{\partial CV_{\psi}}{\partial \psi_i \partial \psi_j \partial \psi_{\ell}} \right| = O_p(1). \quad (6.41)$$

Also, $\lambda_1(N_{\psi_0})$ is clearly bounded as $n \rightarrow \infty$. From Theorem 2, $\lambda_1(\Gamma_{\psi_0})$ is bounded as $n \rightarrow \infty$. Hence, by considering subsequences along which N_{ψ_0} and Γ_{ψ_0} converge, and using (6.38), (6.39), (6.40) and (6.41), we can proceed as in the proof of Proposition D.10 in [122] and show (6.9). $\square$

Proof of Lemma 4. We have, with $\theta = (\sigma^2, \psi) \in \Theta$ and $\theta_0 = (\sigma_0^2, \psi_0)$,

$$\begin{aligned} & \frac{1}{n} \sum_{i,j=1}^n (c_{Y,\psi}(s_i - s_j) - c_{Y,\psi_0}(s_i - s_j))^2 \\ &= \frac{1}{n} \sum_{i,j=1}^n \left(\frac{k_{Y,\theta}(s_i - s_j)}{k_{Y,\theta}(0)} - \frac{k_{Y,\theta_0}(s_i - s_j)}{k_{Y,\theta_0}(0)} \right)^2 \\ &\leq \frac{2}{n} \sum_{i,j=1}^n \left(\frac{k_{Y,\theta}(s_i - s_j)}{k_{Y,\theta}(0)} - \frac{k_{Y,\theta_0}(s_i - s_j)}{k_{Y,\theta}(0)} \right)^2 + \frac{2}{n} \sum_{i,j=1}^n \left(\frac{k_{Y,\theta_0}(s_i - s_j)}{k_{Y,\theta}(0)} - \frac{k_{Y,\theta_0}(s_i - s_j)}{k_{Y,\theta_0}(0)} \right)^2 \\ &\leq \frac{C_{\text{sup}}}{n} \sum_{i,j=1}^n (k_{Y,\theta}(s_i - s_j) - k_{Y,\theta_0}(s_i - s_j))^2 + \left(\frac{1}{k_{Y,\theta}(0)} - \frac{1}{k_{Y,\theta_0}(0)} \right)^2 C_{\text{sup}}, \end{aligned} \quad (6.42)$$

where the second C_{sup} comes from Lemma 12 and from the classical control of the square Frobenius norm by n times the largest square eigenvalue, for $n \times n$ symmetric matrices. If Condition 9 holds, then for all $\chi > 0$,

$$\liminf_{n \rightarrow \infty} \inf_{\|\psi - \psi_0\| \geq \chi} \frac{1}{n} \sum_{i,j=1}^n (c_{Y,\psi}(s_i - s_j) - c_{Y,\psi_0}(s_i - s_j))^2 > 0.$$

Consider a sequence $\theta_n = (\sigma_n^2, \psi_n) \in \Theta$ such that $\|\theta_n - \theta_0\| \geq \chi$. If we can extract a subsequence n_m such that $\liminf_{m \rightarrow \infty} (\sigma_{n_m}^2 - \sigma_0^2)^2 > 0$, then clearly

$$\liminf_{m \rightarrow \infty} \frac{1}{n_m} \sum_{i,j=1}^{n_m} (k_{Y,\theta_{n_m}}(s_i - s_j) - k_{Y,\theta_0}(s_i - s_j))^2 > 0$$

by considering the diagonal terms in the above double sum. If we can not extract such a subsequence, then we can extract a subsequence n_m such that $\|\psi_{n_m} - \psi_0\| \geq \chi/2$ and $\sigma_{n_m}^2 \rightarrow \sigma_0^2$ as $m \rightarrow \infty$. Along this subsequence

$$\liminf_{m \rightarrow \infty} \frac{1}{n_m} \sum_{i,j=1}^{n_m} (k_{Y,\theta_{n_m}}(s_i - s_j) - k_{Y,\theta_0}(s_i - s_j))^2 > 0$$

from (6.42). Hence, Condition 7 holds.

Let us now assume that Condition 8 does not hold. We have, with $\theta = (\sigma^2, \psi) \in \Theta$, with $\theta_0 = (\sigma_0^2, \psi_0)$ and with $(\beta_1, \dots, \beta_p) = (\beta_1, \chi_1, \dots, \chi_{p-1})$, where $\beta_1 \in \mathbb{R}$ is arbitrary,

$$\begin{aligned} & \frac{1}{n} \sum_{i,j=1}^n \left(\sum_{\ell=1}^{p-1} \chi_\ell \frac{\partial}{\partial \psi_\ell} c_{Y,\psi_0}(s_i - s_j) \right)^2 \\ &= \frac{1}{n} \sum_{i,j=1}^n \left(\sum_{\ell=1}^p \beta_\ell \frac{\partial}{\partial \theta_\ell} \left(\frac{k_{Y,\theta_0}(s_i - s_j)}{k_{Y,\theta_0}(0)} \right) \right)^2 \\ &= \frac{1}{n} \sum_{i,j=1}^n \left(\sum_{\ell=1}^p \beta_\ell \frac{\frac{\partial}{\partial \theta_\ell} k_{Y,\theta_0}(s_i - s_j)}{k_{Y,\theta_0}(0)} - \sum_{\ell=1}^p \beta_\ell \frac{k_{Y,\theta_0}(s_i - s_j) \frac{\partial}{\partial \theta_\ell} k_{Y,\theta_0}(0)}{k_{Y,\theta_0}(0)^2} \right)^2 \\ &\leq \frac{2}{n} \sum_{i,j=1}^n \left(\sum_{\ell=1}^p \beta_\ell \frac{\frac{\partial}{\partial \theta_\ell} k_{Y,\theta_0}(s_i - s_j)}{k_{Y,\theta_0}(0)} \right)^2 + \frac{2}{n} \sum_{i,j=1}^n \left(\sum_{\ell=1}^p \beta_\ell \frac{k_{Y,\theta_0}(s_i - s_j) \frac{\partial}{\partial \theta_\ell} k_{Y,\theta_0}(0)}{k_{Y,\theta_0}(0)^2} \right)^2 \\ &\leq \frac{C_{\text{sup}}}{n} \sum_{i,j=1}^n \left(\sum_{\ell=1}^p \beta_\ell \frac{\partial}{\partial \theta_\ell} k_{Y,\theta_0}(s_i - s_j) \right)^2 + C_{\text{sup}} \left(\sum_{\ell=1}^p \beta_\ell \frac{\partial}{\partial \theta_\ell} k_{Y,\theta_0}(0) \right)^2, \end{aligned} \tag{6.43}$$

where the second C_{sup} comes from Lemma 12 and from the classical control of the square Frobenius norm by n times the largest square eigenvalue, for $n \times n$ symmetric matrices. If Condition 8 does not hold, there exists $(\beta_1^*, \dots, \beta_p^*) \neq (0, \dots, 0)$ and a subsequence n_m such that

$$\frac{1}{n_m} \sum_{i,j=1}^{n_m} \left(\sum_{\ell=1}^p \beta_\ell^* \frac{\partial}{\partial \theta_\ell} k_{Y,\theta_0}(s_i - s_j) \right)^2 \xrightarrow{m \rightarrow \infty} 0$$

and thus, considering the diagonal elements in the double sum above,

$$\beta_1^* = \sum_{\ell=1}^p \beta_\ell^* \frac{\partial}{\partial \theta_\ell} k_{Y,\theta_0}(0) = 0.$$

Hence, from (6.43), letting $(\beta_1^*, \dots, \beta_p^*) = (0, \chi_1^*, \dots, \chi_{p-1}^*)$, we have

$$\frac{1}{n_m} \sum_{i,j=1}^{n_m} \left(\sum_{\ell=1}^{p-1} \chi_\ell^* \frac{\partial}{\partial \psi_\ell} c_{Y, \psi_0}(s_i - s_j) \right)^2 \rightarrow 0$$

and thus Condition 10 does not hold. $\square$

Proof of Theorem 7. Let λ and γ be two column vectors in $\mathbb{R}^p$ and $\mathbb{R}^{p-1}$. Let also

$$W_n = \lambda^\top (\hat{\theta}_{\text{ML}} - \theta_0) + \gamma^\top (\hat{\psi}_{\text{CV}} - \psi_0),$$

for $n \in \mathbb{N}$.

From the proofs of Theorems 4 and 6 (see also the proof of Proposition D.10 in [122] that is referred to there), we know

$$\sqrt{n}(\hat{\theta}_{\text{ML}} - \theta_0) = \sqrt{n} M_{\theta_0}^{-1} \frac{\partial}{\partial \theta} L_{\theta_0} + o_p(1)$$

and

$$\sqrt{n}(\hat{\psi}_{\text{CV}} - \psi_0) = \sqrt{n} N_{\psi_0}^{-1} \frac{\partial}{\partial \psi} C V_{\psi_0} + o_p(1).$$

Also, from Condition 4 and Lemma 13 used inside Lemma 11, we have for $i = 1, \dots, p$,

$$\partial L_{\theta_0} / \partial \theta_i = \frac{1}{n} (y^\top A_{\theta_0, i} y) + c_{\theta_0},$$

where $c_{\theta_0} \in \mathbb{R}$ is deterministic and, for $i = 1, \dots, p-1$,

$$\partial C V_{\psi_0} / \partial \psi_i = \frac{1}{n} (y^\top B_{\psi_0, i} y),$$

where $A_{\theta, i}$ is a $n \times n$ symmetric matrix satisfying $\sup_{\theta \in \Theta} |(A_{\theta, i})_{a, b}| \leq C_{\text{sup}} / (1 + |s_a - s_b|^{d+C_{\text{inf}}})$ and $B_{\psi, i}$ is a $n \times n$ matrix satisfying $\sup_{\psi \in \mathcal{S}} |(B_{\psi, i})_{a, b}| \leq C_{\text{sup}} / (1 + |s_a - s_b|^{d+C_{\text{inf}}})$. As in the proofs of Theorems 4 and 6, one can check that $\partial L_{\theta_0} / \partial \theta_i$ has mean zero for $i = 1, \dots, p$ and $\partial C V_{\psi_0} / \partial \psi_i$ has mean zero for $i = 1, \dots, p-1$. Thus, we can rewrite

$$W_n = J_n - \mathbb{E}[J_n] + o_p(n^{-1/2})$$

with

$$J_n = \frac{1}{n} y^\top \left(\sum_{i=1}^p (\lambda^\top M_{\theta_0}^{-1})_i A_{\theta_0, i} + \sum_{i=1}^{p-1} (\gamma^\top N_{\psi_0}^{-1})_i B_{\psi_0, i} \right) y.$$

As the vectors λ and γ as well as the matrices $M_{\theta_0}^{-1}$ and $N_{\psi_0}^{-1}$ are fixed, the bound

$$\left| \left(\sum_{i=1}^p (\lambda^\top M_{\theta_0}^{-1})_i A_{\theta_0, i} + \sum_{i=1}^{p-1} (\gamma^\top N_{\psi_0}^{-1})_i B_{\psi_0, i} \right)_{k, l} \right| \leq \frac{C_{\text{sup}}}{1 + |s_k - s_l|^{d+C_{\text{inf}}}}$$

holds for all $k, l \in \{1, \dots, n\}$.

Let then $\mathcal{L}_{J, \theta_0, n}$ be the distribution of $n^{1/2}(J_n - \mathbb{E}[J_n])$. Then, from Theorem 2, as $n \rightarrow \infty$,

$$d_w(\mathcal{L}_{J, \theta_0, n}, \mathcal{N}[0, n \text{Var}(J_n)]) \rightarrow 0.$$

The variance can be written as

$$\begin{aligned} n \operatorname{Var}(J_n) &= \sum_{i=1}^p \sum_{j=1}^p (\lambda^\top M_{\theta_0}^{-1})_i (\lambda^\top M_{\theta_0}^{-1})_j (\Sigma_{\theta_0})_{i,j} + \sum_{i=1}^{p-1} \sum_{j=1}^{p-1} (\gamma^\top N_{\psi_0}^{-1})_i (\gamma^\top N_{\psi_0}^{-1})_j (\Gamma_{\psi_0})_{i,j} \\ &\quad + 2 \sum_{i=1}^p \sum_{j=1}^{p-1} (\lambda^\top M_{\theta_0}^{-1})_i (\gamma^\top N_{\psi_0}^{-1})_j (\Omega_{\theta_0})_{i,j} \end{aligned}$$

with

$$\Omega_{i,j} = \operatorname{Cov} \left(\sqrt{n} \frac{\partial}{\partial \theta_i} L_{\theta_0}, \sqrt{n} \frac{\partial}{\partial \psi_j} C V_{\psi_0} \right).$$

Hence, by applying product by blocks we get the matrix form expression

$$n \operatorname{Var}(J_n) = \begin{pmatrix} \lambda^\top & \gamma^\top \end{pmatrix} D_{\theta_0}^{-1} \Psi_{\theta_0} D_{\theta_0}^{-1} \begin{pmatrix} \lambda \\ \gamma \end{pmatrix}.$$

We conclude the proof by applying the Wald Theorem. $\square$

Proof of Proposition 1. The proofs of the new versions of the theorems are direct extensions of the proofs of these theorems above. In particular, Condition 11 implies (6.17) in Lemma 12. Furthermore, Theorem 2 can be shown to hold. For instance, one can write, for $s \in \mathbb{R}^d$, $Y(s) = T(Z(s)) + \check{T}(\zeta(s)) = \dot{T}(\dot{Z}(s))$, where $\dot{Z} = (Z, \zeta)$ is a bivariate zero-mean Gaussian process and where $\dot{T} : \mathbb{R}^2 \rightarrow \mathbb{R}$ satisfies $|\dot{T}(x)| \leq C_{\sup} \exp(C_{\sup} \|x\|)$ and $|\partial/\partial x_i \dot{T}(x)| \leq C_{\sup} \exp(C_{\sup} \|x\|)$ for $x \in \mathbb{R}^2$ and $i = 1, 2$. We recall that Z and ζ are independent and $\operatorname{Cov}(\zeta(u), \zeta(v)) = \kappa 1_{u=v}$, for $u, v \in \mathbb{R}^d$, with a constant $0 < \kappa < \infty$.

Then one can check that the proof of Theorem 2 can be repeated, almost identically, by simply replacing Z by $\dot{Z}$ and T by $\dot{T}$. Finally, we remark that for (non-transformed) Gaussian processes with a nugget effect, [168, 156] address the asymptotic properties of maximum likelihood. They observe that the arguments for Gaussian processes without nugget effect in [122] can be extended directly to Gaussian processes with nugget effect. $\square$

Proof of Corollary 3. When Y has constant mean ν_0 and Z has constant mean μ_0 , then we can apply Theorem 2 to the Gaussian process $\dot{Z} = Z - \mu_0$, to the transformation $\dot{T}$ defined by $\dot{T}(x) = T(x + \mu_0)$ for $x \in \mathbb{R}$ and to the transformed process $\dot{Y} = \dot{T}(\dot{Z}) - \nu_0$. This yields a central limit theorem for quadratic forms based on the centered observation vector $y - v_{\nu_0}$, which concludes the proof.

Note that when T satisfies Condition 3, then $\dot{T}$ also satisfies Condition 3. $\square$

Chapter 7

Global conclusions and perspectives

7.1 Synthesis

This thesis raised from the RISCOPE project, devoted to the study and development of metamodels for the constitution of a quick and reliable coastal flood early warning system. Since the beginning of the project, we bet for the optimization of the structural parameters as a mechanism to produce high quality metamodels regardless of the application. [Chapter 2](#) served as a proof of this concept, and resulted in our first prototype of methodology for the calibration of structural parameters. In [Chapter 3](#), we took this idea further by introducing an Ant Colony based algorithm able to efficiently search in wider spaces of structural configuration. [Chapter 2](#) also gave place for a long set of computer trials which served for the depuration and optimization of our metamodeling code scripts. In [Chapter 4](#), those scripts were assembled to the Ant Colony algorithm proposed in [Chapter 3](#), giving rise to the R package `funGp` [169]. Finally, in [Chapter 5](#) we used the model factory implemented in `funGp` for the construction of the metamodels required for multiple output variables of the hydrodynamic code studied in RISCOPE. At the time of writing this manuscript, RISCOPE still has a year of work ahead. Based on the results obtained in [Chapter 5](#), for now we can say that we are on the right track.

On the other hand, in [Chapter 6](#) we conducted a theoretical study which confirms the suitability of the Gaussian process model as regression technique for cases of non-Gaussian outputs. This conclusion covers multiple outputs of the RISCOPE code which present non-Gaussian characteristics such as nonnegativity (e.g., the water height at given points in land). This study could be extended in the future, in order to incorporate the possibility of transforming the output data with the aim of improving learning and inference in the case of non-Gaussian outputs.

7.2 Scientific production

Published papers

- **Gaussian process metamodeling of functional-input code for coastal flood hazard assessment**, *Reliability Engineering & System Safety*, 2020.
In bibliography: [7] Online: [find it here](#) ↗

- **Asymptotic properties of the maximum likelihood and cross validation estimators for transformed Gaussian processes**, *Electronic Journal of Statistics*.
In bibliography: [23] Online: [find it here](#) ↗

Accepted paper

- **Toward a user-based, robust and fast running method for coastal flooding forecast, early warning, and risk prevention**, *Journal of coastal research, proceedings from the International Coastal Symposium (ICS) 2020*.
In bibliography: [104]

Software

- **R package funGp**, Available on [CRAN](#) ↗ and [GitHub](#) ↗ .
In bibliography: [20]
 - * **User manual: Gaussian process regression for scalar and functional inputs with funGp - The in-depth tour**.
In bibliography: [21] Online: [find it here](#) ↗
-

7.3 Research perspectives

For near future development, we propose the following research avenues.

A. On the algorithm for structural parameter optimization

The Ant Colony algorithm presented in [Chapter 3](#) and implemented in the R package funGp, was designed general enough to be usable in a variety of regression contexts. Hence, it would be interesting to use the methodology for addressing some of the following scenarios.

Functional inputs in larger dimensions: in the RISCOPE application, as well as all the other analytic cases revised in this thesis, the functional inputs are time series (functions in dimension one). However, there is nothing at first sight that would prevent our Ant Colony algorithm from properly managing functional inputs in larger dimension, as could be the case of fields or images (functions in dimension two). To do so, tensorized finite dimensional projection spaces could be considered (see e.g., [64] or [24]). For some functional input f from $T \subset \mathbb{R}^d$ to $\mathbb{R}$, the tensorized projection space has a dimension of the form $p^{(1)} \times \dots \times p^{(d)}$. Each projection dimension $p^{(1)}, \dots, p^{(d)}$ could be defined as a structural parameter of the model. Then, the Ant Colony algorithm would work the same as for projections of functions in dimension one (e.g., time series). If it performs well, this approach would prove the suitability of our algorithm for a more general class of functional inputs.

Functional output: the output of the RISCOPE case was scalar. Hence, we focused our experiments in scalar-output black-boxes. However, most functional-output regression problems can be reduced to an aggregation of multiple scalar-output ones. Thus we think our algorithm would still be usable in those cases.

For instance, if the original output is a time series, an individual metamodel could be used to predict the output at each time instant. In that case, we could use our Ant Colony algorithm for tuning the structural parameters of each required metamodel. Another approach to deal with such an output could be to use the time index as an input, making the output scalar (see e.g., [65]). If we do so, we would require the new artificial input variable to remain always active in the model in order to be able to properly recover the output shape following prediction. This could be achieved by specifying a constraint to the algorithm, a possibility already implemented in the funGp model factory.

This research line would then consist in assessing the ability of our algorithm to find high quality metamodels in the case of functional-output codes modeled by one of the aforementioned approaches.

More factors and levels: in this thesis, we used our Ant Colony algorithm for calibrating five structural parameters: (i) the state of each input in the model, (ii) the dimension reduction method to use for each input, (iii) the projection dimension for each input, (iv) the kernel function of the model, and (v) the distance to measure similarities between functional input coordinates. We concentrated on those parameters since we found them to be the most relevant ones for our particular application. However, other applications might merit the analysis of other structural parameters as can be the mean structure (e.g., null, constant, polynomial) or the type of transformation of the output (e.g., different Box-Cox functions [170]), if relevant.

B. On the treatment of strongly skewed output

Most of the outputs in the RISCOPE application show strongly skewed distributions when observations are taken directly from the nature. This happens because most part of the time, the application site (Gâvres, France) is not flooded. Thus, most of the output values indicating degree of flooding fail close to zero, and only a few ones get far away from zero. This behavior is typical of the vast majority of populated regions in the planet. Unfortunately, this natural bias impacts the efficiency of metamodel training, since the majority of learning data will match mild events (leading to minor or no flooding) and the metamodel will not learn well how to predict strong events (leading to major flooding). Below, we describe three research avenues that could be followed in order to improve this situation.

Adaptive DoE for uniform coverage of the output range: this first line consists of the use of sequential design techniques (e.g., [84, 83]) oriented to dynamically add events to the learning set while keeping the bias of the output as controlled as possible. In particular, it would be interesting to find a way to account for characteristic features of the functional inputs such as the parabolic shape of the tidal curve. In addition, it would be desirable to be able to model the dependency between time steps of the same input variable and also of different input variables.

Transformation of the output: another action with a strong positive potential is the study of transform functions that could be applied to the output data in order to reduce

the bias (e.g., [170, 171]). The simplest approach would be to set the parameters of the transformation beforehand and then conduct the optimization of the hyperparameters of the metamodel. This approach could already be beneficial, but a superior method would be to optimize the parameters of the transformation along with those of the metamodel. This way, the transformation will be optimized either for likelihood or metamodel predictability. Once the optimization of a single transform function has been dominated, one could go one step further and include the type of transformation as one of the structural parameters to calibrate with our Ant Colony algorithm.

Theoretical study of output transformation in Gaussian processes: this research line is a direct extension of the theoretical work presented in [Chapter 6](#) and [23]. In that work we proved that the covariance parameters of a non-Gaussian process can be well estimated when we model it as Gaussian. The extension would consist in assessing the case where we perform a parametric transformation of the output and we aim at estimating jointly the parameters of the transformation and the covariance parameters. This theoretical study would help us to get some guarantees about the benefit in learning and inference derived from such an approach.

Bibliography

- [1] R. E. Kirk, “Experimental design,” *Handbook of Psychology, Second Edition*, vol. 2, 2012.
- [2] X. Wu, *Metamodel-based inverse uncertainty quantification of nuclear reactor simulators under the Bayesian framework*. PhD thesis, University of Illinois at Urbana-Champaign, 2017.
- [3] K. R. Vaden and R. N. Simons, “Computer aided design of ka-band waveguide power combining architectures for interplanetary spacecraft,” in *2005 IEEE Antennas and Propagation Society International Symposium*, vol. 1, pp. 635–638, IEEE, 2005.
- [4] J. Rohmer and E. Foerster, “Global sensitivity analysis of large-scale numerical landslide models based on gaussian-process meta-modeling,” *Computers & geosciences*, vol. 37, no. 7, pp. 917–927, 2011.
- [5] J. Villemonteix, E. Vazquez, and E. Walter, “An informational approach to the global optimization of expensive-to-evaluate functions,” *Journal of Global Optimization*, vol. 44, no. 4, p. 509, 2009.
- [6] A. Forrester, A. Sobester, and A. Keane, *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.
- [7] J. Betancourt, F. Bachoc, T. Klein, D. Idier, R. Pedreros, and J. Rohmer, “Gaussian process metamodeling of functional-input code for coastal flood hazard assessment,” *Reliability Engineering & System Safety*, vol. 198, p. 106870, 2020.
- [8] H. G. Pulido, R. De la Vara Salazar, P. G. González, C. T. Martínez, and M. d. C. T. Pérez, *Análisis y diseño de experimentos*. New York: McGraw-Hill, 2012.
- [9] M. Dorigo, V. Maniezzo, and A. Colorni, “Positive feedback as a search strategy,” 1991.
- [10] M. Dorigo, “Optimization, learning and natural algorithms,” *PhD Thesis, Politecnico di Milano*, 1992.
- [11] C. Blum, M. Y. Vallès, and M. J. Blesa, “An ant colony optimization algorithm for dna sequencing by hybridization,” *Computers & Operations Research*, vol. 35, no. 11, pp. 3620–3635, 2008.
- [12] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, “Cloud task scheduling based on load balancing ant colony optimization,” in *2011 sixth annual ChinaGrid conference*, pp. 3–9, IEEE, 2011.

- [13] O. Korb, T. Stützle, and T. E. Exner, “An ant colony optimization approach to flexible protein–ligand docking,” *Swarm Intelligence*, vol. 1, no. 2, pp. 115–134, 2007.
- [14] A. S. Simaria and P. M. Vilarinho, “2-antbal: An ant colony optimisation algorithm for balancing two-sided assembly lines,” *Computers & Industrial Engineering*, vol. 56, no. 2, pp. 489–506, 2009.
- [15] D. Zhao, L. Luo, and K. Zhang, “An improved ant colony optimization for communication network routing problem,” in *2009 Fourth International on Conference on Bio-Inspired Computing*, pp. 1–4, IEEE, 2009.
- [16] E. Bonabeau, M. Dorigo, D. d. R. D. F. Marco, G. Theraulaz, G. Théraulaz, *et al.*, *Swarm intelligence: from natural to artificial systems*. No. 1, Oxford university press, 1999.
- [17] E. Bonabeau, M. Dorigo, and G. Theraulaz, “Inspiration for optimization from social insect behaviour,” *Nature*, vol. 406, no. 6791, pp. 39–42, 2000.
- [18] C. Blum, “Ant colony optimization: Introduction and recent trends,” *Physics of Life reviews*, vol. 2, no. 4, pp. 353–373, 2005.
- [19] J. Betancourt, F. Bachoc, T. Klein, and Gamboa, “Technical report: Ant colony based model selection for functional-input gaussian process regression. Ref. D3.b (WP3.2), *RISCOPE project*.” <https://hal.archives-ouvertes.fr/hal-02532713>, Apr. 2020. hal-02532713.
- [20] J. Betancourt, F. Bachoc, and T. Klein, “funp: Gaussian process models for scalar and functional inputs.” <https://CRAN.R-project.org/package=funGp>, Apr. 2020. R package version 0.1.0.
- [21] J. Betancourt, F. Bachoc, and T. Klein, “Gaussian process regression for scalar and functional inputs with funp - the in-depth tour. *RISCOPE project*.” <https://hal.archives-ouvertes.fr/hal-02536624>, Apr. 2020. hal-02536624.
- [22] C. Lataniotis, S. Marelli, and B. Sudret, “Extending classical surrogate modelling to ultrahigh dimensional problems through supervised dimensionality reduction: a data-driven approach,” Oct. 2019.
- [23] F. Bachoc, J. Betancourt, R. Furrer, T. Klein, *et al.*, “Asymptotic properties of the maximum likelihood and cross validation estimators for transformed gaussian processes,” *Electronic Journal of Statistics*, vol. 14, no. 1, pp. 1962–2008, 2020.
- [24] A. F. López-Lopera, F. Bachoc, N. Durrande, and O. Roustant, “Finite-dimensional gaussian approximation with linear inequality constraints,” *SIAM/ASA Journal on Uncertainty Quantification*, vol. 6, no. 3, pp. 1224–1255, 2018.
- [25] S. Golchi, D. R. Bingham, H. Chipman, and D. A. Campbell, “Monotone emulation of computer experiments,” *SIAM/ASA Journal on Uncertainty Quantification*, vol. 3, no. 1, pp. 370–392, 2015.
- [26] J. Rohmer and D. Idier, “A meta-modelling strategy to identify the critical offshore conditions for coastal flooding,” *Natural Hazards and Earth System Sciences*, vol. 12, no. 9, pp. 2943–2955, 2012.

- [27] G. Jia and A. A. Taflanidis, “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*, vol. 261, pp. 24–38, 2013.
- [28] A. Rueda, B. Gouldby, F. Méndez, A. Tomás, I. Losada, J. Lara, and P. Díaz-Simal, “The use of wave propagation and reduced complexity inundation models and meta-models for coastal flood risk assessment,” *Journal of Flood Risk Management*, vol. 9, no. 4, pp. 390–401, 2016.
- [29] T. Muehlenstaedt, J. Fruth, and O. Roustant, “Computer experiments with functional inputs and scalar outputs by a norm-based approach,” *Statistics and Computing*, vol. 27, no. 4, pp. 1083–1097, 2017.
- [30] S. Nanty, C. Helbert, A. Marrel, N. Pérot, and C. Prieur, “Sampling, metamodeling, and sensitivity analysis of numerical simulators with functional stochastic inputs,” *SIAM/ASA Journal on Uncertainty Quantification*, vol. 4, no. 1, pp. 636–659, 2016.
- [31] T. J. Santner, B. J. Williams, W. Notz, and B. J. Williams, *The design and analysis of computer experiments*, vol. 1. Springer, 2003.
- [32] C. Lataniotis, S. Marelli, and B. Sudret, “The gaussian process modelling module in uqlab,” *Soft Computing in Civil Engineering*, vol. 2, no. 3, pp. 91–116, 2018.
- [33] J. O. Ramsay and B. W. Silverman, *Applied functional data analysis: methods and case studies*. Springer, 2007.
- [34] A. Marrel, B. Iooss, M. Jullien, B. Laurent, and E. Volkova, “Global sensitivity analysis for models with spatially dependent outputs,” *Environmetrics*, vol. 22, no. 3, pp. 383–397, 2011.
- [35] C. V. Mai and B. Sudret, “Surrogate models for oscillatory systems using sparse polynomial chaos expansions and stochastic time warping,” *SIAM/ASA Journal on Uncertainty Quantification*, vol. 5, no. 1, pp. 540–571, 2017.
- [36] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, ACM, 2008.
- [37] P. T. Reiss, J. Goldsmith, H. L. Shang, and R. T. Ogden, “Methods for scalar-on-function regression,” *International Statistical Review*, vol. 85, no. 2, pp. 228–249, 2017.
- [38] A. Antoniadis, C. Helbert, C. Prieur, and L. Viry, “Spatio-temporal metamodeling for west african monsoon,” *Environmetrics*, vol. 23, no. 1, pp. 24–36, 2012.
- [39] J. Rohmer, “Boosting kernel-based dimension reduction for jointly propagating spatial variability and parameter uncertainty in long-running flow simulators,” *Mathematical Geosciences*, vol. 47, no. 2, pp. 227–246, 2015.
- [40] P. G. Constantine, *Active subspaces: Emerging ideas for dimension reduction in parameter studies*, vol. 2. SIAM, 2015.
- [41] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *science*, vol. 313, no. 5786, pp. 504–507, 2006.

- [42] A. Damianou and N. Lawrence, “Deep gaussian processes,” in *Artificial Intelligence and Statistics*, pp. 207–215, 2013.
- [43] W. Huang, D. Zhao, F. Sun, H. Liu, and E. Chang, “Scalable gaussian process regression using deep neural networks,” in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, pp. 3576 – 3582, 2015.
- [44] R. Calandra, J. Peters, C. E. Rasmussen, and M. P. Deisenroth, “Manifold gaussian processes for regression,” in *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 3338–3345, IEEE, 2016.
- [45] M. Fornasier, K. Schnass, and J. Vybiral, “Learning functions of few arbitrary linear parameters in high dimensions,” *Foundations of Computational Mathematics*, vol. 12, no. 2, pp. 229–262, 2012.
- [46] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, “Design and analysis of computer experiments,” *Statistical science*, vol. 4, pp. 409–423, 1989.
- [47] J. Oakley and A. O’Hagan, “Bayesian inference for the uncertainty distribution of computer model outputs,” *Biometrika*, vol. 89, no. 4, pp. 769–784, 2002.
- [48] C. E. Rasmussen, “Gaussian processes in machine learning,” in *Summer School on Machine Learning*, pp. 63–71, Springer, 2003.
- [49] M. L. Stein, *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 2012.
- [50] “Anr riscope project.” <https://perso.math.univ-toulouse.fr/riscope/>, 2020. Accessed: 2018-12-04.
- [51] N. Booij, L. Holthuijsen, and R. Ris, “The "swan" wave model for shallow water,” in *Coastal Engineering 1996*, pp. 668–676, 1997.
- [52] J. Van der Meer, N. Allsop, T. Bruce, J. De Rouck, A. Kortenhaus, T. Pullen, H. Schüttrumpf, P. Troch, and B. Zanuttigh, “Eurotop: Manual on wave overtopping of sea defences and related structures: an overtopping manual largely based on european research, but for worldwide application,” 2016.
- [53] D. Idier, J. Rohmer, R. Pedreros, S. Le Roy, J. Lambert, J. Louisor, G. Le Cozannet, and E. Le Cornec, “Coastal flood: a composite method for past events characterisation providing insights in past, present and future hazards,” in *AGU Fall Meeting 2019*, AGU, 2019.
- [54] M. Moustapha, B. Sudret, J.-M. Bourinet, and B. Guillaume, “Quantile-based optimization under uncertainties using adaptive kriging surrogate models,” *Structural and multidisciplinary optimization*, vol. 54, no. 6, pp. 1403–1421, 2016.
- [55] P. Abrahamsen, “A review of gaussian random fields and correlation functions,” 1997.
- [56] D. Ginsbourger, B. Rossopoff, G. Pirot, N. Durrande, and P. Renard, “Distance-based kriging relying on proxy simulations for inverse conditioning,” *Advances in water resources*, vol. 52, pp. 275–291, 2013.

- [57] D. C. Montgomery, *Design and analysis of experiments*. John Wiley & sons, 2017.
- [58] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, vol. 1. New York: Springer series in statistics, 2001.
- [59] B. D. Ripley, *Spatial statistics*, vol. 575. John Wiley & Sons, 2005.
- [60] F. Bachoc, “Cross validation and maximum likelihood estimations of hyper-parameters of gaussian processes with model misspecification,” *Computational Statistics & Data Analysis*, vol. 66, pp. 55–69, 2013.
- [61] J. Green, J. L. Whalley, and C. G. Johnson, “Automatic programming with ant colony optimization,” in *Proceedings of the 2004 UK Workshop on Computational Intelligence*, pp. 70–77, Loughborough University, 2004.
- [62] D. Karaboga, C. Ozturk, N. Karaboga, and B. Gorkemli, “Artificial bee colony programming for symbolic regression,” *Information Sciences*, vol. 209, pp. 1–15, 2012.
- [63] N. Q. Uy, N. X. Hoai, M. O’Neill, R. I. McKay, and E. Galván-López, “Semantically-based crossover in genetic programming: application to real-valued symbolic regression,” *Genetic Programming and Evolvable Machines*, vol. 12, no. 2, pp. 91–119, 2011.
- [64] H. Maatouk and X. Bay, “A new rejection sampling method for truncated multivariate gaussian random variables restricted to convex sets,” in *Monte carlo and quasi-monte carlo methods*, pp. 521–530, Springer, 2016.
- [65] J. Rougier, “Efficient emulators for multivariate deterministic functions,” *Journal of Computational and Graphical Statistics*, vol. 17, no. 4, pp. 827–843, 2008.
- [66] A. Marrel, B. Iooss, S. Da Veiga, and M. Ribatet, “Global sensitivity analysis of stochastic computer models with joint metamodels,” *Statistics and Computing*, vol. 22, no. 3, pp. 833–847, 2012.
- [67] V. Picheny, D. Ginsbourger, Y. Richet, and G. Caplin, “Quantile-based optimization of noisy computer experiments with tunable precision,” *Technometrics*, vol. 55, no. 1, pp. 2–13, 2013.
- [68] V. Moutoussamy, S. Nanty, and B. Pauwels, “Emulators for stochastic simulation codes,” *ESAIM: Proceedings and Surveys*, vol. 48, pp. 116–155, 2015.
- [69] T. Browne, B. Iooss, L. L. Gratiet, J. Lonchampt, and E. Remy, “Stochastic simulators based optimization by gaussian process metamodels – application to maintenance investments planning issues,” *Quality and Reliability Engineering International*, vol. 32, no. 6, pp. 2067–2080, 2016.
- [70] M. D. McKay, R. J. Beckman, and W. J. Conover, “Comparison of three methods for selecting values of input variables in the analysis of output from a computer code,” *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.
- [71] C. De Boor, “A practical guide to splines,” in *Applied mathematical sciences*, vol. 27, pp. 15–16, Heidelberg: Springer, 1978.

- [72] S. Nanty, C. Helbert, A. Marrel, N. Pérot, and C. Prieur, “Uncertainty quantification for functional dependent random variables,” *Computational Statistics*, vol. 32, no. 2, pp. 559–583, 2017.
- [73] I. Jolliffe, *Principal component analysis*. Springer, 2011.
- [74] I. Papaioannou, M. Ehre, and D. Straub, “Pls-based adaptation for efficient pce representation in high dimensions,” *Journal of Computational Physics*, vol. 387, pp. 186–204, 2019.
- [75] A. Marrel, B. Iooss, F. Van Dorpe, and E. Volkova, “An efficient methodology for modeling complex computer codes with gaussian processes,” *Computational Statistics & Data Analysis*, vol. 52, no. 10, pp. 4731–4744, 2008.
- [76] O. Roustant, D. Ginsbourger, and Y. Deville, “Dicekriging, diceoptim: Two r packages for the analysis of computer experiments by kriging-based metamodelling and optimization,” *Journal of Statistical Software*, vol. 51, no. 1, p. 54p, 2012.
- [77] J. Nilsson, S. de Jong, and A. K. Smilde, “Multiway calibration in 3d qsar,” *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 11, no. 6, pp. 511–524, 1997.
- [78] Z. Shen, K.-C. Toh, and S. Yun, “An accelerated proximal gradient algorithm for frame-based image restoration via the balanced approach,” *SIAM Journal on Imaging Sciences*, vol. 4, no. 2, pp. 573–596, 2011.
- [79] E. Taillard, “Some efficient heuristic methods for the flow shop sequencing problem,” *European journal of Operational research*, vol. 47, no. 1, pp. 65–74, 1990.
- [80] S. Marque-Pucheu, *Gaussian process regression of two nested computer codes*. PhD thesis, Université Paris-Diderot-Paris VII, 2018.
- [81] S. Marque-Pucheu, G. Perrin, and J. Garnier, “Efficient sequential experimental design for surrogate modeling of nested codes,” *ESAIM: Probability and Statistics*, vol. 23, pp. 245–270, 2019.
- [82] B. Dixit, *Elasticsearch Essentials*. Packt Publishing Ltd, 2016.
- [83] W. Hendrickx, D. Gorissen, and T. Dhaene, “Grid enabled sequential design and adaptive metamodeling,” in *Proceedings of the 2006 Winter Simulation Conference*, pp. 872–881, IEEE, 2006.
- [84] W. Hendrickx and T. Dhaene, “Sequential design and rational metamodelling,” in *Proceedings of the 2005 Winter Simulation Conference.*, pp. 290 – 298, IEEE, 2005.
- [85] L. Davis, *Handbook of genetic algorithms*. CUMINCAD, 1991.
- [86] M. Dorigo and M. Birattari, *Ant colony optimization*. Springer, 2010.
- [87] U. Mori, A. Mendiburu, and J. A. Lozano, “Distance measures for time series in r: The tsdist package,” *R journal*, vol. 8, no. 2, pp. 451–459, 2016.

- [88] J. Bigot, R. Gouet, T. Klein, A. López, *et al.*, “Geodesic pca in the wasserstein space by convex pca,” in *Annales de l’Institut Henri Poincaré, Probabilités et Statistiques*, vol. 53, pp. 1–26, Institut Henri Poincaré, 2017.
- [89] L. Carrere, F. Lyard, M. Cancet, A. Guillot, N. Picot, *et al.*, “Fes 2014, a new tidal model—validation results and perspectives for improvements,” in *Proceedings of the ESA living planet symposium*, pp. 9–13, 2016.
- [90] G. Compo, J. Whitaker, P. Sardeshmukh, N. Matsui, R. Allan, X. Yin, B. Gleason, R. Vose, G. Rutledge, P. Bessemoulin, *et al.*, “Noaa/cires twentieth century global reanalysis version 2c.” <https://doi.org/10.5065/D6N877TW>, 2015, updated yearly. Accessed: 28 feb 2017.
- [91] D. Dee, M. Balmaseda, G. Balsamo, R. Engelen, A. Simmons, and J.-N. Thépaut, “Toward a consistent reanalysis of the climate system,” *Bulletin of the American Meteorological Society*, vol. 95, no. 8, pp. 1235–1248, 2014.
- [92] H. Muller, L. Pineau-Guillou, D. Idier, and F. Ardhuin, “Atmospheric storm surge modeling methodology along the french (atlantic and english channel) coast,” *Ocean Dynamics*, vol. 64, no. 11, pp. 1671–1692, 2014.
- [93] X. Bertin, E. Prouteau, and C. Letetrel, “A significant increase in wave height in the north atlantic ocean over the 20th century,” *Global and Planetary Change*, vol. 106, pp. 77–83, 2013.
- [94] E. Charles, D. Idier, J. Thiébot, G. Le Cozannet, R. Pedreros, F. Ardhuin, and S. Plan-ton, “Wave climate variability and trends in the bay of biscay from 1958 to 2001,” *Journal of Climate*, vol. 25, pp. 2020–2039, 2012.
- [95] E. Boudière, C. Maisondieu, F. Ardhuin, M. Accensi, L. Pineau-Guillou, and J. Lepesqueur, “A suitable metocean hindcast database for the design of marine energy converters,” *International Journal of Marine Energy*, vol. 3, pp. 40–52, 2013.
- [96] D. P. Bertsekas, “Nonlinear programming,” *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, 1997.
- [97] R. A. Beezer, T. Hastie, R. Tibshirani, and J. F. Springer, “The elements of statistical learning: Data mining, inference and prediction. by,” 2002.
- [98] K.-T. Fang, R. Li, and A. Sudjianto, *Design and modeling for computer experiments*. Chapman and Hall/CRC, 2005.
- [99] S. Goss, S. Aron, J.-L. Deneubourg, and J. M. Pasteels, “Self-organized shortcuts in the argentine ant,” *Naturwissenschaften*, vol. 76, no. 12, pp. 579–581, 1989.
- [100] M. Dorigo and L. M. Gambardella, “Ant colony system: a cooperative learning approach to the traveling salesman problem,” *IEEE Transactions on evolutionary computation*, vol. 1, no. 1, pp. 53–66, 1997.
- [101] A. K. Das, R. J. Marks, M. El-Sharkawi, P. Arabshahi, and A. Gray, “The minimum power broadcast problem in wireless networks: an ant colony system approach,” in *proceedings of the IEEE Workshop on Wireless Communications and Networking*, 2002.

- [102] M. Garmsiri and M. R. Abassi, “Resource leveling scheduling by an ant colony-based model,” *Journal of Industrial Engineering International*, vol. 8, no. 1, p. 7, 2012.
- [103] A. Marrel, B. Iooss, B. Laurent, and O. Roustant, “Calculations of sobol indices for the gaussian process metamodel,” *Reliability Engineering & System Safety*, vol. 94, no. 3, pp. 742–751, 2009.
- [104] D. Idier, A. Aurouet, F. Bachoc, A. Baills, J. Betancourt, J. Durand, R. Mouche, J. Rohmer, F. Gamboa, K. T, J. Lambert, G. Le Cozannet, S. Le Roy, J. Louisor, R. Pedreros, and A. Véron, “Toward a user-based, robust and fast running method for coastal flooding forecast, early warning, and risk prevention,” *Journal of coastal research*, vol. 95, pp. 11–15, 2020.
- [105] O. Dubrule, “Cross validation of kriging in a unique neighborhood,” *Journal of the International Association for Mathematical Geology*, vol. 15, no. 6, pp. 687–699, 1983.
- [106] A. Cohen, I. Daubechies, and J.-C. Feauveau, “Biorthogonal bases of compactly supported wavelets,” *Communications on pure and applied mathematics*, vol. 45, no. 5, pp. 485–560, 1992.
- [107] B. Schölkopf, A. Smola, and K.-R. Müller, “Nonlinear component analysis as a kernel eigenvalue problem,” *Neural computation*, vol. 10, no. 5, pp. 1299–1319, 1998.
- [108] H. Bengtsson, *doFuture: A Universal Foreach Parallel Adapter using the Future API of the ‘future’ Package*, 2020. R package version 0.9.0.
- [109] H. Bengtsson, *future: Unified Parallel and Distributed Processing in R for Everyone*, 2020. R package version 1.16.0.
- [110] R. L. Smith, “Extreme value analysis of environmental time series: an application to trend detection in ground-level ozone,” *Statistical Science*, pp. 367–377, 1989.
- [111] M. Stein, *Interpolation of Spatial Data: Some Theory for Kriging*. Springer-Verlag, New York, 1999.
- [112] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, 2006.
- [113] G. Matheron, *La Théorie des Variables Régionalisées et ses Applications*. Fasicule 5 in Les Cahiers du Centre de Morphologie Mathématique de Fontainebleau, Ecole Nationale Supérieure des Mines de Paris, 1970.
- [114] J. Sacks, W. Welch, T. Mitchell, and H. Wynn, “Design and analysis of computer experiments,” *Statistical Science*, vol. 4, pp. 409–423, 1989.
- [115] T. Santner, B. Williams, and W. Notz, *The Design and Analysis of Computer Experiments*. Springer, New York, 2003.
- [116] F. Bachoc, K. Ammar, and J. Martinez, “Improvement of code behavior in a design of experiments by metamodeling,” *Nuclear science and engineering*, vol. 183, no. 3, pp. 387–406, 2016.

- [117] R. Paulo, G. Garcia-Donato, and J. Palomo, “Calibration of computer models with multivariate output,” *Computational Statistics and Data Analysis*, vol. 56, pp. 3959–3974, 2012.
- [118] F. Bachoc, G. Bois, J. Garnier, and J. Martinez, “Calibration and improved prediction of computer models by universal Kriging,” *Nuclear Science and Engineering*, vol. 176, no. 1, pp. 81–97, 2014.
- [119] D. Jones, M. Schonlau, and W. Welch, “Efficient global optimization of expensive black box functions,” *Journal of Global Optimization*, vol. 13, pp. 455–492, 1998.
- [120] P. Abrahamsen, “A review of Gaussian random fields and correlation functions,” tech. rep., Norwegian Computing Center, 1997.
- [121] H. Zhang and Y. Wang, “Kriging and cross validation for massive spatial data,” *Environmetrics*, vol. 21, pp. 290–304, 2010.
- [122] F. Bachoc, “Asymptotic analysis of the role of spatial sampling for covariance parameter estimation of Gaussian processes,” *Journal of Multivariate Analysis*, vol. 125, pp. 1–35, 2014.
- [123] F. Bachoc, A. Lagnoux, and T. M. N. Nguyen, “Cross-validation estimation of covariance parameters under fixed-domain asymptotics,” *Journal of Multivariate Analysis*, vol. 160, pp. 42–67, 2017.
- [124] J. Istas and G. Lang, “Quadratic variations and estimation of the local Hölder index of a Gaussian process,” *Annales de l’Institut Henri Poincaré*, vol. 33, pp. 407–436, 1997.
- [125] E. Anderes, “On the consistent separation of scale and variance for Gaussian random fields,” *The Annals of Statistics*, vol. 38, pp. 870–893, 2010.
- [126] J.-M. Azaïs, F. Bachoc, T. Klein, A. Lagnoux, and T. M. N. Nguyen, “Semi-parametric estimation of the variogram of a Gaussian process with stationary increments,” *arXiv:1806.03135*, 2018.
- [127] N. Cressie, *Statistics for Spatial Data*. John Wiley & Sons, 1993.
- [128] I. Ibragimov and Y. Rozanov, *Gaussian Random Processes*. Springer-Verlag, New York, 1978.
- [129] M. Stein, “Asymptotically efficient prediction of a random field with a misspecified covariance function,” *Annals of Statistics*, vol. 16, pp. 55–63, 1988.
- [130] M. Stein, “Bounds on the efficiency of linear predictions using an incorrect covariance function,” *Annals of Statistics*, vol. 18, pp. 1116–1138, 1990.
- [131] M. Stein, “Uniform asymptotic optimality of linear predictions of a random field using an incorrect second-order structure,” *Annals of Statistics*, vol. 18, pp. 850–872, 1990.
- [132] H. Zhang, “Inconsistent estimation and asymptotically equivalent interpolations in model-based geostatistics,” *Journal of the American Statistical Association*, vol. 99, pp. 250–261, 2004.

- [133] C. Kaufman and B. Shaby, “The role of the range parameter for estimation and prediction in geostatistics,” *Biometrika*, vol. 100, pp. 473–484, 2013.
- [134] J. Du, H. Zhang, and V. Mandrekar, “Fixed-domain asymptotic properties of tapered maximum likelihood estimators,” *The Annals of Statistics*, vol. 37, pp. 3330–3361, 2009.
- [135] D. Wang and W.-L. Loh, “On fixed-domain asymptotics and covariance tapering in Gaussian random field models,” *Electronic Journal of Statistics*, vol. 5, pp. 238–269, 2011.
- [136] Z. Ying, “Asymptotic properties of a maximum likelihood estimator with data from a Gaussian process,” *Journal of Multivariate Analysis*, vol. 36, pp. 280–296, 1991.
- [137] Z. Ying, “Maximum likelihood estimation of parameters under a spatial sampling scheme,” *Annals of Statistics*, vol. 21, pp. 1567–1590, 1993.
- [138] K. Mardia and R. Marshall, “Maximum likelihood estimation of models for residual covariance in spatial regression,” *Biometrika*, vol. 71, pp. 135–146, 1984.
- [139] B. A. Shaby and D. Ruppert, “Tapered covariance: Bayesian estimation and asymptotics,” *Journal of Computational and Graphical Statistics*, vol. 21, no. 2, pp. 433–452, 2012.
- [140] R. Furrer, F. Bachoc, and J. Du, “Asymptotic properties of multivariate tapering for estimation and prediction,” *Journal of Multivariate Analysis*, vol. 149, pp. 177–191, 2016.
- [141] J.-P. Chiles and P. Delfiner, *Geostatistics: Modeling Spatial Uncertainty*. John Wiley & Sons, 2009.
- [142] G. Xu and M. G. Genton, “Tukey g-and-h random fields,” *Journal of the American Statistical Association*, vol. 112, no. 519, pp. 1236–1249, 2017.
- [143] A. Alegría, S. Caro, M. Bevilacqua, E. Porcu, and J. Clarke, “Estimating covariance functions of multivariate skew-Gaussian random fields on the sphere,” *Spatial Statistics*, vol. 22, pp. 388–402, 2017.
- [144] Y. Yan and M. G. Genton, “Gaussian likelihood inference on data from trans-Gaussian random fields with Matérn covariance function,” *Environmetrics*, vol. 29, no. 5-6, p. e2458, 2018.
- [145] F. Bachoc and R. Furrer, “On the smallest eigenvalues of covariance matrices of multivariate spatial processes,” *Stat*, vol. 5, no. 1, pp. 102–107, 2016.
- [146] F. Bachoc, “Asymptotic analysis of covariance parameter estimation for Gaussian processes in the misspecified case,” *Bernoulli*, vol. 24, no. 2, pp. 1531–1575, 2018.
- [147] M. Bevilacqua, T. Faouzi, R. Furrer, and E. Porcu, “Estimation and prediction using generalized wendland covariance functions under fixed domain asymptotics,” *The Annals of Statistics*, vol. 47, no. 2, pp. 828–856, 2019.
- [148] T. Gneiting and M. Schlather, “Stochastic models that separate fractal dimension and the hurst effect,” *SIAM review*, vol. 46, no. 2, pp. 269–282, 2004.

- [149] R. M. Dudley, *Real Analysis and Probability*. Cambridge University Press, 2002.
- [150] F. Bachoc, D. Preinerstorfer, and L. Steinberger, “Uniformly valid confidence intervals post-model-selection,” *The Annals of Statistics*, vol. 48, no. 1, pp. 440–463, 2020.
- [151] L. Fahrmeir, “Maximum likelihood estimation in misspecified generalized linear models,” *Statistics*, vol. 21, no. 4, pp. 487–502, 1990.
- [152] H. White, “Maximum likelihood estimation of misspecified models,” *Econometrica*, vol. 50, no. 1, pp. 1–25, 1982.
- [153] A. W. Van der Vaart, *Asymptotic statistics*, vol. 3. Cambridge University Press, 2000.
- [154] A. W. Van der Vaart, “Maximum likelihood estimation under a spatial sampling scheme,” *Annals of Statistics*, vol. 24, no. 5, pp. 2049–2057, 1996.
- [155] J.-M. Azaïs and M. Wschebor, *Level Sets and Extrema of Random Processes and Fields*. John Wiley & Sons, 2009.
- [156] F. Bachoc, F. Gamboa, J.-M. Loubes, and N. Venet, “A Gaussian process regression model for distribution inputs,” *IEEE Transactions on Information Theory*, vol. 64, no. 10, pp. 6620–6637, 2018.
- [157] O. Dubrule, “Cross validation of Kriging in a unique neighborhood,” *Mathematical Geology*, vol. 15, pp. 687–699, 1983.
- [158] F. Lavancier and P. Rochet, “A general procedure to combine estimators,” *Computational Statistics & Data Analysis*, vol. 94, pp. 175–192, 2016.
- [159] J. M. Bates and C. W. Granger, “The combination of forecasts,” *Journal of the Operational Research Society*, vol. 20, no. 4, pp. 451–468, 1969.
- [160] M. B. Salem, F. Bachoc, O. Roustant, F. Gamboa, and L. Tomaso, “Gaussian process based dimension reduction for goal-oriented sequential design,” *SIAM/ASA Journal on uncertainty quantification*, vol. 7, no. 4, pp. 1369–1397, 2019.
- [161] W. Xu and M. L. Stein, “Maximum likelihood estimation for a smooth Gaussian random field model,” *SIAM/ASA Journal on Uncertainty Quantification*, vol. 5, no. 1, pp. 138–175, 2017.
- [162] N. Cressie and S. Lahiri, “The asymptotic distribution of REML estimators,” *Journal of Multivariate Analysis*, vol. 45, pp. 217–233, 1993.
- [163] M. H. Neumann, “A central limit theorem for triangular arrays of weakly dependent random variables, with applications in statistics,” *ESAIM: Probability and Statistics*, vol. 17, pp. 120–134, 2013.
- [164] P. Doukhan, *Mixing, Properties and Examples*. Lecture Notes in Statistics, Springer-Verlag, New York, 1994.
- [165] T. Cacoullos, “On upper and lower bounds for the variance of a function of a random variable,” *The Annals of Probability*, pp. 799–809, 1982.
- [166] P. Billingsley, *Convergence of Probability Measures*. Wiley, New York, 1968.

- [167] N. Jenish and I. R. Prucha, “Central limit theorems and uniform laws of large numbers for arrays of random fields,” *Journal of Econometrics*, vol. 150, no. 1, pp. 86–98, 2009.
- [168] F. Bachoc, B. Broto, F. Gamboa, and J.-M. Loubes, “Gaussian field on the symmetric group: Prediction and learning,” *Electronic Journal of Statistics*, vol. 14, p. 503–546, 2020.
- [169] J. Betancourt, F. Bachoc, and T. Klein, “funGp: Gaussian process models for scalar and functional inputs.” <https://github.com/djbetancourt-gh/funGp>, 2020.
- [170] G. E. Box and D. R. Cox, “An analysis of transformations,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 26, no. 2, pp. 211–243, 1964.
- [171] R. M. Sakia, “The box-cox transformation technique: a review,” *Journal of the Royal Statistical Society: Series D (The Statistician)*, vol. 41, no. 2, pp. 169–178, 1992.

