

Gaussian process regression of two nested computer codes

Sophie Marque-Pucheu

▶ **To cite this version:**

Sophie Marque-Pucheu. Gaussian process regression of two nested computer codes. General Mathematics [math.GM]. Université Sorbonne Paris Cité, 2018. English. NNT : 2018USPCC155 . tel-02092072v4

HAL Id: tel-02092072
https://hal.science/tel-02092072v4

Submitted on 9 Dec 2019

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 de doctorat

de l'Université Sorbonne Paris Cité

Préparée à l'Université Paris Diderot

Ecole doctorale n°386 Mathématiques Paris Centre

Laboratoire de Probabilités, Statistique et Modélisation

Gaussian process regression of two nested computer codes

Par Sophie Marque-Pucheu

Thèse de doctorat de Mathématiques Appliquées

Présentée et soutenue publiquement à Paris le 10 octobre 2018 devant le jury suivant

Examinatrice	Fischer Aurélie	Maître de conférences	Université Paris Diderot
Directeur de thèse	Garnier Josselin	Professeur	École Polytechnique
Examinatrice	Marrel Amandine	Ingénieur de recherche	CEA
Rapporteur	Monod Hervé	Directeur de recherche	INRA
Président du jury	Nouy Anthony	Professeur	École Centrale Nantes
Examinateur	Perrin Guillaume	Ingénieur de recherche	CEA

D'après les rapports de

Monod Hervé	Directeur de recherche	INRA
Marzouk Youssef	Professor	MIT

This work is licensed under **https://creativecommons.org/licenses/by-nc-sa/4.0/**

Introduction

Remerciements

Mes premiers remerciements s'adressent naturellement aux deux encadrants qui m'ont accompagnée lors de ces trois années de thèse. Josselin Garnier, mon directeur de thèse, a suivi mes travaux de manière attentive. Sa très grande culture scientifique et sa réactivité ont été d'une aide précieuse. Guillaume Perrin, mon encadrant au CEA, a également fait preuve d'une très grande implication dans le suivi de cette thèse. Je le remercie pour sa pédagogie et sa confiance.
Je remercie également les deux rapporteurs de cette thèse : Hervé Monod et Youssef Marzouk. Thank you, Mr. Marzouk, for your careful reading of the manuscript. Merci également à M. Monod pour sa lecture attentive du manuscrit, ainsi que pour avoir été membre du jury.
Je tiens aussi à remercier Aurélie Fischer, Amandine Marrel et Anthony Nouy d'avoir accepté de faire partie de mon jury de thèse.
Enfin, je salue tous les membres de l'équipe incertitudes du CEA, ainsi que les doctorants du bâtiment B.
Je remercie également tous les personnels administratifs du CEA et l'Université Paris Diderot qui ont facilité d'une manière ou d'une autre mon quotidien.
Mes derniers remerciements vont à ma famille, en particulier mes parents qui m'ont transmis le goût d'apprendre. Last but not least, je remercie enfin mon conjoint pour son formidable soutien.

Résumé

Cette thèse traite de la métamodélisation (ou émulation) par processus gaussien de deux codes couplés. Le terme « deux codes couplés » désigne ici un système de deux codes chaînés : la sortie du premier code est une des entrées du second code.
Les deux codes sont coûteux. Afin de réaliser une analyse de sensibilité de la sortie du code couplé, on cherche à construire un métamodèle de cette sortie à partir d'un faible nombre d'observations. Trois types d'observations du système existent : celles de la chaîne complète, celles du premier code uniquement, celles du second code uniquement. Le métamodèle obtenu doit être précis dans les zones les plus probables de l'espace d'entrée.
Les métamodèles sont obtenus par krigeage universel, avec une approche bayésienne.
Dans un premier temps, le cas sans information intermédiaire, avec sortie scalaire, est traité. Une méthode innovante de définition de la fonction de la moyenne du processus gaussien, basée sur le couplage de deux polynômes, est proposée. Ensuite le cas avec information intermédiaire est traité. Un prédicteur basé sur le couplage des prédicteurs gaussiens associés aux deux codes est proposé. Des méthodes pour évaluer rapidement la moyenne et la variance du prédicteur obtenu sont proposées. Les résultats obtenus pour le cas scalaire sont ensuite étendus au cas où les deux codes sont à sortie de grande dimension. Pour ce faire, une méthode de réduction de dimension efficace de la variable intermédiaire de grande dimension est proposée pour faciliter la régression par processus gaussien du deuxième code. Les méthodes proposées sont appliquées sur des exemples numériques.

Mots-clés

Codes numériques emboîtés, codes couplés, codes chaînés, régression par processus gaussien, métamodélisation, variable fonctionnelle, réduction de dimension, Stepwise Uncertainty Reduction, plans d'expériences séquentiels.

Abstract

This thesis deals with the Gaussian process regression of two nested codes. The term "nested codes" refers to a system of two chained computer codes: the output of the first code is one of the inputs of the second code.
The two codes are computationally expensive. In order to perform a sensitivity analysis, we aim at emulating the output of the nested code from a small number of observations.
Three types of observations of the system exist: those of the chained code, those of the first code only and those of the second code only. The surrogate model has to be accurate on the most likely regions of the input domain of the nested code.
In this work, the surrogate models are constructed using the Universal Kriging framework, with a Bayesian approach.
First, the case when there is no information about the intermediary variable (the output of the first code) is addressed. An innovative parametrization of the mean function of the Gaussian process modeling the nested code is proposed. It is based on the coupling of two polynomials. Then, the case with intermediary observations is addressed. A stochastic predictor based on the coupling of the predictors associated with the two codes is proposed. Methods aiming at computing quickly the mean and the variance of this predictor are proposed. Finally, the methods obtained for the case of codes with scalar outputs are extended to the case of codes with high dimensional vectorial outputs.
We propose an efficient dimension reduction method of the high dimensional vectorial input of the second code in order to facilitate the Gaussian process regression of this code.

All the proposed methods are applied to numerical examples.

Keywords

Nested computer codes, Gaussian process regression, surrogate modeling, functional variable, dimension reduction, Stepwise Uncertainty Reduction, sequential designs.

Résumé long en français

Cette thèse présente de nouveaux développements pour la métamodélisation de codes coûteux chaînés, où la sortie du premier code est une des entrées du code suivant. Cette configuration et sa généralisation à plus que deux codes sont fréquemment rencontrées en pratique. Mais la construction de métamodèles adaptés à cette configuration a été peu étudiée jusqu'ici.

Ce manuscrit contient trois contributions nouvelles par rapport à l'état de l'art, détaillées dans les chapitres 3 à 5. La première contribution concerne la régression par processus gaussien avec une fonction de moyenne définie par une polynôme. Une nouvelle méthode de définition de la tendance polynomiale, basée sur la composition de deux polynômes, est proposée. Dans ce cas de figure, la variable intermédiaire entre les deux codes n'est pas connue.

La seconde contribution suppose la connaissance de la variable intermédiaire et traite de l'enrichissement du plan d'expériences en vue de la régression par processus gaussien de la sortie de la chaîne de deux codes. Le choix d'une nouvelle observation soulève plusieurs questions. Tout d'abord pour un code donné, il faut choisir les variables d'entrée de la nouvelle observation. Ensuite, comme il y a deux codes, la question se pose également (si cela est possible) de choisir auquel des deux codes ajouter une nouvelle observation.

La troisième contribution traite le cas de deux codes à sortie de très grande dimension (par exemple des fonctions du temps). Dans cette configuration, le second code a une sortie, mais également une entrée fonctionnelle. Une méthode de réduction de dimension de l'entrée fonctionnelle adaptée à ce cas est alors proposée. Les critères d'enrichissement proposés précédemment sont combinés avec cette méthode de réduction de dimension afin de les étendre au cas de deux codes à sortie fonctionnelle. Les méthodes proposées sont ensuite appliquées à un cas test industriel modélisant l'explosion d'une charge dans une cuve sphérique. Ce cas test est associé à un couplage entre un code de détonique et un code de dynamique des structures. Les paragraphes qui suivent présentent plus en détails la structure du manuscrit.

Le premier chapitre passe en revue l'état de l'art concernant la métamodélisation d'un unique code à entrée et sortie de faibles dimensions. Une brève présentation de la régression linéaire et du chaos polynomial est faite, ainsi que de méthodes de régularisation comme LASSO ou LARS. Le reste du chapitre est dédié à la régression par processus gaussien (GP) ou krigeage. Après un rappel des bases de la régression par processus gaussien, comme le choix de la fonction de covariance, le krigeage universel dans un cadre bayésien est présenté. Ensuite, les critères pour plans d'expériences pour la régression par processus gaussien et l'optimisation bayésienne sont passés en revue. Le chapitre se conclut sur une brève partie concernant l'analyse de sensibilité, en particulier les méthodes basées sur une décomposition de la variance (indices de Sobol).

Le deuxième chapitre passe en revue les méthodes pour la régression par processus gaussien d'un code à entrée et/ou sortie définie comme une fonction discrétisée du temps. L'attention se concentre ici sur la réduction de la dimension de l'entrée ou de la sortie. Concernant la réduction de la dimension de l'entrée, certaines méthodes ne prennent en compte que l'entrée fonctionnelle, tandis que d'autres ont pour objectif la réduction de la dimension de l'entrée de manière adaptée à la sortie. Ces dernières sont tout particulièrement adaptées pour le système chaîné considéré dans ce travail. Concernant la sortie fonctionnelle, deux approches sont possibles. La première consiste à projeter la sortie fonctionnelle sur une base de dimension réduite. La seconde repose sur l'utilisation d'une covariance tensorisée, où l'indice de la sortie

fonctionnelle (comme par exemple le temps) est considéré comme une des entrées du modèle.

Le troisième chapitre contient la première contribution de cette thèse : la construction d'une fonction de moyenne du processus gaussien par couplage de deux polynômes. Cette approche intègre l'information que l'on a a priori sur la structure chaînée des deux codes, mais sans observations ni connaissance de la structure de la variable intermédiaire. Dans ce cas, la configuration est proche d'une régression par processus gaussien classique, avec des observations des entrées et sortie de la chaîne de codes. La spécificité de la méthode repose sur l'utilisation de l'information que l'on a sur cette structure chaînée. La définition de la fonction de moyenne comprend une première étape de composition de deux polynômes, puis une seconde étape de linéarisation de cette composition. Cette linéarisation permet de limiter l'impact d'une erreur d'estimation des paramètres de chacun des deux polynômes. Ensuite le prédicteur de la sortie de la chaîne de code est construit en utilisant le krigeage universel dans un cadre bayésien. Par ailleurs, la structure proposée pour la tendance polynomiale offre une grande flexibilité, puisque les ordres totaux de chacun des deux polynômes, mais aussi la dimension de la sortie du premier polynôme, peuvent être optimisés. Cependant, cette flexibilité nécessite la résolution d'un problème d'optimisation complexe car non convexe. Une approche heuristique, basée sur une minimisation alternée par rapport aux variables, est proposée pour résoudre ce problème d'optimisation. Par ailleurs, un critère basé sur l'erreur Leave One Out (LOO) est utilisé pour caractériser la performance de prédiction du prédicteur gaussien. Ce critère est utilisé pour choisir la combinaison de valeurs la plus performante pour les ordres totaux des deux polynômes et la dimension de la sortie du premier polynôme.

Le quatrième chapitre contient la deuxième contribution de cette thèse : la métamodélisation de deux codes chaînés lorsque des observations de la variable intermédiaire sont disponibles. Le prédicteur proposé est basé sur un couplage de prédicteurs gaussiens de chacun des deux codes. Le chapitre propose en particulier deux critères d'enrichissement du plan d'expériences. Ces critères reposent sur une minimisation de la variance de prédiction intégrée (IMSE). La variance de prédiction doit donc être évaluée en un très grand nombre de points. Le premier critère correspond au cas où les deux codes ne peuvent pas être appelés de manière séparée. Le second correspond au cas où les codes peuvent être lancés de manière séparée. Dans ce cas, on peut choisir lequel des deux codes appeler, en retenant celui qui maximise la réduction de la variance de prédiction intégrée par unité de temps de calcul pour une évaluation du code. Une difficulté majeure liée à cette approche tient au fait que le couplage de deux prédicteurs gaussiens n'est pas gaussien. La variance de prédiction doit donc être évaluée en utilisant des méthodes de quadrature ou Monte Carlo. Afin de résoudre ces difficultés numériques, deux méthodes pour une évaluation rapide de la variance de prédiction sont proposées. Dans le premier cas, si le processus gaussien associé au second code a une fonction de covariance gaussienne et une tendance polynomiale, alors la variance peut être évaluée de manière analytique. Dans le cas où ces conditions ne sont pas valables, une autre approche reposant sur la linéarisation du couplage des deux prédicteurs peut être utilisée. Les méthodes proposées sont ensuite appliquées sur deux exemples numériques : un premier analytique et un second portant sur la trajectoire balistique d'un projectile conique. Les résultats obtenus montrent l'intérêt de prendre en compte les observations de la variable intermédiaire et de pouvoir appeler de manière séparée chacun des deux codes.

Le cinquième chapitre contient les contributions finales de cette thèse et concerne la métamodélisation par processus gaussien de deux codes chaînés à sortie fonctionnelle (de très grande dimension). La contribution majeure de ce chapitre est une méthode de réduction de l'entrée fonctionnelle d'un modèle linéaire, qui est adaptée à la sortie de ce modèle linéaire.

Cette méthode de réduction de dimension est combinée à une approximation de la sortie du second code, qui est linéaire par rapport à l'entrée fonctionnelle du second code (qui est également la sortie du premier code). Le modèle linéaire proposé est en fait un filtre causal, paramétré par un petit nombre de variables qui peuvent être estimées à partir d'un faible nombre d'observations.
Cette combinaison d'une approximation linéaire et d'une réduction de dimension adaptée à ce modèle linéaire permet de réduire la dimension de l'entrée fonctionnelle du second code de manière adaptée à la prédiction de la sortie de ce code.
Grâce à cette réduction de dimension, chacun des deux codes peut être associé à un processus gaussien avec un vecteur d'entrées de faible dimension. Deux prédicteurs gaussiens sont obtenus en utilisant une covariance tensorisée pour prendre en compte le caractère multidimensionnel des sorties des fonctions considérées. Les prédicteurs sont ensuite couplés et le couplage est linéarisé. Ceci permet d'obtenir un prédicteur gaussien de la sortie fonctionnelle de la chaîne de deux codes. La moyenne et la variance du prédicteur peuvent alors être évaluées de manière analytique, et donc très rapide. Les critères d'enrichissement proposés dans le chapitre précédent sont ensuite adaptés au cas de deux codes couplés à sortie fonctionnelle.
Enfin, les méthodes proposées sont mises en application sur le cas test industriel qui a motivé cette thèse, à savoir le couplage d'un code de détonique avec un code de dynamique des structures. Les sorties de chacun des codes sont des fonctions discrétisées du temps. Les résultats obtenus montrent l'intérêt de prendre en compte les observations de la variable intermédiaire, par rapport à une simple régression par processus gaussien de la sortie de la chaîne de codes en fonction des entrées.

Contents

Introduction i

Notations xv

I State of the art for the surrogate modeling of computer codes 1

1 Surrogate modeling of a single code with scalar inputs and output 5
1.1 Linear regression . . . 5
1.2 Polynomial Chaos Expansion . . . 6
1.3 Methods for the selection of the regressors of a linear model . . . 8
1.3.1 Stepwise and all-subsets regressions . . . 8
1.3.2 Ridge regression . . . 8
1.3.3 LASSO . . . 9
1.3.4 Forward stagewise regression . . . 9
1.3.5 Least Angle Regression . . . 9
1.3.6 Dantzig selector . . . 11
1.3.7 Conclusions . . . 11
1.4 Gaussian process regression or Kriging . . . 11
1.4.1 Gaussian processes . . . 11
1.4.2 Ordinary, simple and universal Kriging . . . 18
1.4.3 Estimation of a parametric covariance function . . . 20
1.5 Design of experiments . . . 22
1.5.1 Space-filling designs . . . 22
1.5.2 Criterion-based designs . . . 24
1.5.3 Gaussian processes for pointwise global optimization . . . 26
1.6 Sensitivity analysis . . . 26

2 Gaussian process regression of a code with a functional input or output 29
2.1 Dimension reduction of a functional variable . . . 29
2.1.1 Dimension reduction adapted to the functional variable only . . . 30
2.1.2 Dimension reduction adapted to the functional variable and a dependent variable . . . 31
2.2 Gaussian process prediction of a computer code with a functional output . . . 33
2.2.1 Projection of the functional output on a basis . . . 34
2.2.2 Gaussian process regression of the whole functional output . . . 35

II Contributions **37**

3 Nested polynomial trends for the improvement of Gaussian predictors **39**
3.1 Introduction . . . 39
3.2 Gaussian process predictors . . . 41
3.2.1 General framework . . . 41
3.2.2 Choice of the covariance function . . . 42
3.2.3 Choice of the mean function . . . 42
3.3 Nested polynomial trends for Gaussian process predictors . . . 43
3.3.1 Nested polynomial representations . . . 43
3.3.2 Coupling nested representations and Gaussian processes . . . 46
3.3.3 Linearization of the nested polynomial trend . . . 47
3.3.4 Error evaluation . . . 48
3.3.5 Convergence analysis . . . 50
3.4 Applications . . . 50
3.4.1 $d = 1$ . . . 51
3.4.2 $d > 1$ . . . 54
3.4.3 Relevance of the LOO error . . . 55
3.5 Conclusions . . . 57

4 Gaussian process regression of two nested codes with scalar output **59**
4.1 Introduction . . . 59
4.2 Surrogate modeling for two nested computer codes . . . 60
4.2.1 General framework . . . 60
4.2.2 Gaussian process-based surrogate models . . . 61
4.2.3 Sequential designs for the improvement of Gaussian process predictors 64
4.3 Fast computation of the variance of the predictor of the nested code . . . 65
4.3.1 Explicit derivation of the two first statistical moments of the predictor 66
4.3.2 Linearized approach . . . 67
4.4 Applications . . . 68
4.4.1 Characteristics of the examples . . . 69
4.4.2 Prediction performance for a given set of observations . . . 72
4.4.3 Performances of the sequential designs . . . 74
4.5 Conclusions . . . 78
4.6 Proofs . . . 79
4.6.1 Proof of Proposition 4.2.1 . . . 79
4.6.2 Proof of Lemma 4.3.1 . . . 79
4.6.3 Proof of Lemma 4.3.2 . . . 80
4.6.4 Proof of Proposition 4.3.1 . . . 81
4.6.5 Proof of Proposition 4.3.2 . . . 85
4.6.6 Proof of Corollary 4.3.3 . . . 85

5 Gaussian process regression of two nested codes with functional outputs **89**
5.1 Introduction . . . 89
5.2 Dimension reduction of the functional input of a code . . . 91
5.2.1 Formalism . . . 91
5.2.2 Dimension reduction of the functional input only . . . 92
5.2.3 Partial Least Squares regression . . . 93
5.2.4 A linear model-based dimension reduction of the functional input . . . 94
5.3 Gaussian process regression with low dimensional inputs and a functional output 96
5.4 Surrogate modeling of the nested code . . . 101

5.4.1 A linearized Gaussian predictor of the nested code 101
5.4.2 Sequential designs 102
5.5 First numerical example 104
5.5.1 Description of the numerical example 104
5.5.2 Dimension reduction of the functional input of the second code 106
5.5.3 Prediction of the nested code 111
5.5.4 Sensitivity analysis 115
5.6 Second numerical example 120
5.6.1 Description of the numerical example 120
5.6.2 Results of the numerical example 121
5.7 Conclusions 123
5.8 Proofs 125
5.8.1 Proof of Proposition 5.2.1 125
5.8.2 Proof of Lemma 1 126
5.8.3 Proof of Proposition 5.3.1 128
5.8.4 Proof of Proposition 5.4.1 129
5.8.5 Proof of Proposition 5.4.2 129

Conclusions **131**

Context

Surrogate modeling for the sensitivity analysis of two nested computer codes

This thesis is motivated by an application case. This application case is the coupling of two computationally costly computer codes. The first code is a detonation code and the second code is a structural dynamics code. The two codes have functional (i.e. high dimensional vectorial) outputs and the functional output of the first code is one of the inputs of the second code.
If we aim at performing design and certification studies of such a system, the evaluation of the output of the system at a large number of input points is often necessary. This is especially true when methods like sensitivity analysis, risk analysis or optimization are performed.
In this work we aim at performing a sensitivity analysis of the system mentioned above. Given the computational cost of the two codes, the first objective is to build an emulator, or a surrogate model, of the output of the two nested codes. This surrogate model will be constructed from a small set of observations of the two codes. The number of observations cannot be very high because of the computational costs of the codes.
As the role of simulation is increasing, the surrogate modeling of high-cost codes generates growing interest. However, the existing methods are generally applied to a single code or consider a system of codes as a single code.
In this work, the framework of the Gaussian process regression for the surrogate modeling of computer codes is considered. In this framework, the output of a code is considered to be the realization of a Gaussian process. The framework used for the Gaussian process regression is the Universal Kriging framework and a Bayesian approach is utilized. If some not very restrictive assumptions on the prior distribution of the Gaussian process are fulfilled, a Gaussian predictor of the code can be obtained by computing the posterior distribution of the Gaussian process given the observations of the code output.
Moreover, the existing methods for the surrogate modeling of codes generally consider the case of codes with low dimensional vectorial inputs. If a code has a functional input, the dimension of the functional input is often reduced thanks to a projection. The choice of the optimal method of dimension reduction of the functional input for the surrogate modeling of the output remains a research topic.

Contributions of the thesis

This thesis makes contributions to the surrogate modeling of two nested codes with scalar or functional outputs. These contributions aim at solving the following difficulties of the studied system:

- there are two codes,
- the codes are coupled by a functional intermediary variable,
- the second code has a functional input.

First, the case of two nested codes with scalar outputs is investigated. The considered system is then:

$$\begin{array}{ccccc} & & \boldsymbol{x}_2 & & \\ & & & \searrow & \\ \boldsymbol{x}_1 & \rightarrow & y_1(\boldsymbol{x}_1) & \nearrow & y_{\text{nest}}(\boldsymbol{x}_{\text{nest}}) := y_2(y_1(\boldsymbol{x}_1), \boldsymbol{x}_2), \end{array} \tag{0.0.1}$$

with $\boldsymbol{x}_1 \in \mathbb{R}^{d_1}$ and $\boldsymbol{x}_2 \in \mathbb{R}^{d_2}$ the low dimensional vectorial inputs of the two codes, $y_1 \in \mathbb{R}$ and $y_2 \in \mathbb{R}$ the output of the two codes, and d_1 and d_2 two integers.

In a first step, the case where there are no observations of the intermediary variable $y_1(\boldsymbol{x}_1)$ is considered. An innovative parametrization of the mean function of the Gaussian process is proposed. This parametrization is based on the coupling of polynomials and enables to improve the prediction accuracy compared to a classical constant or polynomial mean function.

Then the case where observations of the intermediary variable are available is considered.
A stochastic predictor of the nested code is obtained by coupling the Gaussian predictors of the two codes. Such an approach enables to take into account all the types of observations: observations of the nested code, of the first code only and of the second code only. The predictor is non-Gaussian but its moments can be computed using Monte Carlo methods.
Then we define sequential design criteria which aim at improving the prediction accuracy of the proposed predictor. The criteria are based on a reduction of the integrated prediction variance because the predictor has to be accurate on the most probable areas of the input domain for the sensitivity analysis. Finally, two adaptations of the proposed predictor are developed in order to evaluate the prediction variance and thus the proposed sequential design criteria quickly. The first adaptation is called "analytic" and the second one "linearized". They both enable to compute the mean and the variance of the proposed predictor in closed forms. The "linearized" method leads also to a Gaussian predictor of the nested code. Moreover, the interest of taking into account the intermediary observations is shown.

Finally, the case of two nested code with functional outputs is investigated. The considered system is then:

$$\begin{array}{ccccc} & & \boldsymbol{x}_2 & \searrow & \\ & & & & \boldsymbol{y}_{\text{nest}}(\boldsymbol{x}_{\text{nest}}) := \boldsymbol{y}_2(\boldsymbol{y}_1(\boldsymbol{x}_1), \boldsymbol{x}_2), \\ \boldsymbol{x}_1 & \rightarrow & \boldsymbol{y}_1(\boldsymbol{x}_1) & \nearrow & \end{array} \tag{0.0.2}$$

with $\boldsymbol{y}_1 \in \mathbb{R}^{N_t}$ and $\boldsymbol{y}_2 \in \mathbb{R}^{N_t}$ the output of the two codes when they are functional, $N_t \gg 1$ denoting the number of discretization steps of the functional outputs.
The second code has a functional input and the existing methods of Gaussian process regression generally consider low dimensional vectorial inputs. The Gaussian process regression of the second code requires therefore the reduction of the dimension of this functional input. We propose a dimension reduction of the functional input of a code which is suited for the prediction of the functional output of this code. This dimension reduction method is based on a two-step approach. First, the output of the second code is approximated by a linear causal filter. This linear model has a sparse structure, which is defined by only N_t variables. These variables can be estimated from a small set of observations of the functional input and output of the second code. The second step is the use of a proposed projection basis which is adapted to a linear model. The combination of these two steps enables to obtain a dimension reduction of the functional input of the second code, which:

- is adapted to the output of this code
- can be estimated from a small set of observations,
- does not require the knowledge of the derivatives of the output of the code,

Once the dimension of the functional intermediary variable has been efficiently reduced, the previously defined linearized method is adapted to the case of two nested codes with functional outputs. A Gaussian predictor of the functional output of the nested code, with analytic mean and variance, is obtained. Finally, the previously defined sequential design criteria are adapted to the case of two nested codes with functional outputs.

Outline of the manuscript

The thesis has two parts.

Part I provides a review of the state of the art for the surrogate modeling of computer codes.

In Chapter 1, we review methods for the surrogate modeling of a single code with low dimensional vectorial inputs and a scalar output.
Section 1.1 describes the surrogate modeling of a single code by Linear Regression.
Section 1.2 focuses on the surrogate modeling of a code by Polynomial Chaos Expansion.
Section 1.3 reviews the existing methods for the selection of the regressors in the framework of Linear Regression.
Section 1.4 provides a review of the Gaussian process regression framework for the surrogate modeling of a single code with low dimensional vectorial inputs and a scalar output.
Section 1.5 presents a review of the design of experiments for an accurate surrogate model on the whole input domain of a code with low dimensional vectorial inputs and a scalar output.
Section 1.6 focuses on the sensitivity analysis of the output of a code, or a quantity associated with it, with respect to the inputs of the code.

In Chapter 2, we review methods for the surrogate modeling of a single code with a functional output, low dimensional vectorial inputs and possibly a functional input.
Section 2.1 is devoted to the existing methods for the dimension reduction of a functional variable.
Section 2.2 reviews the existing methods for the Gaussian process regression of a code with low dimensional vectorial inputs and a functional output.

Part II details our contributions to the construction of a surrogate model of two nested codes with scalar or functional outputs.

In Chapter 3, we focus on the case where the two codes have scalar outputs and no observations of the intermediary variable are available. We propose to define the mean function of the Gaussian process modeling the nested code as a coupling of two polynomials. This parametrization is based on the coupling of two polynomials. We show how this parametrization can improve the prediction accuracy of the Gaussian predictor compared to the case where the mean function is defined by polynomials.

In Chapter 4 we focus on the case where the two codes have scalar outputs and observations of the intermediary variable are available. We propose a stochastic predictor of the nested code based on the coupling of the Gaussian predictors of the two codes. This stochastic predictor is non-Gaussian but its mean and variance can be evaluated using Monte Carlo methods. This predictor can take into account all the possible observations: those of the nested code, those of the first code and those of the second code. Then sequential design criteria are proposed. These design criteria aim at improving the prediction accuracy on the whole input domain of the nested code. One of the criteria can also take into account the difference of computational costs between the two codes. Finally, we propose two adaptations of the previously proposed predictor of the nested code in order to accelerate the computation of the mean and the variance of the predictor. They both enable to compute the prediction mean and variance in closed forms. In addition, the proposed linearized predictor of the nested code enables to obtain a Gaussian predictor of the nested code with conditioned mean and variance functions in closed forms.

The application of the proposed methods to numerical examples shows the interest of taking into account the intermediary observations.

In Chapter 5 we focus on the case of the coupling of two codes with functional outputs. We first propose an efficient dimension reduction of the functional input of the second code. This dimension reduction is based on a linear projection of the functional input of the second code. The proposed projection basis can be estimated from a small set of observations of the second code and does not require the knowledge of the derivatives of the code.
We also extend the linearized predictor of the nested code proposed in Chapter 4 to the case of two nested codes with scalar output. This extension relies on the dimension reduction of the functional output and a tensorized structure of the Gaussian process modeling the code. By tensorized structure we mean a separation between the index of the output and the inputs. The sequential design criteria are also adapted to the case of two nested codes with functional outputs.
The proposed methods are applied to numerical examples. The results show again the interest of taking appropriately into account the intermediary observations.
The predictor obtained at the end of the sequential enrichment of the initial design is used in order to perform a sensitivity analysis of a scalar quantity of interest based on the functional output of the nested code.

Notations

Ordinal variables

n	number of observations
d	dimension of an input variable
p	number of functions of a basis of function in the case of Universal Kriging
N_t	dimension of the time-varying output of a code
$\text{card}(\mathbb{A})$	number of elements of the set $\mathbb{A}$

Matrix, vectors and scalar

x	a scalar
$\boldsymbol{x}$	a vector
x_i or $(\boldsymbol{x})_i$	the i-th entry of the vector $\boldsymbol{x}$
$\boldsymbol{X}$	a matrix
$(\boldsymbol{X})_{ij}$	the entry at line i and row j of the matrix $\boldsymbol{X}$
$(\boldsymbol{X})_{\cdot i}$	the vector of the entries of the i-th column of the matrix $\boldsymbol{X}$
$(\boldsymbol{X})_{i\cdot}$	the vector of the entries of the i-th row of the matrix $\boldsymbol{X}$
$\boldsymbol{X}^T$	transpose of the matrix $\boldsymbol{X}$
$\text{diag}(\boldsymbol{x})$	diagonal matrix with diagonal $\boldsymbol{x}$
$\text{diag}(\boldsymbol{X})$	vector corresponding to the diagonal of the matrix $\boldsymbol{X}$
$\text{Tr}(\boldsymbol{X})$	trace of the matrix $\boldsymbol{X}$
$\text{cov}(\boldsymbol{x}, \boldsymbol{y})$	covariance between $\boldsymbol{x}$ and $\boldsymbol{y}$

Probabilistic notations

$\stackrel{d}{=}$	equality in distribution
$\mathbb{E}[\cdot]$	Mean of a random quantity
$\mathbb{V}[\cdot]$	Variance of a random quantity
$\mathcal{N}(\boldsymbol{m}, \boldsymbol{K})$	multivariate normal distribution with mean $\boldsymbol{m}$ and covariance matrix $\boldsymbol{K}$
$\mathrm{GP}(m(\cdot), C(\cdot,\cdot))$	one-dimensional Gaussian process with mean function m and covariance function C
$\mathrm{GP}(\boldsymbol{m}(\cdot), \boldsymbol{C}(\cdot,\cdot))$	multidimensional Gaussian process with vector-valued mean function $\boldsymbol{m}$ and matrix-valued covariance function $\boldsymbol{C}$

Norms and scalar products

$(\cdot,\cdot)_{\mathbb{X}}$	scalar product in the space of square integrable real-valued functions on $\mathbb{X}$, such that $(y,z)_{\mathbb{X}} := \int_{\mathbb{X}} y(\boldsymbol{x})z(\boldsymbol{x})d\boldsymbol{x}$
$\|\cdot\|_{\mathbb{X}}$	norm in the space of square integrable real-valued functions on $\mathbb{X}$, such that $\|y\|^2_{\mathbb{X}} := (y,y)_{\mathbb{X}}$
$\|\cdot\|_F$	Frobenius norm
$\|\cdot\|_1$	L_1 norm, such that $\|\boldsymbol{x}\|_1 = \sum_{i=1}^{d} \|x_i\|$
$\|\cdot\|$	L_2 norm, such that $\|\boldsymbol{x}\|_2 = \sqrt{\sum_{i=1}^{d} x_i^2}$

Part I

State of the art for the surrogate modeling of computer codes

The role of simulation for the design and the certification of complex systems is increasing. However, methods like uncertainty propagation, sensitivity analysis or optimization require the evaluation of the output of the code at a huge number of input points. If the computational cost of the computer code is high, and only a small number of observations of its output is available, the use of a surrogate model is necessary. In this part we review some existing methods for the surrogate modeling of computer codes.
This part includes two chapters. The first one is devoted to the surrogate modeling of a computer code with scalar (i.e. low dimensional vectorial) inputs and output. The second one focuses on the surrogate modeling with Gaussian process regression of a code with functional (i.e. high dimensional vectorial) input and/or output.

Chapter 1

Surrogate modeling of a single code with scalar inputs and output

In this chapter we consider a model of the form $\boldsymbol{x} \mapsto y(\boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{X} \subset \mathbb{R}^d$, d a positive integer, and $\mu_{\mathbb{X}}$ is a probability measure on the space comprising $\mathbb{X}$ and a σ-algebra over $\mathbb{X}$. The following sections detail the state of the art for the surrogate modeling of y from a set of n observations of the input and the output of the code. These observations are denoted by:

$$\boldsymbol{X}^{\text{obs}} = \begin{pmatrix} \boldsymbol{x}^{(1)} \\ \vdots \\ \boldsymbol{x}^{(n)} \end{pmatrix}, \tag{1.0.1}$$

and

$$\boldsymbol{y}^{\text{obs}} = \left(y^{(1)} = y\left(\boldsymbol{x}^{(1)}\right), \ldots, y^{(n)} = y\left(\boldsymbol{x}^{(n)}\right) \right), \tag{1.0.2}$$

where $\boldsymbol{X}^{\text{obs}}$ is a $(n \times d)$-dimensional matrix and $\boldsymbol{y}^{\text{obs}}$ is a n-dimensional vector.

The first section is devoted to linear regression. The second one deals with the use of Polynomial Chaos Expansion as a surrogate model. The third one focuses on the methods for the selection of regressors in regression models. The fourth one presents the Gaussian process regression for the surrogate modeling of a computer code. Finally, the last section reviews some existing designs of experiments which are adapted for the acquisition of knowledge of the computer code or the sequential improvement of a surrogate model.

1.1 Linear regression

Generalized additive models are a very common tool for the emulation of a response surface [Hastie and Tibshirani, 1990]. It is the projection of the output y on a basis of functions h_i, $1 \leq i \leq p$, p a positive integer, of the inputs $\boldsymbol{x}$. The emulator can be written in the form:

$$\widehat{y}(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x})^T \boldsymbol{\beta}, \tag{1.1.1}$$

where $\boldsymbol{h}(\boldsymbol{x})$ and $\boldsymbol{\beta}$ are in $\mathbb{R}^p$. The functions of the basis can be polynomials, with Polynomial Chaos Expansion as a particular case, wavelets, trigonometric functions...
Note that simple linear regression can be regarded as a particular case of the generalized additive models, with a basis of functions comprising only the covariates: $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{x}$.
The regression coefficients $\boldsymbol{\beta}$ can be estimated from a set of n observations of the inputs and the output of the code $\boldsymbol{X}^{\text{obs}}$ and $\boldsymbol{y}^{\text{obs}}$ through the minimization of the quadratic loss function:

$$\widehat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y\left(\boldsymbol{x}^{(i)}\right) - \boldsymbol{h}\left(\boldsymbol{x}^{(i)}\right)^T \boldsymbol{\beta} \right)^2. \tag{1.1.2}$$

If we denote:

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{h}\left(\boldsymbol{x}^{(1)}\right)^T \\ \vdots \\ \boldsymbol{h}\left(\boldsymbol{x}^{(n)}\right)^T \end{pmatrix}, \tag{1.1.3}$$

then the least squares estimate of the regression coefficients can be written:

$$\widehat{\boldsymbol{\beta}} = \boldsymbol{H}^{+}\boldsymbol{y}^{\text{obs}}, \tag{1.1.4}$$

where $\boldsymbol{H}^{+}$ is the pseudo-inverse of $\boldsymbol{H}$. If $n \geq p$ and $\boldsymbol{H}$ is of rank p, then $\boldsymbol{H}^T\boldsymbol{H}$ is invertible and $\boldsymbol{H}^{+} = \left(\boldsymbol{H}^T\boldsymbol{H}\right)^{-1}\boldsymbol{H}^T$. By definition, $\boldsymbol{H}$ is a $(n \times p)$-dimensional matrix.
However, matrix $\left(\boldsymbol{H}^T\boldsymbol{H}\right)$ is not always invertible. The number of observations can be smaller than the number of regression coefficients ($p \leq n$) or the functions of the basis can be correlated according to the probability measure $\mu_{\mathbb{X}}$, which means that the columns of $\boldsymbol{H}$ are correlated, thus reducing the rank of matrix $\boldsymbol{H}$.
The matrix $\left(\boldsymbol{H}^T\boldsymbol{H}\right)$ is more likely to be inverted if the basis functions are decorrelated with respect to the probability measure $\mu_{\mathbb{X}}$ of the inputs, as performed with Polynomial Chaos Expansion. Another possible approach is the use of a regularization term for the inversion of the matrix, or the selection of the most influencing regressors. The two following sections detail these two approaches.

1.2 Polynomial Chaos Expansion

Polynomial Chaos expansion can be used to emulate a model response y with inputs $\boldsymbol{x}$. Besides, the probability measure $\mu_{\mathbb{X}}$ associated with $\boldsymbol{x}$ is a product measure. Therefore, the components of the input vector are independent. It has been applied by Ghanem and Spanos [1990] to stochastic finite elements methods. Polynomial Chaos expansion can be seen as the projection of the model output y on a polynomial basis which depends on the distribution of the model inputs $\boldsymbol{x}$. The polynomials are orthonormal with respect to the distribution of $\boldsymbol{x}$. The model response can therefore be expanded as:

$$y\left(\boldsymbol{x}\right) = \sum_{\boldsymbol{\alpha}\in\mathbb{N}^d} \beta_{\boldsymbol{\alpha}}\Phi_{\boldsymbol{\alpha}}\left(\boldsymbol{x}\right), \tag{1.2.1}$$

with $\beta_{\boldsymbol{\alpha}} \in \mathbb{R}$ and $\Phi_{\boldsymbol{\alpha}}$ orthonormal multidimensional polynomials, which means:

$$\int_{\mathbb{X}} \Phi_{\boldsymbol{\alpha}}\left(\boldsymbol{x}\right)\Phi_{\gamma}\left(\boldsymbol{x}\right) d\mu_{\mathbb{X}}\left(\boldsymbol{x}\right) = \delta_{\boldsymbol{\alpha}\gamma}, \tag{1.2.2}$$

with $\delta_{\boldsymbol{\alpha}\gamma}$ denoting the Kronecker delta.
In practice, the expansion of Eq. (1.2.1) can be truncated in order to obtain a surrogate model of the model response. If we denote by $\mathbb{A} \subset \mathbb{N}^d$ the truncated set of indices, by $\boldsymbol{\beta}_{\mathbb{A}}$ the vector gathering the $\beta_{\boldsymbol{\alpha}}, \boldsymbol{\alpha} \in \mathbb{A}$ and by $\Phi_{\mathbb{A}}$ the vector gathering the selected polynomials, this surrogate model is defined as:

$$\widehat{y}\left(\boldsymbol{x}\right) = \boldsymbol{\Phi}_{\mathbb{A}}\left(\boldsymbol{x}\right)^T\boldsymbol{\beta}_{\mathbb{A}}. \tag{1.2.3}$$

Note that the truncation is generally defined by an upper bound r on the total order of the polynomials, which means $\mathbb{A} = \{\boldsymbol{\alpha} \in \mathbb{N}^d, \|\boldsymbol{\alpha}\|_1 \leq r\}$. The total order r can be chosen adaptively according to a target precision, with an estimation of the error thanks to a cross-validation criterion [Blatman and Sudret, 2010, 2011].
A coefficient $\beta_{\boldsymbol{\alpha}}$ is defined as the projection of the model response on function $\Phi_{\boldsymbol{\alpha}}$:

$$\beta_{\boldsymbol{\alpha}} = \int_{\mathbb{X}} y\left(\boldsymbol{x}\right)\Phi_{\boldsymbol{\alpha}}\left(\boldsymbol{x}\right) d\mu_{\mathbb{X}}\left(\boldsymbol{x}\right). \tag{1.2.4}$$

Distribution	Density	Orthonormal basis
Uniform	$\frac{1}{2}\mathbb{1}_{[-1,1]}(x)$	$\frac{P_k(x)}{\sqrt{2k+1}}$, with P_k Legendre polynomial
Gaussian	$\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right)$	$\frac{H_k(x)}{\sqrt{k!}}$, with H_k Hermite polynomial
Gamma	$\frac{x^a}{\Gamma(a+1)}\exp(-x)\,\mathbb{1}_{x>0}$	$\frac{L_k(x)}{\Gamma(k+a+1)}$, with L_k Laguerre polynomial

Table 1.1: Classical univariate polynomial families used for Polynomial Chaos Expansion.

The integral can be estimated using Monte-Carlo methods, quadrature rules [Ghiocel and Ghanem, 2002] or stochastic collocation methods [Xiu, 2009].
The coefficients can also be estimated by least squares regression [Blatman and Sudret, 2010, 2011] from a set of n observations:

$$\widehat{\boldsymbol{\beta}}_{\mathbb{A}} = \underset{\boldsymbol{\beta}_{\mathbb{A}}\in\mathbb{R}^{\mathrm{card}(\mathbb{A})}}{\operatorname{argmin}} \sum_{i=1}^{n}\left(y^{(i)} - \boldsymbol{\Phi}_{\mathbb{A}}\left(\boldsymbol{x}^{(i)}\right)^T\boldsymbol{\beta}_{\mathbb{A}}\right)^2. \tag{1.2.5}$$

Note that if the observations are drawn according to the distribution of the inputs, the meta-model will be more accurate in the high-probability regions of the input domain.
The usual one-dimensional polynomial families used for Polynomial Chaos Expansion, which are chosen according to the distribution of the one-dimensional variable x, are given in Table 1.1.
Furthermore, the inputs can be transformed using an isoprobabilistic transformation, such as the Nataf or the Rosenblatt transformations [Nataf, 1962; Rosenblatt, 1952; Lebrun and Dutfoy, 2009]. Such transformations map $\boldsymbol{x}$ to a d-dimensional standard Gaussian variable $\boldsymbol{\xi}$ (i.e. d independent standard Gaussian variables). Then a Polynomial Chaos Expansion can be performed using Hermite polynomials [Blatman and Sudret, 2011]. The expansion becomes:

$$y(\boldsymbol{x}) = \sum_{\boldsymbol{\alpha}\in\mathbb{N}^d}\beta_{\boldsymbol{\alpha}}\mathcal{H}_{\boldsymbol{\alpha}}(T(\boldsymbol{x})), \tag{1.2.6}$$

where $\mathcal{H}_{\boldsymbol{\alpha}} = \prod_{i=1}^{d} H_{\alpha_i}$ and

$$\beta_\alpha = \int_{T(\mathbb{X})} y\left(T^{-1}(\boldsymbol{\xi})\right)\mathcal{H}_{\boldsymbol{\alpha}}(\boldsymbol{\xi})\prod_{i=1}^{d}\varphi(\xi_i)\,d\boldsymbol{\xi}. \tag{1.2.7}$$

Here, $T:\boldsymbol{x}\mapsto\boldsymbol{\xi}$ is the isoprobabilistic transformation and T^{-1} its inverse, $\mathcal{H}_{\boldsymbol{\alpha}}$ are Hermite polynomials, and φ the standard univariate Gaussian probability density function.
Thanks to this isoprobabilistic transformation, the Polynomial Chaos Expansion of a computer code with dependent inputs can be performed.

1.3 Methods for the selection of the regressors of a linear model

In this section we review the existing methods for the selection of the most influential regressors for linear regression or Polynomial Chaos Expansion. The methods are presented in the chronological order of their appearance. Two approaches can be distinguished: the first one selects the regressors which are the most influential. The second one minimizes the coefficients associated with the least influential regressors.

1.3.1 Stepwise and all-subsets regressions

Stepwise regression aims at selecting the regressors which improve the prediction accuracy the most. There are three main approaches to perform this selection: forward selection, backward elimination and bidirectional elimination.
In the forward method, the set of the selected regressors is empty at the initial step. Then, at each step, one adds the regressor which best improves the prediction accuracy of the regression model. The addition continues until a stopping criterion is reached.
On the contrary, with the backward elimination, a huge number of regressors are selected at the initial step. Then the regressors which contribute the least to the prediction accuracy are removed step by step from the regression model.
Efroymson [1960] introduced an approach combining forward selection and backward elimination. At each step of the forward selection, the interest of removing one of the previous selected regressors is studied.
However, stepwise regression is known as being greedy and quite unstable [Hesterberg et al., 2008].

In parallel, all-subsets regression has been introduced by Furnival and Wilson [1974]. It relies on the evaluation of the accuracy of all the regression models based on all the subsets of the set of regressors. Even though exhaustive, this approach can be computationally expensive, especially when the number of regressors is high.

1.3.2 Ridge regression

Introduced by Hoerl and Kennard [1970], ridge regression is based on a penalization of the coefficients of the regressors. This penalization can be seen as a regularization of the regression problem. The coefficients obtained with the ridge regression are the solutions of the following optimization problem:

$$\widehat{\boldsymbol{\beta}}^{\text{ridge}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y\left(\boldsymbol{x}^{(i)}\right) - \boldsymbol{h}\left(\boldsymbol{x}^{(i)}\right)^T \boldsymbol{\beta} \right)^2 + \delta \left\|\boldsymbol{\beta}\right\|_2^2, \tag{1.3.1}$$

with δ a non-negative real-valued constant.
This leads to the normal equation:

$$\left(\boldsymbol{H}^T\boldsymbol{H} + \delta \boldsymbol{I}_p\right) \widehat{\boldsymbol{\beta}}^{\text{ridge}} = \boldsymbol{H}^T \boldsymbol{y}^{\text{obs}}. \tag{1.3.2}$$

Practically, the optimal value of δ can be estimated thanks to a Cross validation criterion. The absolute value of the coefficients decreases as δ increases. When $\delta = 0$, the result is the same as the one of ordinary least squares. If $\delta > 0$ then the matrix $\left(\boldsymbol{H}^T\boldsymbol{H} + \delta \boldsymbol{I}_p\right)$ is positive definite and thus invertible.

The ridge regression can be seen as a particular case of the Tikhonov regularization [Tikhonov

and Arsenin, 1977], which is defined as follows:

$$\widehat{\boldsymbol{\beta}}^{\text{Tikhonov}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y\left(\boldsymbol{x}^{(i)}\right) - \boldsymbol{h}\left(\boldsymbol{x}^{(i)}\right)^T \boldsymbol{\beta} \right)^2 + \|\boldsymbol{\Gamma}\boldsymbol{\beta}\|^2, \tag{1.3.3}$$

with $\boldsymbol{\Gamma}$ a $d \times d$-dimensional matrix.
If $\boldsymbol{\Gamma}^T\boldsymbol{\Gamma}$ is positive definite, this problem has the following explicit solution:

$$\widehat{\boldsymbol{\beta}}^{\text{Tikhonov}} = \left(\boldsymbol{H}^T\boldsymbol{H} + \boldsymbol{\Gamma}^T\boldsymbol{\Gamma}\right)^{-1} \boldsymbol{H}^T\boldsymbol{y}^{\text{obs}}. \tag{1.3.4}$$

Note that if $\boldsymbol{\Gamma}$ is defined such that $\boldsymbol{\Gamma}^T\boldsymbol{\Gamma}$ is positive definite, then the matrix $\boldsymbol{H}^T\boldsymbol{H} + \boldsymbol{\Gamma}^T\boldsymbol{\Gamma}$ is an invertible matrix.

1.3.3 LASSO

The Least Absolute Shrinkage and Selection Operator (LASSO) method has been introduced by Tibshirani [1989]. It relies on a L_1-penalization of the estimation of $\boldsymbol{\beta}$, which can be written:

$$\widehat{\boldsymbol{\beta}}^{\text{LASSO}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y\left(\boldsymbol{x}^{(i)}\right) - \boldsymbol{h}\left(\boldsymbol{x}^{(i)}\right)^T \boldsymbol{\beta} \right)^2 + \delta \|\boldsymbol{\beta}\|_1, \tag{1.3.5}$$

with δ a non-negative constant.
The higher δ is, the more zero coefficients there are and the sparser the regression model is.

1.3.4 Forward stagewise regression

Hastie et al. [2001] have introduced the forward stagewise regression. Although different from LASSO, it yields similar results. The procedure can be defined by the following algorithm:

- Initialize with $\boldsymbol{R} = \boldsymbol{y}^{\text{obs}}$ and $\beta_i = 0,\ i \in \{1,\dots,p\}$, then repeat until no regressor is correlated with $\boldsymbol{R}$:
  - Find $i \in \{1,\dots,p\}$ such that $\boldsymbol{h}_i\left(\boldsymbol{X}^{\text{obs}}\right)$ is the most correlated with $\boldsymbol{R}$,
  - Update $\beta_i = \beta_i + \epsilon_i$, $\epsilon_i = \epsilon \operatorname{sign}\left(\operatorname{cor}\left(\boldsymbol{h}_i\left(\boldsymbol{X}^{\text{obs}}\right), \boldsymbol{R}\right)\right)$,
  - Update $\boldsymbol{R} = \boldsymbol{R} - \epsilon_i \boldsymbol{h}_i\left(\boldsymbol{X}^{\text{obs}}\right)$,

where, by abuse of notation $\boldsymbol{h}_i\left(\boldsymbol{X}^{\text{obs}}\right) = \left(\boldsymbol{h}_i\left(\boldsymbol{x}^{(1)}\right), \dots, \boldsymbol{h}_i\left(\boldsymbol{x}^{(n)}\right)\right)$. In practice, ϵ is set to a small value, like $\epsilon = 0.01$. In general, this approach is more reliable than the classical stepwise regression.

1.3.5 Least Angle Regression

Introduced by Efron et al. [2004], Least Angle Regression (LAR) is similar to the forward stagewise regression, given that it selects the regressor $\boldsymbol{h}_i\left(\boldsymbol{X}^{\text{obs}}\right)$ which is the most correlated with the current residual $\boldsymbol{R}$. However, the computation of the value of β_i is different. Instead of being slightly modified, the value of β_i is chosen such that the correlation between the new residual $\boldsymbol{R} - \beta_i \boldsymbol{h}_i\left(\boldsymbol{X}^{\text{obs}}\right)$ and its most correlated regressor $\boldsymbol{h}_j\left(\boldsymbol{X}^{\text{obs}}\right)$ is equal to the correlation between $\boldsymbol{R} - \beta_i \boldsymbol{h}_i\left(\boldsymbol{X}^{\text{obs}}\right)$ and $\boldsymbol{h}_i\left(\boldsymbol{X}^{\text{obs}}\right)$. This method can also be seen as an intermediate method between forward regression and forward stagewise regression.

1.3.5.1 The algorithm

Least Angle Regression (LAR) is associated with the following algorithm:

1. Initialize with $\boldsymbol{R} = \boldsymbol{y}^{\text{obs}}$ and $\beta_i = 0,\ i \in \{1, \ldots, p\}$.
2. Find $i \in \{1, \ldots, p\}$ such that $\boldsymbol{h}_i\left(\boldsymbol{X}^{\text{obs}}\right)$ is the most correlated with $\boldsymbol{R}$.
3. Move β_i from 0 toward its least squares coefficient, until another regressor $\boldsymbol{h}_j\left(\boldsymbol{X}^{\text{obs}}\right)$ has as much correlation with $\boldsymbol{R} - \beta_i \boldsymbol{h}_i\left(\boldsymbol{X}^{\text{obs}}\right)$ as $\boldsymbol{h}_i\left(\boldsymbol{X}^{\text{obs}}\right)$.
4. Move jointly (β_i, β_j) in the direction defined by their joint least squares coefficient of the current residual on $\left(\boldsymbol{h}_i\left(\boldsymbol{X}^{\text{obs}}\right), \boldsymbol{h}_j\left(\boldsymbol{X}^{\text{obs}}\right)\right)$, until some regressor $\boldsymbol{h}_k\left(\boldsymbol{X}^{\text{obs}}\right)$ is as much correlated with the current residual.
5. Continue until $\min(p, n-1)$ regressors have been retained.

1.3.5.2 LASSO can be seen as specific case of LAR

Efron et al. [2004] and Hastie et al. [2007] have shown that a slightly modified LAR algorithm can provide the entire paths of the LASSO coefficients as the δ coefficient increases. This modified algorithm is defined as follows:

- Run the LAR algorithm from step 1 to 4,
- If a non-zero coefficient achieves zero, remove the associated regressor from the linear model and recompute the joint least squares direction,
- Continue until $\min(p, n-1)$ regressors have been retained.

In the same way, a modified LAR algorithm can be used to perform a forward stagewise regression in the case of $\epsilon \to 0$ [Hastie et al., 2007]. Note that the label LARS generally refers to this modified LAR algorithm (where S refers to Stagewise or LASSO).

1.3.5.3 Hybrid LARS

Introduced by Efron et al. [2004], hybrid LARS is derived from the original LARS (referring to the original LAR or LASSO here). This modified algorithm comprises a LAR step which enables to select the regressors. The next step is the estimation by ordinary least squares of the coefficients associated with the selected regressors.
Hybrid LARS relies on a separation between the choice of the regressors and the estimation of the linear model.
It enables to increase the accuracy of the linear model compared to the original LARS.

Relaxed LASSO [Meinshausen et al., 2007] is an extension of the LARS-based LASSO algorithm. The first step is the same as for hybrid LARS. The ordinary least squares estimation of the coefficients at the second step is replaced by a LASSO estimation with a small penalty. In this approach, for the selected regressors at a given step of the LARS algorithm, one performs LASSO with a small penalty coefficient δ, such that no regressor is eliminated. Hybrid LASSO is a particular case of this algorithm, with $\delta = 0$.

1.3.6 Dantzig selector

The Dantzig selector of Candes and Tao [2007] is based on the resolution of the following optimization problem:

$$\boldsymbol{\beta}^{\text{Dantzig}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \left\| \boldsymbol{H}^T \left(y^{\text{obs}} - \boldsymbol{H}\boldsymbol{\beta} \right) \right\|_\infty \quad \text{subject to } \|\boldsymbol{\beta}\|_1 \leq t, \tag{1.3.6}$$

with $t \in \mathbb{R}^+$
In the same way as LARS, the Dantzig selector sets some coefficients to zero, thus selecting some regressors.
However, Efron et al. [2004] and Meinshausen et al. [2007] have shown that the linear model obtained with LASSO is as accurate as or more accurate than the one obtained with the Dantzig selector.
Note that a DASSO (DAntzig Selector with Sequential Optimization) algorithm has been proposed by James et al. [2008] in order to compute in one step the whole path of the Dantzig selector.

1.3.7 Conclusions

In this section, methods which enable to select the regressors of a linear model have been reviewed. Such approaches are particularly useful when the number of observations n is small compared to the number of possible regressors p of the linear model.

1.4 Gaussian process regression or Kriging

This section is devoted to the surrogate modeling of a computer code by Gaussian Process Regression.
Gaussian process regression is widely used in computer experiments [Sacks et al., 1989; Santner et al., 2003; Rasmussen and Williams, 2006]. In the Gaussian process regression framework, the output y of the code can be seen as a realization of a Gaussian process.
In the remainder of the section, we first outline the multidimensional Gaussian distribution and the definition of a Gaussian process. Then the Gaussian process regression framework for a known covariance function is presented. Finally, the estimation of the hyperparameters of parametric covariance functions is described.

1.4.1 Gaussian processes

1.4.1.1 Multidimensional (multivariate) Gaussian distribution

A random vector $\boldsymbol{u} = (u_1, \ldots, u_n)$, $n \geq 1$, is a Gaussian vector if the following equivalent assumptions are verified:

- for any $\boldsymbol{a} \in \mathbb{R}^n$, $\boldsymbol{a}^T\boldsymbol{u}$ has a Gaussian distribution,
- the characteristic function of $\boldsymbol{u}$ is of the form $\boldsymbol{v} \mapsto \exp\left(i\boldsymbol{v}^T\boldsymbol{m} - \frac{1}{2}\boldsymbol{v}^T\boldsymbol{K}\boldsymbol{v} \right)$ with $\boldsymbol{m}$ a n-dimensional vector and $\boldsymbol{K}$ a $(n \times n)$-dimensional matrix, which is symmetric and positive definite.

If these assumptions are verified, we have $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{m}, \boldsymbol{K})$ with $\boldsymbol{m} = \mathbb{E}[\boldsymbol{u}]$ and $\boldsymbol{K} = \operatorname{cov}(\boldsymbol{u})$.

1.4.1.2 Gaussian processes

A random process associates to any value of $\boldsymbol{x}$ a random variable $Y(\boldsymbol{x})$. A random process is a Gaussian process if its finite-dimensional distributions are Gaussian distributions. A Gaussian process Y is characterized by its mean and covariance functions. The mean function is defined by:

$$m(\boldsymbol{x}) = \mathbb{E}[Y(\boldsymbol{x})]. \tag{1.4.1}$$

The covariance function is defined by:

$$C(\boldsymbol{x}, \boldsymbol{x}') = \operatorname{cov}\left(Y(\boldsymbol{x}), Y(\boldsymbol{x}')\right), \tag{1.4.2}$$

$\boldsymbol{x}'$ in $\mathbb{X}$.
A Gaussian process is said to be stationary if, for all $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}$ in $\mathbb{X}$ and $\boldsymbol{h} \in \mathbb{R}^d$ such that $\boldsymbol{x}^{(1)} + \boldsymbol{h}, \ldots, \boldsymbol{x}^{(n)} + \boldsymbol{h}$ are still in $\mathbb{X}$, the multidimensional distribution of the Gaussian process Y at $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}$ is the same as the one at $\boldsymbol{x}^{(1)} + \boldsymbol{h}, \ldots, \boldsymbol{x}^{(n)} + \boldsymbol{h}$.
It follows that a covariance function is said to be stationary, if, for all $\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{x} + \boldsymbol{h}, \boldsymbol{x}' + \boldsymbol{h} \in \mathbb{X}$, one has:

$$C(\boldsymbol{x} + \boldsymbol{h}, \boldsymbol{x}' + \boldsymbol{h}) = C(\boldsymbol{x}, \boldsymbol{x}') = C(\boldsymbol{x} - \boldsymbol{x}', \boldsymbol{0}). \tag{1.4.3}$$

Finally, a Gaussian process is stationary if and only if its mean function is constant and its covariance function is stationary.
The next section outlines some classical parametric families of stationary covariance functions and their properties. For a more detailed review of covariance functions, the interested reader may refer to Abrahamsen [1997] and Rasmussen and Williams [2006].

1.4.1.3 Parametric families of stationary covariance functions

Typical parametric families of covariance functions are of the form:

$$C(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 K_{\boldsymbol{\ell}}(\boldsymbol{x} - \boldsymbol{x}') \tag{1.4.4}$$

where $K_{\boldsymbol{\ell}}$ is a correlation function parametrized by the vector of correlation lengths $\boldsymbol{\ell} \in (0, +\infty)^d$, and $\sigma^2 \in (0, +\infty)$ is a variance parameter.
The following paragraphs present some classical stationary correlation functions $K_{\boldsymbol{\ell}}$.

The nugget correlation function
The nugget correlation function is defined by:

$$K_{\boldsymbol{\ell}}(\boldsymbol{x} - \boldsymbol{x}') = \delta_{\boldsymbol{x}=\boldsymbol{x}'}, \tag{1.4.5}$$

where δ denotes the Kronecker delta. Note that this covariance function does not depend on any correlation length.
By construction, the observations of a Gaussian process with a nugget correlation function are not correlated and consequently independent and identically distributed.
Figure 1.1 presents an example of a path of the centered Gaussian process with the nugget correlation function and a unit variance σ^2. The trajectory is very rough and all the observations are independent of each other.

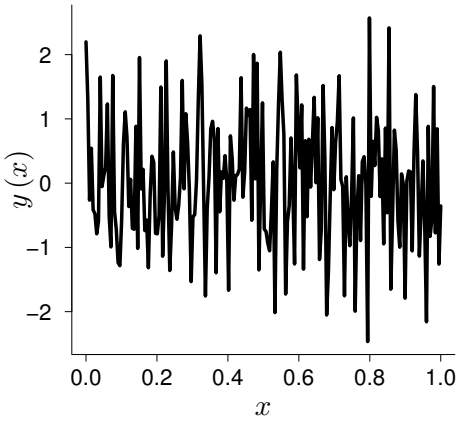


Figure 1.1: An example of a path of the centered Gaussian process with the nugget correlation function and a unit variance σ^2.

The squared exponential correlation function
The squared exponential (or Gaussian) correlation function is defined by:

$$K_{\boldsymbol{\ell}}\left(\boldsymbol{x}-\boldsymbol{x}'\right)=\exp\left(-d_{\boldsymbol{\ell}}\left(\boldsymbol{x}-\boldsymbol{x}'\right)^2\right), \tag{1.4.6}$$

where $d_{\boldsymbol{\ell}}\left(\boldsymbol{x}-\boldsymbol{x}'\right)=\sqrt{\sum_{i=1}^{d}\left(\frac{x_i-x_i'}{\ell_i}\right)^2}$. The trajectories of a Gaussian process with a squared exponential correlation function are infinitely differentiable. This covariance function is widely used in Kriging models. However, the assumption of infinite differentiability may be unrealistic [Stein, 1999].

Figure 1.2 presents the squared-exponential correlation function and an example of a path of the centered Gaussian process with a squared-exponential correlation function, a unit variance σ^2, and the following correlation lengths: $\ell \in \{0.05, 0.1, 0.2\}$. It can be seen that the shorter the correlation length is, the faster the correlation function decreases. Besides, the path varies more if the correlation length is short. Finally, note that the trajectories are very smooth, in agreement with their infinite differentiability.

The Matérn correlation function
The multi-dimensional Matérn kernel can be defined as:

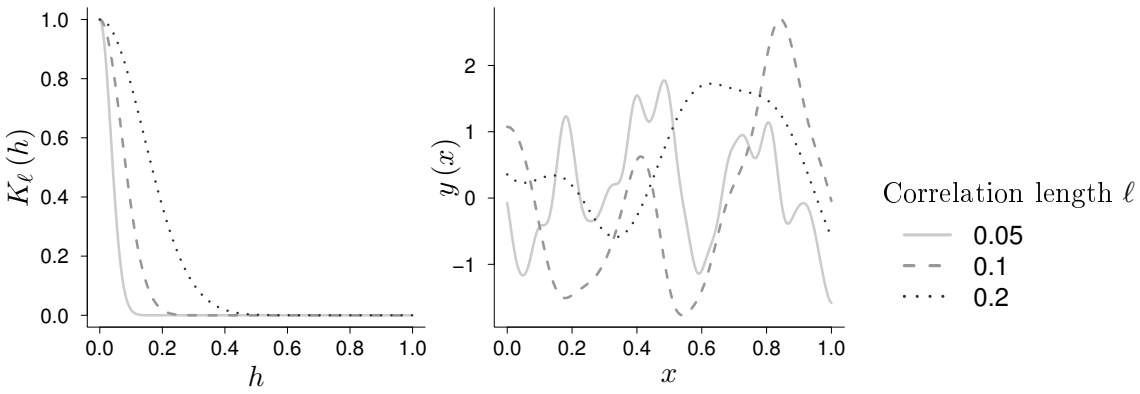
$$K_{\boldsymbol{\ell}}\left(\boldsymbol{x}-\boldsymbol{x}'\right)=\frac{1}{\Gamma\left(\nu\right)2^{\nu-1}}\left(2\sqrt{\nu}d_{\boldsymbol{\ell}}\left(\boldsymbol{x}-\boldsymbol{x}'\right)\right)^{\nu}K_{\nu}\left(2\sqrt{\nu}d_{\boldsymbol{\ell}}\left(\boldsymbol{x}-\boldsymbol{x}'\right)\right), \tag{1.4.7}$$

with $\Gamma\left(\cdot\right)$ the gamma function, K_ν a modified Bessel function [Abramowitz and Stegun, 1965] and $\nu \geq \frac{1}{2}$ the smoothness hyperparameter.
Note that as $\nu \to \infty$, the Matérn kernel tends to the squared exponential correlation function. Besides, when $\nu = k+\frac{1}{2}, k \in \mathbb{N}$, the Matérn kernel has a simpler form. In particular, we have:

- if $\nu = \frac{1}{2}$:

$$K_{\boldsymbol{\ell}}\left(\boldsymbol{x}-\boldsymbol{x}'\right)=\exp\left(-d_{\boldsymbol{\ell}}\left(\boldsymbol{x}-\boldsymbol{x}'\right)\right), \tag{1.4.8}$$

  this kernel is also known as the exponential kernel,



(a) Correlation function (b) Trajectories

Figure 1.2: On the left figure: plot of the squared-exponential correlation function. On the right plot: an example of a path of the centered Gaussian processes with a squared-exponential correlation function K_ℓ, $\ell \in \{0.05, 0.1, 0.2\}$ and a unit variance.

- if $\nu = \dfrac{3}{2}$:

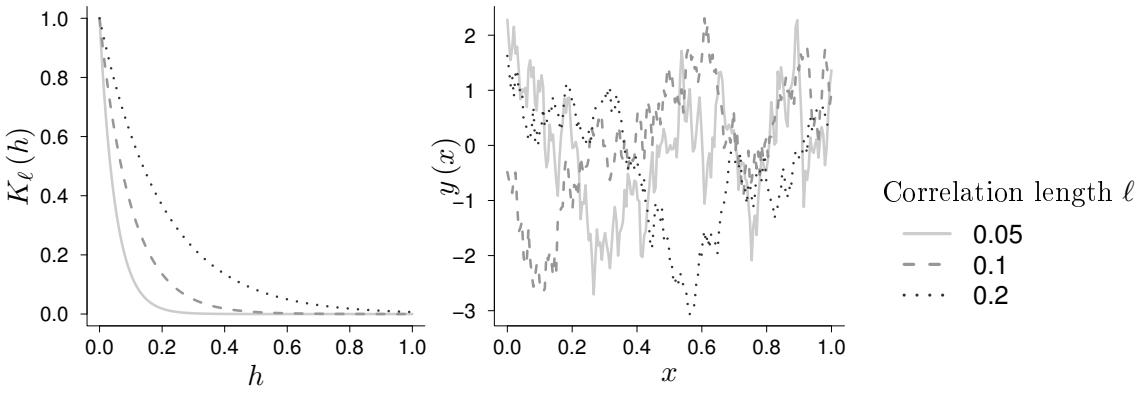
$$K_{\boldsymbol{\ell}}\left(\boldsymbol{x}-\boldsymbol{x}'\right) = \left(1+\sqrt{3}\; d_{\boldsymbol{\ell}}\left(\boldsymbol{x}-\boldsymbol{x}'\right)\right)\exp\left(-\sqrt{3}\; d_{\boldsymbol{\ell}}\left(\boldsymbol{x}-\boldsymbol{x}'\right)\right), \tag{1.4.9}$$

- if $\nu = \dfrac{5}{2}$:

$$K_{\boldsymbol{\ell}}\left(\boldsymbol{x}-\boldsymbol{x}'\right) = \left(1+\sqrt{5}\; d_{\boldsymbol{\ell}}\left(\boldsymbol{x}-\boldsymbol{x}'\right)+\frac{5}{3}\; d_{\boldsymbol{\ell}}\left(\boldsymbol{x}-\boldsymbol{x}'\right)^2\right)\exp\left(-\sqrt{5}\; d_{\boldsymbol{\ell}}\left(\boldsymbol{x}-\boldsymbol{x}'\right)\right). \tag{1.4.10}$$

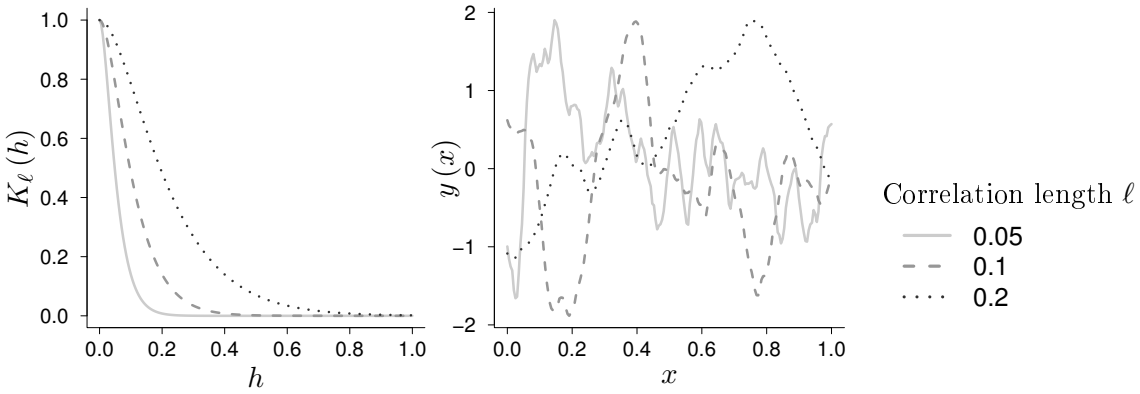
Figure 1.3 presents the exponential correlation function and an example of a path of the centered Gaussian processes with an exponential correlation function, a unit variance σ^2 and the following correlation lengths: $\ell \in \{0.05, 0.1, 0.2\}$. The trajectories are not differentiable.
Figure 1.4 presents the Matérn $\dfrac{3}{2}$ correlation function and examples of a path of the centered Gaussian processes with a Matérn $\dfrac{3}{2}$ correlation function, a unit variance σ^2, and the following correlation lengths: $\ell \in \{0.05, 0.1, 0.2\}$. The trajectories are not very smooth, but smoother than with the exponential correlation function.
Figure 1.5 presents the Matérn $\dfrac{5}{2}$ correlation function and an example of a path of the centered Gaussian processes with a Matérn $\dfrac{5}{2}$ correlation function, a unit variance σ^2, and the following correlation lengths: $\ell \in \{0.05, 0.1, 0.2\}$. The trajectories are relatively smooth.
It can be seen on Figures 1.2 to 1.5 that the shorter the correlation length is, the faster the correlation function decreases. Besides, the path varies more if the correlation length is short.

Figure 1.6 presents the Matérn correlation function and examples of a path of the centered Gaussian processes with a Matérn correlation function, a correlation length equal to 0.5, a unit variance σ^2, and the following values of the smoothness parameter: $\nu \in \{\frac{1}{2}, \frac{3}{2}, \frac{5}{2}, \infty\}$. It can be seen that the smoothness parameter strongly impacts the form of the correlation function. Besides, the higher ν is, the smoother the paths are.



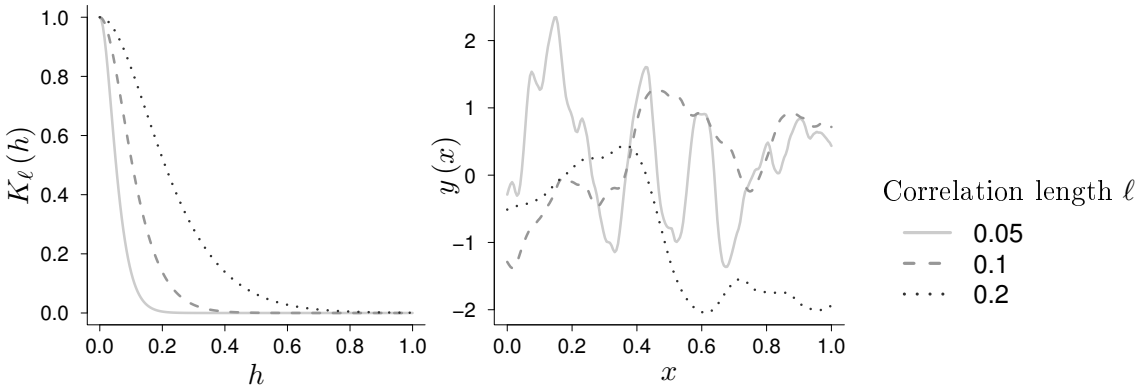
(a) Correlation function (b) Trajectories

Figure 1.3: On the left figure: plot of the exponential correlation function. On the right plot: an example of paths of the centered Gaussian processes with an exponential correlation function K_ℓ, $\ell \in \{0.05, 0.1, 0.2\}$ and a unit variance.



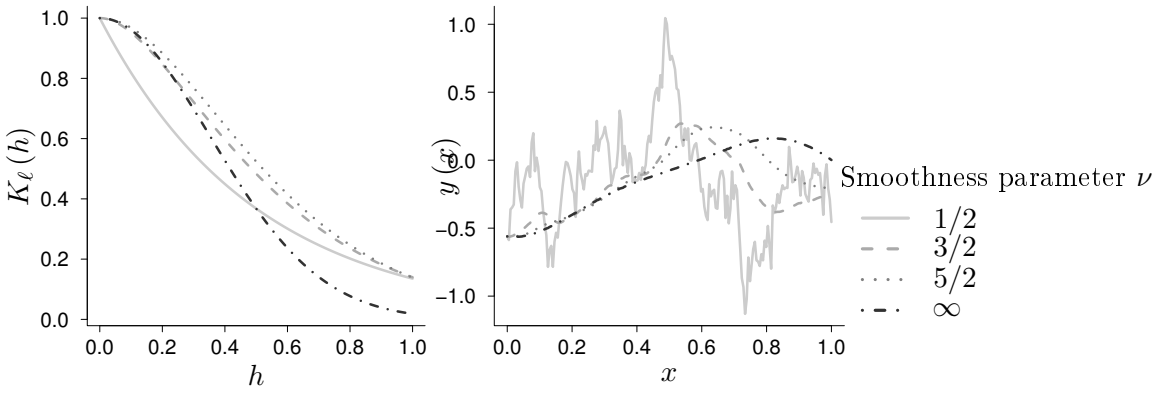
(a) Correlation function (b) Trajectories

Figure 1.4: On the left figure: plot of the Matérn $\frac{3}{2}$ correlation function. On the right plot: an example of a path of the centered Gaussian processes with a Matérn $\frac{3}{2}$ correlation function K_ℓ, a unit variance and $\ell \in \{0.05, 0.1, 0.2\}$.



(a) Correlation function (b) Trajectories

Figure 1.5: On the left figure: plot of the Matérn $\frac{5}{2}$ correlation function. On the right plot: examples of a path of the centered Gaussian processes with a Matérn $\frac{5}{2}$ correlation function K_ℓ, a unit variance and $\ell \in \{0.05, 0.1, 0.2\}$.



(a) Correlation function (b) Trajectories

Figure 1.6: On the left figure: plot of the Matérn correlation function for different values of the smoothness parameter ν. On the right plot: an example of a path of the centered Gaussian processes with a Matérn correlation function K_ℓ, the following values of the smoothness parameter $\nu \in \{\frac{1}{2}, \frac{3}{2}, \frac{5}{2}, \infty\}$, a correlation length ℓ equal to 0.5 and a unit variance σ^2.

The power exponential correlation function
The power exponential correlation kernel is defined by:

$$K_{\boldsymbol{\ell}}\left(\boldsymbol{x}-\boldsymbol{x}'\right)=\exp\left(-\sum_{i=1}^{d}\left(\frac{x_i-x_i'}{\ell_i}\right)^p\right), \tag{1.4.11}$$

$p\in(0,2]$, with the particular case of $p=2$ corresponding to the squared exponential correlation function.

Finally, note that the multidimensional correlation functions can also be defined as a product of univariate correlation functions:

$$K_{\boldsymbol{\ell}}\left(\boldsymbol{x}-\boldsymbol{x}'\right)=\prod_{i=1}^{d}K_{\ell_i}\left(x_i-x_i'\right), \tag{1.4.12}$$

where the K_{ℓ_i} may belong to different families of correlation functions.

1.4.1.4 The relationship between the covariance function and the mean square regularity

In this section, we consider a centered Gaussian process Y with covariance function C. Some properties concerning the mean square regularity of a centered Gaussian process and its relationship with the covariance function are reviewed.

A zero-mean Gaussian process Y is mean square continuous if and only if its covariance function is continuous at each pair $(\boldsymbol{x},\boldsymbol{x})$, $\boldsymbol{x}\in\mathbb{X}$. Besides, if a covariance function is continuous at each pair $(\boldsymbol{x},\boldsymbol{x})$, $\boldsymbol{x}\in\mathbb{X}$, then it is continuous on $\mathbb{X}\times\mathbb{X}$ [Bachoc, 2013b].

If one defines the following notation:

$$\operatorname{cov}\left(\frac{\partial Y\left(\boldsymbol{x}\right)}{\partial x_i},\frac{\partial Y\left(\boldsymbol{x}'\right)}{\partial x_i'}\right)=\frac{\partial^2 C}{\partial x_i\partial x_i'}\left(\boldsymbol{x},\boldsymbol{x}'\right), \tag{1.4.13}$$

the derivative $\dfrac{\partial}{\partial x_{i_1}}\ldots\dfrac{\partial}{\partial x_{i_k}}Y$, with $\{i_1,\ldots,i_k\}$ a subset of $\{1,\ldots,d\}$, exists in the mean square sense and is a Gaussian process if the derivative function $\dfrac{\partial^2}{\partial x_{i_1}\partial x_{i_1}'}\ldots\dfrac{\partial^2}{\partial x_{i_k}\partial x_{i_k}'}C$ exists and is finite.

In the case of a Gaussian process Y with stationary covariance function C, the three following assumptions are a consequence of the previous assumptions:

1. the Fourier transform $\widehat{C}$ of C is such that:

$$\int_{\mathbb{R}}\omega^{2k}\widehat{C}\left(\omega\right)d\omega<+\infty,$$

2. the covariance function C of Y is $2k$ times differentiable,

3. Y is k times mean square differentiable.

1.4.2 Ordinary, simple and universal Kriging

The term Kriging [Matheron and Blondel, 1962] refers to the prediction of the value of a random field at unobserved points of this random field. In this work, we assume that the random field is a Gaussian process.
In the framework of Kriging, three cases can be distinguished according to different assumptions on the mean function:

- Simple Kriging corresponds to the case where the mean function is known. Then, thanks to the subtraction of this known mean, the Gaussian process can be assumed to be centered.
- Ordinary Kriging corresponds to the case where the mean function is assumed to be constant and unknown.
- Universal Kriging corresponds to the case where the mean function is unknown and of the form $m(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x})^T \boldsymbol{\beta}$, where $\boldsymbol{h}(\boldsymbol{x})$ defines a p-dimensional basis of functions and $\boldsymbol{\beta} \in \mathbb{R}^p$ a vector of unknown coefficients.

Note that, if the covariance function of the Gaussian process is considered as being stationary, the use of a non-stationary mean function (universal Kriging) can make this assumption of stationarity of the covariance function more likely.

In the following paragraphs, we review the predictors obtained by the computation of the conditioned mean and variance of the Gaussian process in the frameworks of simple, ordinary and universal Kriging. At this stage, the covariance function of the Gaussian process is assumed to be known. Besides, we consider a Bayesian framework [Robert, 2007; Santner et al., 2003].
The following notations will be used. The prior distribution of the Gaussian process Y can be denoted by:

$$Y(\cdot)\,|m, C \sim \mathrm{GP}\left(m(\cdot), C(\cdot,\cdot)\right), \tag{1.4.14}$$

and the posterior distribution of the Gaussian process Y by:

$$Y(\cdot)\,|\boldsymbol{y}^{\mathrm{obs}}, m, C \sim \mathrm{GP}\left(m^c(\cdot), C^c(\cdot,\cdot)\right). \tag{1.4.15}$$

1.4.2.1 Simple Kriging

Simple Kriging corresponds to the case of a Gaussian process with known mean. For the sake of simplicity, this mean is assumed to be set at zero, thanks to the subtraction of the known mean of the Gaussian Process. Thus, one has:

$$m(\boldsymbol{x}) = 0, \tag{1.4.16}$$

and:

$$Y(\cdot)\,|C \sim \mathrm{GP}\left(0, C(\cdot,\cdot)\right). \tag{1.4.17}$$

In such a case, the conditioned distribution of the Gaussian process is still Gaussian, with conditioned mean and variance which are given by:

$$m^c(\boldsymbol{x}) = C\left(\boldsymbol{x}, \boldsymbol{X}^{\mathrm{obs}}\right) C\left(\boldsymbol{X}^{\mathrm{obs}}, \boldsymbol{X}^{\mathrm{obs}}\right)^{-1} \boldsymbol{y}^{\mathrm{obs}}, \tag{1.4.18}$$

and

$$C^c\left(\boldsymbol{x}, \boldsymbol{x}'\right) = C\left(\boldsymbol{x}, \boldsymbol{x}'\right) - C\left(\boldsymbol{x}, \boldsymbol{X}^{\mathrm{obs}}\right) C\left(\boldsymbol{X}^{\mathrm{obs}}, \boldsymbol{X}^{\mathrm{obs}}\right)^{-1} C\left(\boldsymbol{X}^{\mathrm{obs}}, \boldsymbol{x}\right), \tag{1.4.19}$$

where $\boldsymbol{X}^{\text{obs}}$ is defined by Eq. (1.0.1) and $C\left(\boldsymbol{x}, \boldsymbol{X}^{\text{obs}}\right)$ is a n-dimensional vector and $C\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{X}^{\text{obs}}\right)$ is a $(n \times n)$-dimensional matrix, so that:

$$\left(C\left(\boldsymbol{x}, \boldsymbol{X}^{\text{obs}}\right)\right)_i = C\left(\boldsymbol{x}, \boldsymbol{x}^{(i)}\right), \tag{1.4.20}$$

and

$$\left(C\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{X}^{\text{obs}}\right)\right)_{ij} = C\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right). \tag{1.4.21}$$

1.4.2.2 Ordinary Kriging

Ordinary Kriging can be regarded as a specific case of Universal Kriging, with constant mean $\beta \in \mathbb{R}$ to be determined:

$$m\left(\boldsymbol{x}\right) = \beta. \tag{1.4.22}$$

Therefore, in the case of Ordinary Kriging, one has:

$$Y\left(\cdot\right)|\beta, C \sim \text{GP}\left(\beta, C\left(\cdot,\cdot\right)\right). \tag{1.4.23}$$

We consider a Bayesian framework and we have no a priori information about β. The prior distribution of β is therefore assumed to be an improper uniform distribution on $\mathbb{R}$. In such a framework, the conditioned distribution of the Gaussian process is still Gaussian, with the following conditioned mean and variance functions:

$$m^c\left(\boldsymbol{x}\right) = \widehat{\beta} + C\left(\boldsymbol{x}, \boldsymbol{X}^{\text{obs}}\right) C\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{X}^{\text{obs}}\right)^{-1}\left(y\left(\boldsymbol{x}\right) - \widehat{\beta}\right), \tag{1.4.24}$$

and

$$\begin{aligned} C^c\left(\boldsymbol{x}, \boldsymbol{x}'\right) = \; & C\left(\boldsymbol{x}, \boldsymbol{x}'\right) - C\left(\boldsymbol{x}, \boldsymbol{X}^{\text{obs}}\right) C\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{X}^{\text{obs}}\right)^{-1} C\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{x}\right) + \\ & \boldsymbol{u}\left(\boldsymbol{x}\right)\left(\mathbb{1}^T C\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{X}^{\text{obs}}\right)^{-1}\mathbb{1}\right)^{-1}\boldsymbol{u}\left(\boldsymbol{x}'\right), \end{aligned} \tag{1.4.25}$$

where

$$\boldsymbol{u}\left(\boldsymbol{x}\right) = 1 - \mathbb{1}^T C\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{X}^{\text{obs}}\right)^{-1} C\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{x}\right), \tag{1.4.26}$$

$$\widehat{\beta} = \left(\mathbb{1}^T C\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{X}^{\text{obs}}\right)^{-1}\mathbb{1}\right)^{-1}\mathbb{1}^T C\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{X}^{\text{obs}}\right)^{-1}\boldsymbol{y}^{\text{obs}}, \tag{1.4.27}$$

and:

$$\mathbb{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}. \tag{1.4.28}$$

1.4.2.3 Universal Kriging

In the case of Universal Kriging, the mean function of the Gaussian process is defined as follows:

$$m\left(\boldsymbol{x}\right) = \boldsymbol{h}\left(\boldsymbol{x}\right)^T\boldsymbol{\beta}, \tag{1.4.29}$$

with $\boldsymbol{\beta}$ a vector of unknown parameters.
Therefore, in the case of Universal Kriging, the prior distribution of the Gaussian process is:

$$Y\left(\cdot\right)|\boldsymbol{h}, \boldsymbol{\beta}, C \sim \text{GP}\left(\boldsymbol{h}\left(\cdot\right)^T\boldsymbol{\beta}, C\left(\cdot,\cdot\right)\right). \tag{1.4.30}$$

If we assume that $\boldsymbol{\beta}$ follows an improper uniform distribution on $\mathbb{R}^p$ and that the covariance function is known, then the conditional distribution of the Gaussian process is still Gaussian and its conditioned mean and covariance functions can be computed analytically. The conditioned mean and variance of the Gaussian process can be written:

$$m^c(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x})^T \widehat{\boldsymbol{\beta}} + C\left(\boldsymbol{x}, \boldsymbol{X}^{\text{obs}}\right) C\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{X}^{\text{obs}}\right)^{-1} \left(\boldsymbol{y}^{\text{obs}} - \boldsymbol{H}\widehat{\boldsymbol{\beta}}\right), \tag{1.4.31}$$

and

$$\begin{aligned} C^c(\boldsymbol{x}, \boldsymbol{x}') = \; & C(\boldsymbol{x}, \boldsymbol{x}') - C\left(\boldsymbol{x}, \boldsymbol{X}^{\text{obs}}\right) C\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{X}^{\text{obs}}\right)^{-1} C\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{x}\right) + \\ & \boldsymbol{u}(\boldsymbol{x})^T \left(\boldsymbol{H}^T C\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{X}^{\text{obs}}\right)^{-1} \boldsymbol{H}\right)^{-1} \boldsymbol{u}(\boldsymbol{x}'), \end{aligned} \tag{1.4.32}$$

where

$$\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{H}^T C\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{X}^{\text{obs}}\right)^{-1} C\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{x}\right), \tag{1.4.33}$$

and:

$$\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{H}^T C\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{X}^{\text{obs}}\right)^{-1} \boldsymbol{H}\right)^{-1} \boldsymbol{H}^T C\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{X}^{\text{obs}}\right)^{-1} \boldsymbol{y}^{\text{obs}}. \tag{1.4.34}$$

Besides, the posterior distribution of the parameters $\boldsymbol{\beta}$ is Gaussian with mean $\widehat{\boldsymbol{\beta}}$ and covariance:

$$\boldsymbol{R}_{\boldsymbol{\beta}} = \left(\boldsymbol{H}^T C\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{X}^{\text{obs}}\right)^{-1} \boldsymbol{H}\right)^{-1}. \tag{1.4.35}$$

Note that the classical linear regression leads to the same results as Universal Kriging with a nugget covariance function. A nugget covariance function is defined by $C(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \delta_{\boldsymbol{x}=\boldsymbol{x}'}$, with δ denoting the Kronecker delta.

1.4.3 Estimation of a parametric covariance function

The previous section has detailed the properties of a Gaussian process and has presented some parametric families of covariance function and the conditioned distribution of the Gaussian process for several assumptions on the mean function of the process and a known covariance function.

In this section, we review some methods of estimation of the hyperparameters of the covariance function, when the covariance function belongs to a known parametric family.
There are two main approaches for the plug-in estimation of the covariance hyperparameters $\boldsymbol{\ell}$ and σ^2. The first one is based on the maximization of the likelihood of the observations given the hyperparameters. The second one is based on the minimization of the Leave One Out Mean Square Error for the estimation of $\boldsymbol{\ell}$ and on the Leave One Out Prediction Variance for the estimation of σ^2. Alternatively, a full Bayesian approach can be used [Robert, 2007]. But, in such a case the posterior distribution of the Gaussian process is no longer Gaussian.

1.4.3.1 Maximum Likelihood Estimation

By definition of the prior distribution of the Gaussian process modeling the code, one can write:

$$\boldsymbol{y}^{\text{obs}} \mid \boldsymbol{\beta}, \boldsymbol{\ell}, \sigma^2 \sim \mathcal{N}\left(\boldsymbol{H}\boldsymbol{\beta}, \sigma^2 K_{\boldsymbol{\ell}}\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{X}^{\text{obs}}\right)\right), \tag{1.4.36}$$

with $K_{\boldsymbol{\ell}}$ such that $C(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 K_{\boldsymbol{\ell}}(\boldsymbol{x}, \boldsymbol{x}')$.
The log-likelihood of the observations can therefore be written as a function of $\boldsymbol{\ell}$, σ^2 and $\boldsymbol{\beta}$:

$$\mathcal{L}\left(\boldsymbol{\beta}, \boldsymbol{\ell}, \sigma^2\right) = -\frac{1}{2}\ln\left|\sigma^2 \boldsymbol{R}_{\boldsymbol{\ell}}\right| - \frac{1}{2}\frac{1}{\sigma^2}\left(\boldsymbol{y}^{\text{obs}} - \boldsymbol{H}\boldsymbol{\beta}\right)^T \boldsymbol{R}_{\boldsymbol{\ell}}^{-1}\left(\boldsymbol{y}^{\text{obs}} - \boldsymbol{H}\boldsymbol{\beta}\right), \tag{1.4.37}$$

with $\boldsymbol{R}_{\boldsymbol{\ell}} = K_{\boldsymbol{\ell}}\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{X}^{\text{obs}}\right)$.

The derivatives of $\mathcal{L}\left(\boldsymbol{\beta}, \boldsymbol{\ell}, \sigma^2\right)$ with respect to $\boldsymbol{\beta}$ and σ^2 are defined as follows:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}}\left(\boldsymbol{\beta}, \boldsymbol{\ell}, \sigma^2\right) = \frac{1}{2}\frac{1}{\sigma^2}\boldsymbol{H}^T \boldsymbol{R}_{\boldsymbol{\ell}}^{-1}\left(\boldsymbol{y}^{\text{obs}} - \boldsymbol{H}\boldsymbol{\beta}\right), \tag{1.4.38}$$

$$\frac{\partial \mathcal{L}}{\partial \sigma^2}\left(\boldsymbol{\beta}, \boldsymbol{\ell}, \sigma^2\right) = -\frac{n}{2\sigma^2} + \frac{1}{2}\frac{1}{\sigma^4}\left(\boldsymbol{y}^{\text{obs}} - \boldsymbol{H}\boldsymbol{\beta}\right)^T \boldsymbol{R}_{\boldsymbol{\ell}}^{-1}\left(\boldsymbol{y}^{\text{obs}} - \boldsymbol{H}\boldsymbol{\beta}\right). \tag{1.4.39}$$

From Eqs. (1.4.38) and (1.4.39), it can be inferred that the maximization of the log-likelihood criterion $\mathcal{L}\left(\boldsymbol{\beta}, \boldsymbol{\ell}, \sigma^2\right)$ with respect to σ^2 and $\boldsymbol{\beta}$ can be solved explicitly. Finally, the Maximum Likelihood estimates of $\boldsymbol{\ell}$, σ^2 and $\boldsymbol{\beta}$ are:

$$\boldsymbol{\beta}_{ML}\left(\boldsymbol{\ell}\right) = \left(\boldsymbol{H}^T \boldsymbol{R}_{\boldsymbol{\ell}}^{-1} \boldsymbol{H}\right)^{-1} \boldsymbol{H}^T \boldsymbol{R}_{\boldsymbol{\ell}}^{-1} \boldsymbol{y}^{\text{obs}}, \tag{1.4.40}$$

$$\sigma^2_{ML}\left(\boldsymbol{\ell}\right) = \frac{1}{n}\left(\boldsymbol{y}^{\text{obs}} - \boldsymbol{H}\boldsymbol{\beta}_{ML}\right)^T \boldsymbol{R}_{\boldsymbol{\ell}}^{-1}\left(\boldsymbol{y}^{\text{obs}} - \boldsymbol{H}\boldsymbol{\beta}_{ML}\right), \tag{1.4.41}$$

$$\boldsymbol{\ell}_{ML} = \underset{\boldsymbol{\ell} \in \mathbb{R}^d}{\operatorname{argmin}} \ln\left|\sigma^2_{ML}\left(\boldsymbol{\ell}\right) \boldsymbol{R}_{\boldsymbol{\ell}}\right| \tag{1.4.42}$$

1.4.3.2 Restricted Maximum Likelihood Estimation

Restricted Maximum Likelihood Estimation (REML) enables to estimate the hyperparameters of the covariance function and the parameters $\boldsymbol{\beta}$ independently. This method is particularly appropriate if the prior distribution of $\boldsymbol{\beta}$ is not a uniform improper distribution. It is based on the left null space of matrix $\boldsymbol{H}$. This null space can be associated with a $((n-p)\times n)$-dimensional matrix $\boldsymbol{W}$, such that $\boldsymbol{W}\boldsymbol{H} = 0$. If one introduces $\boldsymbol{w}^{\text{obs}} = \boldsymbol{W}\boldsymbol{y}^{\text{obs}}$, one has:

$$\boldsymbol{w}^{\text{obs}} \sim \mathcal{N}\left(0, \sigma^2 \boldsymbol{W}\boldsymbol{R}_{\boldsymbol{\ell}}\boldsymbol{W}^T\right). \tag{1.4.43}$$

The Restricted Maximum Likelihood can thus be written:

$$\mathcal{L}^{REML}\left(\boldsymbol{\ell}, \sigma^2\right) = -\frac{1}{2}\ln\left|\sigma^2 \boldsymbol{W}\boldsymbol{R}_{\boldsymbol{\ell}}\boldsymbol{W}^T\right| - \frac{1}{2}\frac{1}{\sigma^2}\left(\boldsymbol{w}^{\text{obs}}\right)^T \left(\boldsymbol{W}\boldsymbol{R}_{\boldsymbol{\ell}}\boldsymbol{W}^T\right)^{-1} \boldsymbol{w}^{\text{obs}}, \tag{1.4.44}$$

and does not depend on the parameters $\boldsymbol{\beta}$.

1.4.3.3 Cross Validation Estimation

Following Dubrule [1983] and Bachoc [2013b], the correlation length of the covariance function can be estimated by minimizing the Leave One Out Mean Square error. The Leave One Out estimate of the correlation length $\boldsymbol{\ell}$ is given by:

$$\boldsymbol{\ell}_{LOO} = \underset{\boldsymbol{\ell}}{\operatorname{argmin}}\ MSE_{LOO}, \tag{1.4.45}$$

with

$$MSE_{LOO} = \sum_{i=1}^{n}\left[\left(\boldsymbol{y}^{\text{obs}}\right)_i - m^c_{-i,\boldsymbol{\ell}}\left(\boldsymbol{x}^{(i)}\right)\right]^2, \tag{1.4.46}$$

$m^c_{-i,\boldsymbol{\ell}}\left(\boldsymbol{x}^{(i)}\right) = \mathbb{E}\left[Y\left(\boldsymbol{x}^{(i)}\right) \mid \left(\boldsymbol{y}^{\text{obs}}\right)_{-i}, \boldsymbol{\ell}\right]$ and $\left(\boldsymbol{y}^{\text{obs}}\right)_{-i}$ denoting all the observations except the i-th observation.

The variance hyperparameter σ^2 can be estimated by setting the value of the Leave-One-Out prediction error to 1. The Leave-One-Out prediction error is defined by:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{\left(\left(\boldsymbol{y}^{\text{obs}}\right)_i - m^c_{-i,\boldsymbol{\ell}_{LOO}}\left(\boldsymbol{x}^{(i)}\right)\right)^2}{\sigma^2 K^c_{-i,\boldsymbol{\ell}_{LOO}}\left(\boldsymbol{x}^{(i)}\right)}, \tag{1.4.47}$$

with $\sigma^2 K^c_{-i,\boldsymbol{\ell}_{LOO}}\left(\boldsymbol{x}^{(i)}\right) = \mathbb{V}\left[Y\left(\boldsymbol{x}^{(i)}\right) \mid \left(\boldsymbol{y}^{\text{obs}}\right)_{-i}, \boldsymbol{\ell}_{LOO}, \sigma^2\right]$.
Thus, the prediction variance estimate is:

$$\sigma^2_{LOO} = \frac{1}{n}\sum_{i=1}^{n}\frac{\left(\left(\boldsymbol{y}^{\text{obs}}\right)_i - m^c_{-i,\boldsymbol{\ell}_{LOO}}\left(\boldsymbol{x}^{(i)}\right)\right)^2}{K^c_{-i,\boldsymbol{\ell}_{LOO}}\left(\boldsymbol{x}^{(i)}\right)}. \tag{1.4.48}$$

Moreover, the two criteria can be evaluated using matrix forms:

$$MSE_{LOO} = \frac{1}{n}\left(\boldsymbol{y}^{\text{obs}}\right)^T \tilde{\boldsymbol{R}}^-_{\boldsymbol{\ell}}\,\text{diag}\left(\tilde{\boldsymbol{R}}^-_{\boldsymbol{\ell}}\right)^{-2}\tilde{\boldsymbol{R}}^-_{\boldsymbol{\ell}}\boldsymbol{y}^{\text{obs}}, \tag{1.4.49}$$

and:

$$\sigma^2_{LOO} = \frac{1}{n}\left(\boldsymbol{y}^{\text{obs}}\right)^T \tilde{\boldsymbol{R}}^-_{\boldsymbol{\ell}}\,\text{diag}\left(\tilde{\boldsymbol{R}}^-_{\boldsymbol{\ell}}\right)^{-1}\tilde{\boldsymbol{R}}^-_{\boldsymbol{\ell}}\boldsymbol{y}^{\text{obs}}, \tag{1.4.50}$$

with $\tilde{\boldsymbol{R}}^-_{\boldsymbol{\ell}} = \boldsymbol{R}^{-1}_{\boldsymbol{\ell}} - \boldsymbol{R}^{-1}_{\boldsymbol{\ell}}\boldsymbol{H}\left(\boldsymbol{H}^T\boldsymbol{R}^{-1}_{\boldsymbol{\ell}}\boldsymbol{H}\right)^{-1}\boldsymbol{H}^T\boldsymbol{R}^{-1}_{\boldsymbol{\ell}}$ in the Universal Kriging framework, and $\tilde{\boldsymbol{R}}^-_{\boldsymbol{\ell}} = \boldsymbol{R}^{-1}_{\boldsymbol{\ell}}$ in the simple Kriging framework.

1.5 Design of experiments

From the previous sections it can be inferred that, by construction, the accuracy of the linear model, in the case of a linear regression, or of the Gaussian process, in the case of Kriging, depends on the choice of the observations. In the following section, we focus on the design of experiments, that is to say the choice of the observations of the code.

1.5.1 Space-filling designs

In this section we focus on space-filling designs. Such designs are adapted to the case of inputs with a uniform distribution on the unit hypercube $[0,1]^d$. Note that if the inputs have a non-uniform distribution, an isoprobabilistic transformation can be used to make them uniformly distributed over $[0,1]^d$.

1.5.1.1 LHS designs

Introduced by McKay et al. [1979], Latin Hypercube sampling enables to obtain a sample whose marginals are uniform. If one considers the unit hypercube $[0,1]^d$, a sample of n points is generated by first dividing each of the d axes of the input domain into n parts. Thus, the unit hypercube is divided into n^d parts and the n observations are drawn uniformly into a selection of n of these small hypercubes. As mentioned above, the n small hypercubes are chosen such that the projection onto each axis leads to exactly n different boxes. Figure 1.7 shows an example of a Latin Hypercube Design.

However, if the projections of a Latin Hypercube design on the marginals are uniformly distributed, the projections of higher dimension are not necessarily uniformly distributed. Two distance-based criteria [Johnson et al., 1990] can be used to characterize the space-filling properties of a design of experiments:

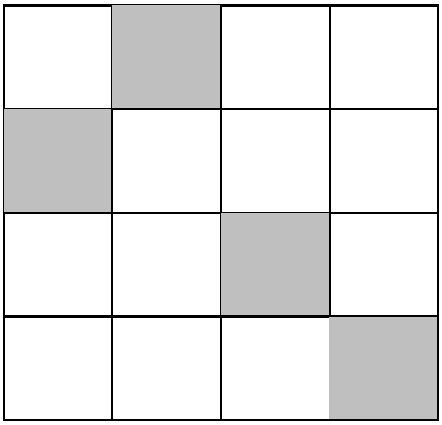


Figure 1.7: An example of Latin Hypercube Design. The observations are drawn in the grey cells.

- the maximin criterion maximizes the Euclidean distance between two points of the design:

$$\boldsymbol{X}^{\text{obs}}_{maximinLHS} = \underset{\boldsymbol{X}^{\text{obs}} \in \mathbb{X}^n}{\operatorname{argmax}} \min_{\substack{i \neq j \\ 1 \leq i,j \leq n}} \left\| \left(\boldsymbol{X}^{\text{obs}}\right)_i - \left(\boldsymbol{X}^{\text{obs}}\right)_j \right\|, \tag{1.5.1}$$

- the minimax criterion minimizes the distance between any points of $\mathbb{X}$ and the design:

$$\boldsymbol{X}^{\text{obs}}_{minimaxLHS} = \underset{\boldsymbol{X}^{\text{obs}} \in \mathbb{X}^n}{\operatorname{argmin}} \max_{\boldsymbol{x} \in \mathbb{X}} \max_{1 \leq i \leq n} \left\| \boldsymbol{x} - \left(\boldsymbol{X}^{\text{obs}}\right)_i \right\|. \tag{1.5.2}$$

These criteria can be used in order to sample LHS designs which have good space-filling properties.

1.5.1.2 Quasi-random designs

Low discrepancy sequences like Sobol sequences can also be utilized to ensure good space-filling properties of the design. The notion of discrepancy has been introduced by Niederreiter [1978] and is a measure of the divergence between a set of observations and the uniform distribution. If the definition set is the unit hypercube $[0,1]^d$, then the discrepancy is defined by:

$$\mathcal{D}\left(\boldsymbol{X}^{\text{obs}}\right) = \sup_{\boldsymbol{a},\boldsymbol{b} \in [0,1]^d, \boldsymbol{a} < \boldsymbol{b}} \left| \frac{\operatorname{card}\left(\left\{\boldsymbol{x} \in \boldsymbol{X}^{\text{obs}} | \boldsymbol{x} \in \prod_{i=1}^{d} [a_i, b_i)\right\}\right)}{n} - \prod_{i=1}^{d} (b_i - a_i) \right|, \tag{1.5.3}$$

with $\operatorname{card}(\Omega)$ denoting the number of elements of the finite set Ω.
Low-discrepancy sequences [Niederreiter, 1978] are also known as quasi-random designs. They are defined such that the discrepancy of the sequence tends to zero when the size of the sequence tends to infinity. The low-discrepancy sequences have a smaller discrepancy than a uniform Monte Carlo sample, thus covering better the unit hypercube.
The best-known low-discrepancy sequences are the Van der Corput [Van der Corput, 1935], Halton [Halton, 1964], Sobol [Sobol, 1967], Faure and Hammersley [Hammersley, 1964] sequences.

Space-filling designs can also be defined for the case of a non-hypercube domain (see Perrin

and Cannamela [2017] for example).

If there is no a priori information about the basis of functions in the linear regression case or about the covariance function in the Gaussian process regression case, then space-filling designs are very appropriate to acquire a knowledge of the computer code. Once some information is available, criterion-based designs can be used. The following section details criterion-based designs which are suited for linear regression and Gaussian process regression.

1.5.2 Criterion-based designs

In this section we focus on the optimal designs which can be used when some information about the model is available. The two first sections focus on the criteria which are suited for linear regression and Gaussian process regression. The third section presents the sequential designs, that is to say the enrichment of an initial design (which can be empty) according to a criterion.

1.5.2.1 Designs for linear regression

Elfving [1952] introduced optimal designs for linear regression, with criteria such as D-optimality. Since then, many other criteria, and algorithms of construction of the optimal designs have been proposed [Kiefer and Wolfowitz, 1959; Kiefer, 1961; Fedorov, 1972; Wu and Wynn, 1978; Cook and Nachtsheim, 1980; Fedorov and Hackl, 1997; Molchanov and Zuyev, 2002]. Such designs aim generally at minimizing or maximizing a criterion associated with the variance of the estimation of the regression coefficients $\boldsymbol{\beta}$.
According to Eq. (1.4.35), with a nugget covariance of variance σ^2, the covariance matrix of the posterior distribution of the parameters is given by:

$$\operatorname{cov}(\boldsymbol{\beta}) = \sigma^2 \left(\boldsymbol{h}\left(\boldsymbol{X}^{\text{obs}}\right) \boldsymbol{h}\left(\boldsymbol{X}^{\text{obs}}\right)^T \right)^{-1}, \tag{1.5.4}$$

where, by abuse of notation, $\boldsymbol{h}\left(\boldsymbol{X}^{\text{obs}}\right)$ is a $(p \times n)$-dimensional matrix defined by:

$$\boldsymbol{h}\left(\boldsymbol{X}^{\text{obs}}\right) = \left[\boldsymbol{h}\left(\boldsymbol{x}^{(1)}\right); \cdots; \boldsymbol{h}\left(\boldsymbol{x}^{(n)}\right) \right]. \tag{1.5.5}$$

Note that the inverse of the covariance matrix of the parameters is also known as the information matrix.
Several criterion-based designs can be used for linear regression:

- The D-optimal criterion aims at maximizing the determinant of the inverse of the covariance matrix:

$$\boldsymbol{X}_D^{\text{obs}} = \underset{\boldsymbol{X}^{\text{obs}} \in \mathbb{X}^n}{\operatorname{argmax}} \det \left(\boldsymbol{h}\left(\boldsymbol{X}^{\text{obs}}\right) \boldsymbol{h}\left(\boldsymbol{X}^{\text{obs}}\right)^T \right), \tag{1.5.6}$$

- The A-optimal criterion aims at minimizing the trace of the covariance matrix:

$$\boldsymbol{X}_A^{\text{obs}} = \underset{\boldsymbol{X}^{\text{obs}} \in \mathbb{X}^n}{\operatorname{argmin}} \operatorname{Tr} \left(\left(\boldsymbol{h}\left(\boldsymbol{X}^{\text{obs}}\right) \boldsymbol{h}\left(\boldsymbol{X}^{\text{obs}}\right)^T \right)^{-1} \right). \tag{1.5.7}$$

1.5.2.2 Designs for Gaussian process regression

In the case of the Gaussian process regression, the design can aim either at improving the estimation of the parameters $\boldsymbol{\beta}$ of the mean function or at improving the prediction accuracy of the posterior distribution of the Gaussian process.

In the first case, a D-optimal criterion can be used. In the Gaussian process regression framework, this criterion is defined as:

$$\boldsymbol{X}_D^{\text{obs}} = \underset{\boldsymbol{x}^{\text{obs}} \in \mathbb{X}^n}{\text{argmax}} \det \left(\boldsymbol{h}\left(\boldsymbol{X}^{\text{obs}}\right) C\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{X}^{\text{obs}}\right)^{-1} \boldsymbol{h}\left(\boldsymbol{X}^{\text{obs}}\right)^T \right). \tag{1.5.8}$$

If the aim is the improvement of the prediction accuracy, the following criterion, generally referred as I-optimal design, can be used:

$$\boldsymbol{X}_I^{\text{obs}} = \underset{\boldsymbol{x}^{\text{obs}} \in \mathbb{X}^n}{\text{argmin}} \int_{\mathbb{X}} \mathbb{V}\left[Y(\boldsymbol{x}) \,|\boldsymbol{y}^{\text{obs}}\right] d\mu_{\mathbb{X}}(\boldsymbol{x}). \tag{1.5.9}$$

Note that it can be inferred from Eqs. (1.4.19), (1.4.25) and (1.4.32) that $\mathbb{V}\left[Y(\boldsymbol{x}) \,|\boldsymbol{y}^{\text{obs}}\right]$ depends only on $\boldsymbol{X}^{\text{obs}}$ and the covariance function C. By abuse of notation, the previous criterion can be rewritten:

$$\boldsymbol{X}_I^{\text{obs}} = \underset{\boldsymbol{x}^{\text{obs}} \in \mathbb{X}^n}{\text{argmin}} \int_{\mathbb{X}} \mathbb{V}\left[Y(\boldsymbol{x}) \,|\boldsymbol{X}^{\text{obs}}\right] d\mu_{\mathbb{X}}(\boldsymbol{x}). \tag{1.5.10}$$

The integral $\int_{\mathbb{X}} \mathbb{V}\left[Y(\boldsymbol{x}) \,|\boldsymbol{X}^{\text{obs}}\right] d\mu_{\mathbb{X}}(\boldsymbol{x})$ is defined as the Integrated Mean Square Error (IMSE) [Sacks et al., 1989].
However, the choice of a criterion-based design may pose some difficulties:

- if a discrete search is performed, the number of possible combinations can be very high: $\binom{n}{\mathcal{N}}$, where $\mathcal{N}$ is the number of candidates of the search set.
- in the case of a Gaussian process, the covariance function can be unknown or not precisely known at the beginning.

In those cases, sequential designs can be used. In the case of Gaussian process regression an initial design drawn according to $\mu_{\mathbb{X}}$ can be used for the initial estimation of the covariance function hyperparameters. Then the hyperparameters of the covariance function can be re-estimated at each step of the sequential design.
The stochastic properties of the Gaussian process regression are useful for the definition of sequential designs. Sacks et al. [1989] proposed a sequential design based on the division of the input domain into boxes. The new point is added in the box with the largest contribution to the current IMSE.

Vazquez and Bect [2009] and Bect et al. [2012] proposed a Stepwise Uncertainty Reduction strategy [Geman and Jedynak, 1996] based on a sequential enrichment of the design which is adapted to the estimation of a probability of failure, using a Kriging metamodel and a Bayesian framework.
Such a Stepwise Uncertainty Reduction approach is based on the choice of a new observation point that improves the most a given criterion at the next step.
Bates et al. [1996], then Picheny et al. [2010] proposed a sequential design which is based on the integrated prediction variance (or Integrated Mean Square Error, IMSE) criterion. The associated criterion can be written in the form:

$$\boldsymbol{x}_{\text{new}} = \underset{\boldsymbol{x}^* \in \mathbb{X}}{\text{argmin}} \int_{\mathbb{X}} \mathbb{V}\left[Y(\boldsymbol{x}) \,|\boldsymbol{X}^{\text{obs}}, \boldsymbol{x}^*\right] d\mu_{\mathbb{X}}(\boldsymbol{x}), \tag{1.5.11}$$

where, by abuse of notation, $\mathbb{V}\left[Y(\boldsymbol{x}) \,|\boldsymbol{X}^{\text{obs}}, \boldsymbol{x}^*\right] = \mathbb{V}\left[Y(\boldsymbol{x}) \,|\boldsymbol{y}^{\text{obs}}, y(\boldsymbol{x}^*)\right]$. Such a notation can be used, because, for a given covariance function C, the conditioned variance $C^c(\boldsymbol{x}, \boldsymbol{x}')$

does not depend on the observations of the output (see Eqs. (1.4.25), (1.4.19) and (1.4.32) for further details).

The above-mentioned design criteria aim at improving the accuracy of the surrogate model, of the posterior distribution of the parameters or of the estimation of a probability of failure. They are all based on the minimization or maximization of a criterion associated with the variance of the estimator of the quantity of interest.

In the next section, we present the Efficient Global Optimization (EGO) algorithm. This is a widely used algorithm which adds to the design a new point which is in the most likely region of a minimum of the function y.

1.5.3 Gaussian processes for pointwise global optimization

Jones et al. [1998] proposed a sequential design aiming at finding the global minimum of an expensive to evaluate function (or computer code). The Efficient Global Optimization algorithm is based on a Gaussian process emulator of the expensive function and takes advantage of the stochastic property of the Gaussian predictor to determine which new point to add. The criterion is based on an Improvement function defined as:

$$I\left(\boldsymbol{x}\right)|\boldsymbol{y}^{\text{obs}} = \max\left(\min\left(\boldsymbol{y}^{\text{obs}}\right) - Y\left(\boldsymbol{x}\right)|\boldsymbol{y}^{\text{obs}}, 0\right). \tag{1.5.12}$$

The new observation point minimizes the Expected Improvement (EI):

$$\begin{aligned} EI\left(\boldsymbol{x}\right) = &\ \mathbb{E}\left[I\left(\boldsymbol{x}\right)|\boldsymbol{y}^{\text{obs}}\right] \\ = &\ \left(\min\left(\boldsymbol{y}^{\text{obs}}\right) - \mu^{c}\left(\boldsymbol{x}\right)\right)\Phi\left(\frac{\min\left(\boldsymbol{y}^{\text{obs}}\right) - \mu^{c}\left(\boldsymbol{x}\right)}{\sigma^{c}\left(\boldsymbol{x}\right)}\right) + \sigma^{c}\left(\boldsymbol{x}\right)\varphi\left(\frac{\min\left(\boldsymbol{y}^{\text{obs}}\right) - \mu^{c}\left(\boldsymbol{x}\right)}{\sigma^{c}\left(\boldsymbol{x}\right)}\right), \end{aligned} \tag{1.5.13}$$

with φ the standard Gaussian probability density function and Φ the standard Gaussian cumulative distribution function. The new observation point $\boldsymbol{x}_{\text{new}}$ is therefore chosen according to the following criterion:

$$\boldsymbol{x}_{\text{new}} = \underset{\boldsymbol{x}\in\mathbb{X}}{\text{argmax}}\ EI\left(\boldsymbol{x}\right). \tag{1.5.14}$$

EGO is a compromise between exploration and exploitation.

1.6 Sensitivity analysis

The sensitivity analysis aims at estimating the importance of the influence of the inputs of a code over the output of the code or over a quantity of interest associated with it. By abuse of notation, this quantity of interest will be denoted by y in the remainder of this section. The sensitivity analysis methods can be divided into two groups:

- the local sensitivity analysis studies the influence of small variations of the input parameters over a quantity of interest associated with the output of the code,
- the global sensitivity analysis quantifies the influence of the inputs over a quantity of interest associated with the output of the code by considering the variations of the inputs on the whole input domain.

The interested reader can refer to Saltelli et al. [2000] for further details on both groups. In what follows, we will focus on global sensitivity analysis.

Among the methods of global sensitivity analysis, two types of approaches can be distinguished:

- regression-based methods, which are based on the linear regression of the quantity of interest with respect to the inputs. It is worth noticing that such an approach is not adapted to the case of a significantly nonlinear mapping between the inputs and the quantity of interest [Saltelli and Sobol, 1995].
- variance-based methods, which are based on the decomposition of the variance of the quantity of interest with respect to the inputs. This decomposition of variance is also known as ANOVA (Analysis of Variance) [Fisher, 1925]. In particular, the Sobol indices [Sobol, 1993] belong to this category.

In the remainder of the section, we focus on the Sobol indices.
If the variance of the function of interest y is finite and the inputs $\boldsymbol{x}$ are independent, then the function of interest can be decomposed into first-order effects and interactions [Hoeffding, 1948] :

$$y(\boldsymbol{x}) = f_0 + \sum_{i=1}^{d} f_i(x_i) + \sum_{1 \leq i < j \leq d} f_{i,j}(x_i, x_j) + \cdots + f_{1,\ldots,d}(\boldsymbol{x}). \tag{1.6.1}$$

The unique decomposition of y of the form of Eq. (1.6.1) which verifies

$$\operatorname{cov}\left(f_{i_1,\ldots,i_s}(x_{i_1}, \ldots, x_{i_s}), f_{j_1,\ldots,j_t}(x_{j_1}, \ldots, x_{j_t})\right) = 0, \tag{1.6.2}$$

with $\{i_1, \ldots, i_s\} \in \mathbb{N}^s$, $1 \leq i_1 < \cdots < i_s \leq d$, $s \in \{1, \ldots, d\}$; $\{j_1, \ldots, j_t\} \in \mathbb{N}^t$, $1 \leq j_1 < \cdots < j_t \leq d$, $t \in \{1, \ldots, d\}$ and $\{i_1, \ldots, i_s\} \neq \{j_1, \ldots, j_t\}$, is defined by [Sobol, 1993] :

$$\begin{aligned} f_0 &= \mathbb{E}[y(\boldsymbol{x})] \\ f_i(x_i) &= \mathbb{E}[y(\boldsymbol{x}) | x_i] - f_0 \\ f_{i,j}(x_i, x_j) &= \mathbb{E}[y(\boldsymbol{x}) | x_i, x_j] - f_i(x_i) - f_j(x_j) - f_0 \\ &\ldots \end{aligned} \tag{1.6.3}$$

Given the uncorrelation of the terms of Eq. (1.6.3), the variance of $y(\boldsymbol{x})$ can thus be decomposed as follows:

$$\mathbb{V}[y(\boldsymbol{x})] = \sum_{i=1}^{d} \mathbb{V}[f_i(x_i)] + \sum_{1 \leq i < j \leq d} \mathbb{V}[f_{i,j}(x_i, x_j)] + \cdots + \mathbb{V}[f_{1,\ldots,d}(\boldsymbol{x})], \tag{1.6.4}$$

with the $f_i, f_{i,j} \ldots$ defined by Eq. (1.6.3).
The Sobol sensitivity index [Sobol, 1993] corresponding to the subset of input variables $\{x_{i_1}, \ldots, x_{i_s}\}$, is defined as:

$$S_{i_1,\ldots,i_s} = \frac{\mathbb{V}[\mathbb{E}[y(\boldsymbol{x}) | x_{i_1}, \ldots, x_{i_s}]]}{\mathbb{V}[y(\boldsymbol{x})]}. \tag{1.6.5}$$

It follows that:

$$1 = \sum_{i=1}^{d} S_i + \sum_{1 \leq i < j \leq d} S_{i,j} + \cdots + S_{1,\ldots,d}. \tag{1.6.6}$$

The first-order Sobol indices are often used to evaluate the individual effect of x_i on y. They are defined as:

$$S_i = \frac{\mathbb{V}\left[\mathbb{E}\left[y\left(\boldsymbol{x}\right)|x_i\right]\right]}{\mathbb{V}\left[y\left(\boldsymbol{x}\right)\right]}. \tag{1.6.7}$$

Moreover, a total sensitivity index [Homma and Saltelli, 1996] can be defined in order to evaluate the whole contribution of the variable x_i to the variance of the quantity of interest. These total sensitivity indices can be written:

$$T_i = \sum_{\{i_1,\ldots,i_s\}\subset\Omega_i} S_{i_1,\ldots,i_s}, \tag{1.6.8}$$

where Ω_i denotes the set of all the subsets of $\{1,\ldots,d\}$ containing i. These indices can also be written:

$$T_i = 1 - \frac{\mathbb{V}\left[\mathbb{E}\left[y\left(\boldsymbol{x}\right)|\boldsymbol{x}_{-i}\right]\right]}{\mathbb{V}\left[y\left(\boldsymbol{x}\right)\right]}, \tag{1.6.9}$$

where $\boldsymbol{x}_{-i}$ denotes the vector $\boldsymbol{x}$ except its i-th component .

In practice, the computation of the Sobol indices is performed using Monte Carlo methods [Sobol, 1993]. This computation requires the evaluation of the quantity of interest y at a large number of inputs points. If the computer code associated with this quantity of interest is computationally costly, then the use of a surrogate model of the code becomes necessary [Oakley and O'Hagan, 2004; Le Gratiet, 2013].

Chapter 2

Gaussian process regression of a code with a functional input or output

In this chapter, we review several existing methods for the Gaussian process regression of a computer code with a functional input and/or a functional output. By functional variable, we mean high dimensional vectorial variable. The functional variable is considered to be indexed by the time. The number of indices will be denoted by $N_t \in \mathbb{N}$ in the remainder of this document.

When aiming at performing a Gaussian process regression of a computer code with a functional input, a commonly used approach is to first reduce the dimension of the functional input thanks to a projection technique and then to construct a predictor which is a function of the projection coefficients.
When aiming at performing a Gaussian process regression of the functional output of a computer code with functional output and low dimensional vectorial inputs, two approaches exist. The first one is based on the projection of the output and the independent Gaussian process regression of the projected variables. The second one considers the whole functional output thanks to a tensorized structure of the covariance function of the Gaussian process modeling the code.

This chapter includes therefore two parts. The first one is devoted to the dimension reduction of a functional variable which can be the input or he output of a code. The second one focuses on the Gaussian process regression of the functional output of a code with scalar inputs.

2.1 Dimension reduction of a functional variable

When dealing with functional variables, dimension reduction techniques are often used. In this section, we present some existing methods for the dimension reduction of a functional variable. All the reviewed methods are based on a linear transformation of the functional variable.

The functional variable is denoted by $\boldsymbol{x}_t$. Moreover $\boldsymbol{x}_t \in \mathbb{X}_t \subset \mathbb{R}^{N_t}$, with $N_t \gg 1$, and is associated with the probability measure $\mu_{\mathbb{X}_t}$.
In the considered framework, a set of n observations of the N_t-dimensional vectorial variable $\boldsymbol{x}_t$ is available. The observations are independently drawn according to $\mu_{\mathbb{X}_t}$ and are centered and gathered in a $(N_t \times n)$-dimensional matrix $\boldsymbol{X}_t^{\text{obs}}$:

$$\boldsymbol{X}_t^{\text{obs}} = \left(\boldsymbol{x}_t^{(1)} - \overline{\boldsymbol{x}_t}^{\text{obs}}; \ldots; \boldsymbol{x}_t^{(n)} - \overline{\boldsymbol{x}_t}^{\text{obs}}\right), \tag{2.1.1}$$

where

$$\overline{\boldsymbol{x}_t}^{\text{obs}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_t^{(i)}. \tag{2.1.2}$$

Based on this set of observations and for a given dimension of the projection space m, the goal is to find the best m-dimensional set of N_t-dimensional vectors $\{\boldsymbol{f}_\alpha, \alpha \in \{1,\dots,m\}\}$, and the associated real-valued functions $\boldsymbol{x}_t \mapsto \beta_\alpha(\boldsymbol{x}_t)$, which are defined on $\mathbb{X}_t$. The formalism associated with the dimension reduction of the functional variable $\boldsymbol{x}_t$ is thus:

$$\boldsymbol{x}_t \approx \overline{\boldsymbol{x}_t}^{\text{obs}} + \sum_{\alpha=1}^{m} \boldsymbol{f}_\alpha \beta_\alpha(\boldsymbol{x}_t). \tag{2.1.3}$$

Besides, in the remainder of the section we will consider two types of dimension reduction methods. The first type considers the functional variable only. Such an approach is adapted to the dimension reduction of the functional output of a code, but can also be used for a functional input. The second type reduces the dimension of a functional input $\boldsymbol{x}_t$ of a code adequately with respect to the output of the code $\boldsymbol{y}_{\boldsymbol{x}_t}(\boldsymbol{x}_t)$.
In this work, we will consider only dimension reductions based on a linear transformation of $\boldsymbol{x}_t$ and the projection bases are always estimated from the observations.

Note that when considering a code with a functional input, a ridge approximation [Pinkus, 2015; Constantine et al., 2014] can be obtained thanks to the projection of the functional input. Such a ridge approximation can be written in the form:

$$\boldsymbol{y}_{\boldsymbol{x}_t}(\boldsymbol{x}_t) \approx \boldsymbol{g}_m\left(\boldsymbol{B}_m^{\text{obs}}\left(\boldsymbol{x}_t - \overline{\boldsymbol{x}_t}^{\text{obs}}\right)\right), \tag{2.1.4}$$

where $\boldsymbol{B}_m^{\text{obs}}$ is a $(m \times N_t)$-dimensional matrix, $\boldsymbol{g}_m$ is a function defined on $\mathbb{R}^m$, whose output has the same dimension as $\boldsymbol{y}_{\boldsymbol{x}_t}(\boldsymbol{x}_t)$, and $\overline{\boldsymbol{x}_t}^{\text{obs}}$ is defined by Eq. (2.1.2).

In the remainder of the section, we review some methods of dimension reduction of the two types mentioned above:

1. projection of the functional variable which is adapted to the functional variable only,
2. projection of the functional variable which is adapted to a dependent variable.

2.1.1 Methods of dimension reduction based on the functional variable only

When considering only the functional variable and no dependent variable, two types of projection methods can be distinguished. The first type is based on the projection of the functional variable on a basis of *a priori* known functions. The second one, the Principal Components Analysis, relies on the estimation of a projection basis from a set of available observations. These methods can be applied to the case of a functional input or a functional output.
The remainder of this section reviews these two types of approaches.

2.1.1.1 Methods based on the projection on a basis of existing functions

In the case of a basis of existing functions, the vectors $\boldsymbol{f}_\alpha$ of Eq. (2.1.3) are the discretized versions of functions of time.
The functions can be polynomials, wavelets [Meyer and Salinger, 1995], splines [Hastie et al., 2001], sine and cosine functions...

A set of functions of the basis of size m can be chosen thanks to one of the selection criteria described in Section 1.3. The subset $\mathbb{A}_m$ denotes the indices of the functions which have been kept after the selection procedure.
Moreover, the coefficients $\beta_\alpha(\boldsymbol{x}_t)$ of Eq. (2.1.3), $\alpha \in \mathbb{A}_m$ can be estimated by solving the following optimization problem:

$$\boldsymbol{\beta}(\boldsymbol{x}_t) = \underset{\boldsymbol{\beta}\in\mathbb{R}^m}{\operatorname{argmin}} \left\| \boldsymbol{x}_t - \overline{\boldsymbol{x}_t}^{\text{obs}} - \boldsymbol{F}_m\boldsymbol{\beta} \right\|^2, \tag{2.1.5}$$

where $\boldsymbol{F}_m$ is a $(N_t \times m)$-dimensional matrix gathering the $\boldsymbol{f}_\alpha, \alpha \in \mathbb{A}_m$ and $\boldsymbol{\beta}(\boldsymbol{x}_t)$ is a m-dimensional vector which gathers the $\beta_\alpha(\boldsymbol{x}_t)$, $\alpha \in \mathbb{A}_m$.
Consequently, $\boldsymbol{\beta}$ is an affine function of $\boldsymbol{x}_t$.
The Principal Component Analysis, introduced by Pearson [1901], is a widely used dimension reduction method. It is also known as the Karhunen-Loève expansion [Loève, 1955]. It is based on the eigendecomposition of the covariance matrix of the functional variable. The covariance matrix $\operatorname{cov}(\boldsymbol{x}_t)$ can be estimated from the set of observations of the functional variable $\boldsymbol{X}_t^{\text{obs}}$, where $\boldsymbol{X}_t^{\text{obs}}$ is defined by Eq. (2.1.1). This estimate of the covariance matrix is thus given by:

$$\boldsymbol{R}_{\boldsymbol{x}_t}^{\text{obs}} = \frac{1}{n-1}\boldsymbol{X}_t^{\text{obs}}\left(\boldsymbol{X}_t^{\text{obs}}\right)^T. \tag{2.1.6}$$

The projection basis is then defined by the eigenvectors of the covariance matrix $\boldsymbol{R}_{\boldsymbol{x}_t}^{\text{obs}}$. In other words, if the eigendecomposition of $\boldsymbol{R}_{\boldsymbol{x}_t}^{\text{obs}}$ is denoted by:

$$\boldsymbol{R}_{\boldsymbol{x}_t}^{\text{obs}} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T, \tag{2.1.7}$$

where the diagonal of $\boldsymbol{\Lambda}$ gathers the positive decreasing eigenvalues of $\boldsymbol{R}_{\boldsymbol{x}_t}^{\text{obs}}$, the m first projected variables are $\boldsymbol{V}_m^T\boldsymbol{x}_t$, with $\boldsymbol{V}_m$ gathering the m first columns of $\boldsymbol{V}$.
Note that the accuracy of the approximation $\boldsymbol{R}_{\boldsymbol{x}_t}^{\text{obs}}$ of the covariance matrix $\operatorname{cov}(\boldsymbol{x}_t)$ and thus the one of the projection basis, depend on the available observations of the functional variable.

It is also worth noticing that the Principal Component Analysis can also be used in combination with a ridge approximation. In such a case, the matrix $\boldsymbol{B}_m^{\text{obs}}$ of Eq. (2.1.4) is equal to $\boldsymbol{V}_m^T$.

2.1.2 Methods of dimension reduction of a functional variable which are adapted to a dependent variable

In this section, we focus on linear transformations aiming at reducing the dimension of the functional input of a code, such that the projected variable is adapted to the output of the code.
The two parts of this section present two methods of dimension reduction: the first one is based on Partial Least Squares [Wold, 1966] and the second one is based on the Active Subspaces method [Russi, 2010].

2.1.2.1 Partial Least Squares

Introduced by Wold [1966], Partial Least Squares aim at reducing the dimension of a functional variable $\boldsymbol{x}_t$ by taking into account a dependent variable which can be a scalar variable $y_{\boldsymbol{x}_t}$ or a functional variable $\boldsymbol{y}_{\boldsymbol{x}_t}$. In our framework, this dependent variable is the output of the code, whereas $\boldsymbol{x}_t$ is the input. The projection basis is determined from the covariance matrix between the functional variable $\boldsymbol{x}_t$ and the dependent variable. In this way, the functional input can be projected on a basis which is adapted to the output.

If a set of observations of the output of the code is available, and is denoted by:

$$\boldsymbol{Y}_{\boldsymbol{x}_t}^{\text{obs}} = \left(\boldsymbol{y}_{\boldsymbol{x}_t}\left(\boldsymbol{x}_t^{(1)}\right); \ldots; \boldsymbol{y}_{\boldsymbol{x}_t}\left(\boldsymbol{x}_t^{(n)}\right)\right), \tag{2.1.8}$$

where $\boldsymbol{Y}_{\boldsymbol{x}_t}^{\text{obs}}$ is a $(N_y \times n)$-dimensional matrix, N_y is the dimension of the output of $\boldsymbol{y}_{\boldsymbol{x}_t}$, then the covariance matrix $\text{cov}\left(\boldsymbol{x}_t, \boldsymbol{y}_{\boldsymbol{x}_t}\right)$ can be approximated by:

$$\boldsymbol{R}_{\boldsymbol{x}_t,\boldsymbol{y}_{\boldsymbol{x}_t}}^{\text{obs}} = \frac{1}{n-1}\boldsymbol{X}_t^{\text{obs}}\left(\boldsymbol{Y}_{\boldsymbol{x}_t}^{\text{obs}} - \overline{\boldsymbol{y}_{\boldsymbol{x}_t}}^{\text{obs}}\right)^T, \tag{2.1.9}$$

where

$$\overline{\boldsymbol{y}_{\boldsymbol{x}_t}}^{\text{obs}} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{y}_{\boldsymbol{x}_t}\left(\boldsymbol{x}_t^{(i)}\right). \tag{2.1.10}$$

Following Höskuldsson [1988], if the singular value decomposition of the covariance matrix between the functional variable and the dependent variable is denoted by:

$$\boldsymbol{R}_{\boldsymbol{x}_t,\boldsymbol{y}_{\boldsymbol{x}_t}}^{\text{obs}} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T, \tag{2.1.11}$$

where the diagonal of $\boldsymbol{D}$ gathers the positive singular values in decreasing order, then the m first projected variables of the functional input which are adapted to the output of the code are given by $\boldsymbol{U}_m^T\boldsymbol{x}_t$, with $\boldsymbol{U}_m$ gathering the m first columns of $\boldsymbol{U}$.

If we refer to the ridge approximation of Eq. (2.1.4), then, in the case of Partial Least Squares, $\boldsymbol{B}_m^{\text{obs}} = \boldsymbol{U}_m^T$.
Note that the accuracy of the estimation of the covariance matrix $\text{cov}\left(\boldsymbol{x}_t, \boldsymbol{y}_{\boldsymbol{x}_t}\right)$ and thus the one of the projection basis depend on the number of observations of the functional variable and its dependent variable.

Finally, Nanty et al. [2017] have studied ridge approximations based on the conditioned mean of a Gaussian process (also known as Kriging of Gaussian process regression, see section 1.4 for further details) indexed by the projection of the functional input. They have compared the prediction accuracy with a projection based on Principal Components Analysis or on Partial Least Squares. It is shown that, in many cases, the prediction accuracy of the ridge approximation is better with a dimension reduction based on Partial Least Squares than on Principal Components Analysis.

2.1.2.2 Active Subspaces

Introduced by Russi [2010], the Active Subspace refers to the projection of the functional input on an "Active Subspace", estimated from the observations of the derivatives of the output of the code with respect to the functional input $\boldsymbol{x}_t$. Using the formalism introduced by Constantine et al. [2014] for $N_y = 1$, if the set of n observations of the derivatives is denoted by:

$$\nabla\boldsymbol{y}_{\boldsymbol{x}_t}^{\text{obs}} = \left(\nabla y_{\boldsymbol{x}_t}\left(\boldsymbol{x}_t^{(1)}\right); \ldots; \nabla y_{\boldsymbol{x}_t}\left(\boldsymbol{x}_t^{(n)}\right)\right), \tag{2.1.12}$$

where $\nabla\boldsymbol{y}_{\boldsymbol{x}_t}^{\text{obs}}$ is a $(N_t \times n)$-dimensional matrix, then the projection basis is given by the eigenvectors of the $(N_t \times N_t)$-dimensional matrix $\nabla\boldsymbol{y}_{\boldsymbol{x}_t}^{\text{obs}}\left(\nabla\boldsymbol{y}_{\boldsymbol{x}_t}^{\text{obs}}\right)^T$. In other words, if one denotes by:

$$\nabla\boldsymbol{y}_{\boldsymbol{x}_t}^{\text{obs}}\left(\nabla\boldsymbol{y}_{\boldsymbol{x}_t}^{\text{obs}}\right)^T = \boldsymbol{W}\boldsymbol{\lambda}\boldsymbol{W}^T \tag{2.1.13}$$

the eigendecomposition of the matrix $\nabla \boldsymbol{y}_{\boldsymbol{x}_t}^{\text{obs}} \left(\nabla \boldsymbol{y}_{\boldsymbol{x}_t}^{\text{obs}}\right)^T$, where the diagonal of $\boldsymbol{\lambda}$ gathers the eigenvalues in decreasing order, then the m-dimensional vector of projection coefficients of the functional input $\boldsymbol{x}_t$ is given by $\boldsymbol{W}_m^T \boldsymbol{x}_t$ where $\boldsymbol{W}_m$ gathers the m first columns of the matrix $\boldsymbol{W}$.

If a ridge approximation is performed, the projection matrix $\boldsymbol{B}_m^{\text{obs}}$, defined by Eq. (2.1.4) is such that $\boldsymbol{B}_m^{\text{obs}} = \boldsymbol{W}_m$.

Zahm et al. compare the ridge approximation of $\boldsymbol{y}_{\boldsymbol{x}_t}$ for the case of $N_y > 1$ with a projection of the functional input based either on Principal Components Analysis (also called Karhunen-Loève expansion) or on Active Subspaces.
The Active Subspace, given by the projector P_m, is computed from the following matrix:

$$\frac{1}{n} \sum_{i=1}^{n} \nabla \boldsymbol{y}_{\boldsymbol{x}_t} \left(\boldsymbol{x}_t^{(i)}\right)^T \nabla \boldsymbol{y}_{\boldsymbol{x}_t} \left(\boldsymbol{x}_t^{(i)}\right) \tag{2.1.14}$$

with $\nabla \boldsymbol{y}_{\boldsymbol{x}_t} \left(\boldsymbol{x}_t^{(i)}\right)$ the $(N_t \times N_y)$-dimensional matrix of the derivatives at $\boldsymbol{x}_t^{(i)}$.
The studied ridge approximation of $\boldsymbol{y}_{\boldsymbol{x}_t}(\boldsymbol{x}_t)$ is of the form $\mathbb{E}\left[\boldsymbol{y}_{\boldsymbol{x}_t}(P_m \boldsymbol{x}_t + P_m^c \boldsymbol{X}_t) | \boldsymbol{x}_t\right]$, where P_m is a projector from $\mathbb{R}^{N_t}$ to $\mathbb{R}^m$, P_m^c its complement, and $\boldsymbol{X}_t$ is N_t-dimensional vector with probability measure $\mu_{\mathbb{X}_t}$. The authors conclude that Active Subspaces can yield more effective dimension reduction for the ridge approximation than Principal Components Analysis. They also observe that, if there is no low dimensional structure in the input-output map, then a dimension reduction based on the covariance of the input only (PCA) is more efficient.

This section has been devoted to the dimension reduction of a functional variable, which can be the input or the output of a computer code. In the next section, we focus on the Gaussian process regression of the functional output of a computer code. The notations used will be similar to those of Chapter 1.

2.2 Gaussian process prediction of a computer code with a functional output

In this section, we consider a computer code with low dimensional vectorial inputs and a functional output, that is to say, of the form $\boldsymbol{x} \mapsto \boldsymbol{y}(\boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{X} \subset \mathbb{R}^d$ and $\boldsymbol{y} \in \mathbb{R}^{N_t}$, $N_t \gg 1$. Moreover, $\mu_{\mathbb{X}}$ is a probability measure associated with $\boldsymbol{x}$.
The following sections detail the state of the art for the Gaussian process regression of $\boldsymbol{y}$ from a set of n observations of the input and the output of the code. These observations are denoted by:

$$\boldsymbol{X}^{\text{obs}} = \begin{pmatrix} \boldsymbol{x}^{(1)} \\ \vdots \\ \boldsymbol{x}^{(n)} \end{pmatrix}, \tag{2.2.1}$$

and

$$\boldsymbol{Y}^{\text{obs}} = \left(\boldsymbol{y}^{(1)} = \boldsymbol{y}\left(\boldsymbol{x}^{(1)}\right); \ldots; \boldsymbol{y}^{(n)} = \boldsymbol{y}\left(\boldsymbol{x}^{(n)}\right)\right), \tag{2.2.2}$$

where $\boldsymbol{X}^{\text{obs}}$ is a $(n \times d)$-dimensional matrix and $\boldsymbol{Y}^{\text{obs}}$ is a $(N_t \times n)$-dimensional matrix.

The first subsection of this section focuses on the Gaussian process prediction of a functional output thanks to the projection of this output on a basis. The second subsection is devoted to the Gaussian process regression of the whole functional output of the code.

2.2.1 Projection of the functional output on a basis

Bayarri et al. [2007] proposed to use a wavelet decomposition as a basis representation of the functional output. A thresholding procedure is performed in order to reduce the size of the set of the projection functions while obtaining an accurate projection. Then independent Gaussian predictors of each of the coefficients of the retained projection functions are constructed.

Higdon et al. [2008] proposed to build a Gaussian process emulator of the functional output of a code through a Principal Component Analysis of the functional output. First, a Principal Component Analysis of the functional output is performed. A number m of the projected variables is chosen such that these m components represent 99% of the total variance of the output. If $\overline{\boldsymbol{y}^{\text{obs}}}$ is the empirical mean of the observations of the output of the code:

$$\overline{\boldsymbol{y}^{\text{obs}}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{y}^{(i)}, \tag{2.2.3}$$

where the $\boldsymbol{y}^{(i)}$ are defined in Eq. (2.2.2), then the functional output of the code can be approximated by:

$$\boldsymbol{y}(\boldsymbol{x}) \approx \overline{\boldsymbol{y}^{\text{obs}}} + \boldsymbol{V}_m \boldsymbol{\omega}(\boldsymbol{x}), \tag{2.2.4}$$

with $\boldsymbol{V}_m$ a $(N_t \times m)$-dimensional matrix, whose columns are the m first eigenvectors of the empirical covariance matrix $\text{cov}\left(\boldsymbol{Y}^{\text{obs}}\right)$ and $\boldsymbol{\omega}$ a m-dimensional function giving the projection coefficients.
The observations of the function giving the projection coefficients are given by:

$$\boldsymbol{\omega}^{\text{obs}} = \boldsymbol{V}_m^T \left(\boldsymbol{Y}^{\text{obs}} - \overline{\boldsymbol{y}^{\text{obs}}}\right), \tag{2.2.5}$$

and $\boldsymbol{\omega}^{\text{obs}}$ is a $(m \times n)$-dimensional matrix.
Note that, by construction, $\boldsymbol{\omega}$ is expected to be a zero-mean m-dimensional vector.
The components of the function $\boldsymbol{\omega}$ are treated as being independent and a predictor of each component is constructed using the simple Kriging framework:

$$\omega_i(\cdot) \sim \text{GP}\left(0, C_{\omega_i}(\cdot,\cdot)\right), \tag{2.2.6}$$

with C_{ω_i} a covariance function, and $1 \leq i \leq m$.
The posterior predictor of the i-th component of function $\boldsymbol{\omega}$ is given by:

$$\omega_i(\cdot) \,|\boldsymbol{\omega}_i^{\text{obs}} \sim \text{GP}\left(\mu_{\omega_i}^c(\cdot), C_{\omega_i}^c(\cdot,\cdot)\right), \tag{2.2.7}$$

where $\boldsymbol{\omega}_i^{\text{obs}}$ corresponds to the i-th line of $\boldsymbol{\omega}^{\text{obs}}$, and:

$$\mu_{\omega_i}^c(\boldsymbol{x}) = C_{\omega_i}^c\left(\boldsymbol{x}, \boldsymbol{x}^{\text{obs}}\right) C_{\omega_i}^c\left(\boldsymbol{x}^{\text{obs}}, \boldsymbol{X}^{\text{obs}}\right)^{-1} \boldsymbol{\omega}_i^{\text{obs}}, \tag{2.2.8}$$

and:

$$C_{\omega_i}^c\left(\boldsymbol{x}, \boldsymbol{x}'\right) = C_{\omega_i}\left(\boldsymbol{x}, \boldsymbol{x}'\right) - C_{\omega_i}\left(\boldsymbol{x}, \boldsymbol{X}^{\text{obs}}\right) C_{\omega_i}\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{X}^{\text{obs}}\right)^{-1} C_{\omega_i}\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{x}'\right), \tag{2.2.9}$$

where $C_{\omega_i}\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{X}^{\text{obs}}\right)$ is a $(n \times n)$-dimensional matrix such that

$$C_{\omega_i}\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{X}^{\text{obs}}\right)_{kl} = C_{\omega_i}\left(\boldsymbol{x}^{(k)}, \boldsymbol{x}^{(l)}\right), \tag{2.2.10}$$

and $C_{\omega_i}\left(\boldsymbol{x}, \boldsymbol{X}^{\text{obs}}\right)$ is a n-dimensional vector such that:

$$C_{\omega_i}\left(\boldsymbol{x}, \boldsymbol{X}^{\text{obs}}\right)_k = C_{\omega_i}\left(\boldsymbol{x}, \boldsymbol{x}^{(k)}\right). \tag{2.2.11}$$

The multivariate predictor of $\boldsymbol{\omega}$ is therefore defined by:

$$\boldsymbol{\omega}(\cdot)\,|\boldsymbol{\omega}^{\text{obs}} \sim \text{GP}\left(\boldsymbol{\mu}_{\omega}^{c}(\cdot), \boldsymbol{C}_{\omega}^{c}(\cdot,\cdot)\right), \tag{2.2.12}$$

where:

$$\left(\boldsymbol{\mu}_{\omega}^{c}(\boldsymbol{x})\right)_i = \mu_{\omega_i}^{c}(\boldsymbol{x}) \tag{2.2.13}$$

and:

$$\left(\boldsymbol{C}_{\omega}^{c}\left(\boldsymbol{x},\boldsymbol{x}'\right)\right)_{ij} = C_{\omega_i}^{c}\left(\boldsymbol{x},\boldsymbol{x}'\right)\delta_{i=j}. \tag{2.2.14}$$

Finally, a predictor of $\boldsymbol{y}$ is given by:

$$\boldsymbol{y}(\cdot)\,|\boldsymbol{Y}^{\text{obs}} \sim \text{GP}\left(\overline{\boldsymbol{y}^{\text{obs}}} + \boldsymbol{V}_m\boldsymbol{\mu}_{\omega}^{c}(\cdot), \boldsymbol{V}_m\boldsymbol{C}_{\omega}^{c}(\cdot,\cdot)\,\boldsymbol{V}_m^{T}\right). \tag{2.2.15}$$

Perrin [2018] mentions that if the projection basis is estimated from a small set of observations, its estimation may be not very accurate. The accuracy of the prediction of the functional output using the method described above can thus suffer from this lack of accuracy of the projection basis.

2.2.2 Gaussian process regression of the whole functional output

Another possible approach for the Gaussian process regression of a functional output is to choose an appropriate structure of the covariance function of the Gaussian process. Such an approach enables to emulate the whole functional output of a code.

Williams et al. [2006] proposed to treat the index of the functional input as one of the inputs of the model. The covariance function of the Gaussian process depends on the inputs of the code and on the index of the functional output. The output can therefore be treated as a univariate output, indexed by an index input. A power exponential covariance function is used, such that the covariance function has a tensorized structure between the index (time) and the other inputs.

Rougier [2008] and Conti et al. [2009] have used a tensorized structure for the mean and covariance functions of the process. In this framework, the functional output of the code $\boldsymbol{y}$ can be seen as a Gaussian process $\boldsymbol{Y}$ with the following properties:

$$\boldsymbol{Y}(\cdot)\,|\boldsymbol{M},\boldsymbol{R}_t,C \sim \text{GP}\left(\boldsymbol{M}\boldsymbol{h}(\cdot), \boldsymbol{R}_t \otimes C(\cdot,\cdot)\right), \tag{2.2.16}$$

with $\boldsymbol{M}$ a $(N_t \times p)$-dimensional matrix, $\boldsymbol{h}$ a vector of p basis functions, $\boldsymbol{R}_t$ a $(N_t \times N_t)$-dimensional covariance matrix and C a covariance function, and $\otimes$ denoting the Kronecker product.
In this framework, if $\boldsymbol{M}$ has a uninformative prior distribution given by the uniform distribution on the space of the real-valued $(N_t \times p)$-dimensional matrices, then the distribution of $\boldsymbol{M}$ given the observations is Gaussian, with the following mean:

$$\begin{aligned}\widehat{\boldsymbol{M}} &= \mathbb{E}\left[\boldsymbol{M}|\boldsymbol{y}^{\text{obs}},\boldsymbol{R}_t,C\right] \\ &= \mathbb{E}\left[\boldsymbol{M}|\boldsymbol{y}^{\text{obs}},C\right] \\ &= \boldsymbol{y}^{\text{obs}}\left(\boldsymbol{R}^{\text{obs}}\right)^{-1}\left(\boldsymbol{H}^{\text{obs}}\right)^{T}\left(\boldsymbol{H}^{\text{obs}}\left(\boldsymbol{R}^{\text{obs}}\right)^{-1}\left(\boldsymbol{H}^{\text{obs}}\right)^{T}\right)^{-1},\end{aligned} \tag{2.2.17}$$

where $\boldsymbol{R}^{\text{obs}}$ is a $(n \times n)$-dimensional matrix such that:

$$\left(\boldsymbol{R}^{\text{obs}}\right)_{kl} = C\left(\boldsymbol{x}^{(k)},\boldsymbol{x}^{(l)}\right), \tag{2.2.18}$$

and $\boldsymbol{H}^{\text{obs}}$ is a $(p \times n)$-dimensional matrix whose j-th column is given by $\boldsymbol{h}\left(\boldsymbol{x}^{(j)}\right)$.
From Eq. (2.2.16), it can be inferred that:

$$\boldsymbol{Y}^{\text{obs}}|\boldsymbol{M}, \boldsymbol{R}_t, C \sim \mathcal{N}\left(\boldsymbol{M}\boldsymbol{H}^{\text{obs}}, \boldsymbol{R}_t \otimes \boldsymbol{R}^{\text{obs}}\right). \tag{2.2.19}$$

Therefore, the matrix $\boldsymbol{R}_t$ can be estimated by maximizing the likelihood of the observations, as proposed in Perrin [2018]:

$$\widehat{\boldsymbol{R}}_t = \frac{1}{n}\left(\boldsymbol{Y}^{\text{obs}} - \widehat{\boldsymbol{M}}\boldsymbol{H}^{\text{obs}}\right)\left(\boldsymbol{R}^{\text{obs}}\right)^{-1}\left(\boldsymbol{Y}^{\text{obs}} - \widehat{\boldsymbol{M}}\boldsymbol{H}^{\text{obs}}\right)^{T}. \tag{2.2.20}$$

Finally, in the Universal Kriging framework, with an improper uniform prior for $\boldsymbol{M}$, the conditioned distribution of $\boldsymbol{Y}$ is given by:

$$\boldsymbol{Y}^{c}(\cdot) := \boldsymbol{Y}(\cdot)\,|\boldsymbol{Y}^{\text{obs}}, C \sim \text{GP}\left(\boldsymbol{\mu}^{c}(\cdot), \widehat{\boldsymbol{R}}_t \otimes C^{c}(\cdot,\cdot)\right), \tag{2.2.21}$$

where:

$$\begin{aligned}
\boldsymbol{\mu}^{c}(\boldsymbol{x}) = &\ \widehat{\boldsymbol{M}}\boldsymbol{h}(\boldsymbol{x}) + \left[\boldsymbol{Y}^{\text{obs}} - \widehat{\boldsymbol{M}}\boldsymbol{H}^{\text{obs}}\right]\left(\boldsymbol{R}^{\text{obs}}\right)^{-1} C\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{x}\right), \\
C^{c}\left(\boldsymbol{x}, \boldsymbol{x}'\right) = &\ C\left(\boldsymbol{x}, \boldsymbol{x}'\right) - C\left(\boldsymbol{x}, \boldsymbol{X}^{\text{obs}}\right)\left(\boldsymbol{R}^{\text{obs}}\right)^{-1} C\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{x}'\right) \\
&+\boldsymbol{u}(\boldsymbol{x})^{T}\left(\boldsymbol{H}^{\text{obs}}\left(\boldsymbol{R}^{\text{obs}}\right)^{-1}\left(\boldsymbol{H}^{\text{obs}}\right)^{T}\right)^{-1}\boldsymbol{u}\left(\boldsymbol{x}'\right), \\
\boldsymbol{u}(\boldsymbol{x}) = &\ \boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{H}^{\text{obs}}\left(\boldsymbol{R}^{\text{obs}}\right)^{-1} C\left(\boldsymbol{X}^{\text{obs}}, \boldsymbol{x}\right),
\end{aligned} \tag{2.2.22}$$

and $C\left(\boldsymbol{x}, \boldsymbol{X}^{\text{obs}}\right)$ is a n-dimensional vector, such that:

$$C\left(\boldsymbol{x}, \boldsymbol{X}^{\text{obs}}\right)_k = C\left(\boldsymbol{x}, \boldsymbol{x}^{(k)}\right). \tag{2.2.23}$$

Part II

Contributions

Chapter 3

Nested polynomial trends for the improvement of Gaussian predictors

In this chapter, we focus on the case of two nested codes with scalar outputs. Moreover, there are no observations of the intermediary variable. We therefore consider the following system:

$$\begin{array}{ccccc} & & \boldsymbol{x}_2 & \searrow & \\ & & & & y_{\text{nest}}(\boldsymbol{x}_{\text{nest}}) := y_2(y_1(\boldsymbol{x}_1), \boldsymbol{x}_2), \\ \boldsymbol{x}_1 & \rightarrow & y_1(\boldsymbol{x}_1) & \nearrow & \end{array} \tag{3.0.1}$$

where $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ and $\boldsymbol{x}_{\text{nest}}$ are low dimensional vectors and y_1, y_2 and y_{nest} are scalars. This system becomes therefore:

$$\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2) \ \rightarrow \ y(\boldsymbol{x}) := y_2(y_1(\boldsymbol{x}_1), \boldsymbol{x}_2). \tag{3.0.2}$$

The work presented in this chapter has been published in [Perrin et al., 2017]. The framework of Gaussian process regression is considered (see Chapter 1 for further details). An innovative parametrization of the mean function of the Gaussian process, based on the composition of two polynomials, is proposed.

3.1 Introduction

The numerical cost of many codes to simulate complex physical systems is very high. In order to perform sensitivity analyses, uncertainty quantification or reliability studies, these computer models have therefore to be replaced by surrogate models, that is to say by fast and inexpensive mathematical functions. Within the computational science community, when the maximal available information is a finite set of code evaluations, the most widely used surrogate models are the generalized polynomial chaos expansion (PCE) [Ghanem and Spanos, 1990, 2003; Soize and Ghanem, 2004; Das et al., 2009; Le Maître and Knio, 2010; Arnst et al., 2010; Perrin et al., 2012] and the Gaussian process regression (GPR), or Kriging (see Sacks et al. [1989]; Oakley and O'Hagan [2002]; Rasmussen and Williams [2006]).
On the one hand, the main idea of PCE is to expand the code output, which is denoted by y in the following, onto an appropriate basis made of orthonormal multivariate polynomials, which are related to the distribution of the code input variables. As the number of unknown expansion coefficients usually grows exponentially with the number of input parameters, the relevance of these approaches strongly depends on their ability to select the most relevant basis functions. To this end, several penalization techniques, such as the ℓ_1-minimization [Tibshirani, 1989; Jakeman et al., 2015] and the least Angle Regression (LAR) methods [Hastie et al., 2002; Efron et al., 2004; Blatman and Sudret, 2011], have been introduced to select

polynomial basis sets that lead to more accurate PCE than would have been obtained if the basis is *a priori* fixed. Taking advantage of the tensor-product structure of the multivariate polynomial basis, separated representations, such as low-rank approximations [Nouy, 2010; Konakli and Sudret, 2016], have alternatively been proposed to develop surrogate models with polynomial functions in highly-compressed formats.

On the other hand, the GPR is based on the assumption that the code output is a particular realization of a Gaussian stochastic process, Y. This hypothesis, which was first introduced in time series analysis [Parzen, 1962] and in optimization [Kushner, 1964], is widely used as it allows dealing with the conditional probability and expectation, while leading to very interesting results in terms of computer code prediction. Hence, contrary to the PCE, the GPR is not associated with an *a priori* projection basis, but requires the introduction of the mean and the covariance functions of Y. In practice, we observe that the role of the mean function of Y on the prediction decreases when the number of code evaluations increases. This explains that in applications where many code evaluations are available, good GPR-based surrogate models can be obtained using constant or linear trends for the mean function. On the contrary, when the number of code evaluations is small compared to the complexity of y, it can be very useful to optimize it. In that case, searching the mean function of Y as a well-chosen sum of polynomial functions can indeed strongly improve the relevance of the associated GPR. In particular, the authors refer to [Joseph et al., 2008] and [Kersaudy et al., 2015] for an illustration of the interest of using variable selection techniques to optimize this polynomial representation of the mean function of Y.

Following these works, the idea of this part is to propose an alternative parametrization of the mean function of Y, which is particularly adapted to the case when the number of code evaluations is small compared to the complexity of y. Instead of searching sparse polynomial approximations, we look for high dimensional polynomial approximations that are characterized by a small number of parameters. In other words, if we want to model a complex code response with a very limited number of code evaluations, we believe that it can be more efficient to use complex but approximated models than simple but fully optimized models. We thus propose to consider the composition of two polynomials for the mean function of Y. Indeed, the composition of two polynomial functions is still a polynomial function, but of much higher order. In particular, such a formalism can be used to model separately a transformation of each code input and the dependence structure between them.

The main difficulty concerning this specific representation is the identification of the parameters of the two combined polynomials. Indeed, by composing two polynomial functions that are linear with respect to their parameters, we get a strongly non-linear representation, which is likely to be very sensitive to small changes in the parameters' values. In addition, distinct values for these parameters can lead to the same nested representation, which does not help for the identification. To avoid such redundancies, minimal nested parametrizations are introduced, and we show to what extent integrating this nested structure in the Gaussian process formalism can increase the robustness of the results, make easier the error control, and limit as much as possible over-fitting.

The outline of this chapter is as follows. First, Section 3.2 presents the theoretical framework for the definition of a Gaussian-process regression with a linear polynomial trend. Then, the nested polynomial trends we propose are detailed in Section 3.3. At last, the efficiency of the method is illustrated on a series of analytic examples in Section 3.4.

3.2 Gaussian process predictors

3.2.1 General framework

For $d \geq 1$, let $L^2(\mathbb{X}, \mathbb{R})$ be the space of square integrable functions on any compact subset $\mathbb{X}$ of $\mathbb{R}^d$, with values in $\mathbb{R}$, equipped with the inner product $(\cdot, \cdot)_{\mathbb{X}}$, and the associated norm $\|\cdot\|_{\mathbb{X}}$, such that for all u and v in $L^2(\mathbb{X}, \mathbb{R})$,

$$(u, v)_{\mathbb{X}} := \int_{\mathbb{X}} u(\boldsymbol{x}) v(\boldsymbol{x}) d\boldsymbol{x}, \quad \|u\|_{\mathbb{X}}^2 := (u, u)_{\mathbb{X}}\,. \tag{3.2.1}$$

If $\mathbb{X}$ is not compact, it is possible to introduce a weighted L_2 space.
Let $\mathcal{S}$ be a physical system, whose response depends on a d-dimensional input vector $\boldsymbol{x} = (x_1, \ldots, x_d)$, and whose performance can be evaluated from the computation of a quantity of interest, $y(\boldsymbol{x})$. Function y is a deterministic mapping that is assumed to be an element of $L^2(\mathbb{X}, \mathbb{R})$. In this chapter, we suppose that the maximal available information about y is a set of n code evaluations at the points $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\}$ in $\mathbb{X}$. Given this information, we are interested in the identification of the *best* predictor $\widehat{y}$ of y.
In that context, the Gaussian process regression (GPR), or Kriging, plays a major role [Sacks et al., 1989; Oakley and O'Hagan, 2002; Santner et al., 2003; Rasmussen and Williams, 2006]. It is indeed able to provide a prediction of $y(\boldsymbol{x})$, which is optimal in the class of the linear predictors of y, and whose precision can be *a posteriori* quantified. Such a method considers function y as a sample path of a real-valued Gaussian stochastic process Y. Let μ and C be respectively the mean and the covariance functions of Y:

$$Y(\cdot) \;\sim\; \mathrm{GP}\left(\mu(\cdot), C(\cdot, \cdot)\right). \tag{3.2.2}$$

Besides, a set of observations of y is available. These observations are gathered in a n-dimensional vector:

$$\boldsymbol{y}^{\mathrm{obs}} = \left(y^{(1)} = y(\boldsymbol{x}^{(1)}), \ldots, y^{(n)} = y(\boldsymbol{x}^{(n)})\right), \tag{3.2.3}$$

such that $\mathbb{P}(\,\cdot \mid \boldsymbol{y}^{\mathrm{obs}})$ and $\mathbb{E}\left[\,\cdot \mid \boldsymbol{y}^{\mathrm{obs}}\right]$ denote the conditional probability and conditional mathematical expectation respectively.

Therefore, gathering in the vector $\boldsymbol{\mu}$ and in the matrix $\boldsymbol{R}$ the evaluations of μ and C at the available points, such that:

$$\begin{cases} \boldsymbol{\mu} := \left(\mu(\boldsymbol{x}^{(1)}), \ldots, \mu(\boldsymbol{x}^{(n)})\right), \\ \boldsymbol{R}_{ij} := C(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}), \quad 1 \leq i, j \leq n, \end{cases} \tag{3.2.4}$$

it can be shown [O'Hagan, 1978] that if matrix $\boldsymbol{R}$ is invertible, then:

$$Y(\cdot) \mid \mu, C, \boldsymbol{y}^{\mathrm{obs}} \;\sim\; \mathrm{GP}\left(\mu^c(\cdot), C^c(\cdot, \cdot)\right), \tag{3.2.5}$$

where, for all $\boldsymbol{x}, \boldsymbol{x}'$ in $\mathbb{X}$:

$$\begin{cases} \mu^c(\boldsymbol{x}) := \mu(\boldsymbol{x}) + \boldsymbol{r}(\boldsymbol{x})^T \boldsymbol{R}^{-1} \left(\boldsymbol{y} - \boldsymbol{\mu}\right), \\ C^c(\boldsymbol{x}, \boldsymbol{x}') := C(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{r}(\boldsymbol{x})^T \boldsymbol{R}^{-1} \boldsymbol{r}(\boldsymbol{x}'), \\ \boldsymbol{r}(\boldsymbol{x}) := \left(C(\boldsymbol{x}, \boldsymbol{x}^{(1)}), \ldots, C(\boldsymbol{x}, \boldsymbol{x}^{(n)})\right). \end{cases} \tag{3.2.6}$$

Under this formalism, also known as simple Kriging (see Section 1.4), the best prediction of y in an unobserved point $\boldsymbol{x}$ is given by the mean value of $(Y(\boldsymbol{x}) \mid \boldsymbol{y}^{\mathrm{obs}})$, $\mu^c(\boldsymbol{x})$, whereas $C^c(\boldsymbol{x}, \boldsymbol{x})$ quantifies the trust we can put in that prediction.

In practice, it appears that $\boldsymbol{R}$ may not be invertible due to numerical reasons. This can generally be overcome by adding a small nugget to the covariance matrix and optimizing with respect to it too (see [Gramacy and Lee, 2012]).

3.2.2 Choice of the covariance function

Without information about the regularity of y, function C is generally chosen in general parametric families. In this chapter, function C is supposed to be an element of the Matern-5/2 class, such that for all $\boldsymbol{x}, \boldsymbol{x}'$ in $\mathbb{X}$:

$$C(\boldsymbol{x}, \boldsymbol{x}') := \sigma^2 \prod_{i=1}^{d} (1 + \sqrt{5}h_i + 5h_i^2/3) \exp(-\sqrt{5}h_i), \quad h_i = |x_i - x_i'|/\ell_i. \tag{3.2.7}$$

Hence, covariance function C is characterized by a vector of hyper-parameters, $\boldsymbol{\Theta} := (\sigma, \ell_1, \ldots, \ell_d)$, whose values also have to be conditioned by $\boldsymbol{y}^{\text{obs}}$. More details about other usual parametric expressions for C can be found in Santner et al. [2003]. A *full Bayesian* approach would then require the introduction of a prior distribution for this vector, and the use of sampling techniques (such as Monte Carlo Markov Chains [Rubinstein and Kroese, 2008]) to approximate the posterior distribution of $(Y \mid \boldsymbol{y}^{\text{obs}})$ [Handcock and Stein, 1993; Kennedy and O'Hagan, 2001; Bilionis et al., 2013]. In this chapter, we will adopt an alternative approach, which consists in conditioning all the results by the maximum likelihood estimate of the covariance parameters. This method, which is generally called *plug-in* approach, has been used in many papers for the definition of Gaussian process-based predictors, as it presents a good compromise between complexity, efficiency, and errors control [Bichon et al., 2008; Bect et al., 2012]. In that case, explicit formula can be derived to evaluate the relevance of the GPR-based metamodel from a cross validation procedure [Dubrule, 1983].

3.2.3 Choice of the mean function

In the same way as for the covariance function, the mean function of Y is supposed to be parametrized by a p-dimensional vector $\boldsymbol{\beta}$. In the general case, the computation of $\mathbb{E}\left[Y(\boldsymbol{x}) \mid \boldsymbol{y}^{\text{obs}}\right]$ is not direct, but if:

- covariance function C is known,
- μ is linear with respect to $\boldsymbol{\beta}$, that is to say it exists a p-dimensional vector-valued function $\boldsymbol{h}$ such that $\mu(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x})^T\boldsymbol{\beta}$,
- $\boldsymbol{\beta}$ is uniformly distributed on $\mathbb{R}^p$ (improper prior distribution),

then a Universal Kriging predictor can be defined (see Section 1.4 for further details):

$$Y(\cdot) \mid C, \boldsymbol{y}^{\text{obs}} \sim \text{GP}\left(\mu^c(\cdot), C^c(\cdot,\cdot)\right), \tag{3.2.8}$$

$$\begin{cases} \mu^c(\boldsymbol{x}) := \boldsymbol{h}(\boldsymbol{x})^T\widehat{\boldsymbol{\beta}} + \boldsymbol{r}(\boldsymbol{x})^T\boldsymbol{R}^{-1}\left(\boldsymbol{y}^{\text{obs}} - \boldsymbol{H}\widehat{\boldsymbol{\beta}}\right), \\ C^c(\boldsymbol{x}, \boldsymbol{x}') := C(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{r}(\boldsymbol{x})^T\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{x}') + \boldsymbol{u}(\boldsymbol{x})^T(\boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{H})^{-1}\boldsymbol{u}(\boldsymbol{x}'), \\ \widehat{\boldsymbol{\beta}} := (\boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{y}^{\text{obs}}, \\ \boldsymbol{u}(\boldsymbol{x}) := \boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{x}) - \boldsymbol{h}(\boldsymbol{x}), \\ \boldsymbol{H}_{ij} := f_j(\boldsymbol{x}^{(i)}), \quad 1 \leq j \leq p, \ 1 \leq i \leq n, \end{cases} \tag{3.2.9}$$

where the term $\boldsymbol{u}(\boldsymbol{x})^T(\boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{H})^{-1}\boldsymbol{u}(\boldsymbol{x}')$ can be interpreted as the prediction uncertainty that is due to the estimation of $\boldsymbol{\beta}$. Under these assumptions, the best prediction of $y(\boldsymbol{x})$ is now given by μ^c. The last thing that can be done to minimize $\|y-\mu^c\|_{\mathbb{X}}$ is working on the choice of $\boldsymbol{h}$.

Without information about y, polynomials are generally chosen for $\boldsymbol{h}$. Indeed, the set $\{m_{\boldsymbol{\alpha}},\ \boldsymbol{\alpha}\in\mathbb{N}^d\}$, with

$$m_{\boldsymbol{\alpha}}(\boldsymbol{x}) := x_1^{\alpha_1}\times\cdots\times x_d^{\alpha_d},\quad \boldsymbol{x}\in\mathbb{X}, \tag{3.2.10}$$

defines a basis of $L^2(\mathbb{X},\mathbb{R})$. For a given value of p, characterizing $\boldsymbol{h}$ amounts at identifying the best p-dimensional subset of $\{m_{\boldsymbol{\alpha}},\ \boldsymbol{\alpha}\in\mathbb{N}^d\}$ to minimize $\|y-\mu^c\|_{\mathbb{X}}$.

In practice, this optimization problem over a very vast space is replaced by an optimization over a finite dimensional subset of $\{m_{\boldsymbol{\alpha}},\ \boldsymbol{\alpha}\in\mathbb{N}^d\}$. Different truncation schemes have been proposed to choose such a relevant subset, which are mostly based on the assumption that the most influential elements of $\{m_{\boldsymbol{\alpha}},\ \boldsymbol{\alpha}\in\mathbb{N}^d\}$ correspond to the elements of lowest total polynomial order. Denoting by r the maximal polynomial order of the projection basis, we can introduce:

$$\mathcal{P}(r,d) := \{m_{\boldsymbol{\alpha}}\mid \boldsymbol{\alpha}\in\mathbb{N}^d,\ \sum_{i=1}^{d}|\alpha_i|\leq r\}. \tag{3.2.11}$$

By construction, it can be noticed that the cardinal $\mathcal{C}(r,d)$ of $\mathcal{P}(r,d)$ increases exponentially with respect to r and d:

$$\mathcal{C}(r,d) = (d+r)!/(d!\times r!). \tag{3.2.12}$$

For $p\leq\mathcal{C}(r,d)$, vector $\boldsymbol{h}$ can finally be searched using a penalization technique, such as the Least Angle Regression (LAR) method [Hastie et al., 2002; Efron et al., 2004; Blatman and Sudret, 2011], which allows disregarding insignificant terms. Such an approach will be referred as "LAR+UK" approach in the following.

3.3 Nested polynomial trends for Gaussian process predictors

As presented in Introduction, we are interested in identifying the best predictor of y in any unobserved point $\boldsymbol{x}$ in $\mathbb{X}$, when the maximal information is a fixed number of code evaluations. Instead of considering sparse representations for the parametrization of the mean function in the GPR formalism, this section proposes to focus on nested polynomial representations. First, the notations and the motivations for this new parametrization are presented. Then, it is explained why and how it is integrated in the GPR formalism. Finally, a method to *a posteriori* evaluate the projection error is introduced.

3.3.1 Nested polynomial representations

Using the notations given by Eqs. (3.2.11) and (3.2.12), for p_2,p_1,d_2 in $\mathbb{N}^*$, let $\boldsymbol{m}^{(p_2,u_2)}$ and $\boldsymbol{m}^{(p_1,u_1)}$ be the vector-valued functions that gather all the elements of $\mathcal{P}(p_2,u_2)$ and $\mathcal{P}(p_1,u_1)$ respectively, and let $\mathcal{C}(p_2,u_2)$ and $\mathcal{C}(p_1,u_1)$ be their respective dimensions. The elements of these two vectors are sorted in an increasing total polynomial order. In particular, it comes:

$$m_1^{(p_2,u_2)} \;=\; m_1^{(p_1,u_1)} = 1, \tag{3.3.1}$$

where $u_1 = d$.
Hence, for all $(u_2 \times \mathcal{C}(p_1, u_1))$-dimensional matrix $\boldsymbol{A}$ and all $\mathcal{C}(p_2, u_2)$-dimensional vector $\boldsymbol{\beta}_2$, the mapping

$$\boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{m}^{(p_1,u_1)}(\boldsymbol{x}) \tag{3.3.2}$$

is a function with values in $\mathbb{R}^{u_2}$, and the mapping

$$\boldsymbol{x} \mapsto \boldsymbol{m}^{(p_2,u_2)}(\boldsymbol{A}\boldsymbol{m}^{(p_1,u_1)}(\boldsymbol{x}))^T\boldsymbol{\beta}_2 \tag{3.3.3}$$

defines a nested polynomial representation. For $u_1 = d > 1$, such a representation allows us to model separately the dependence structure between the different input parameters, which is characterized by p_2 and u_2, and the individual actions of each input parameter, which are characterized by the polynomial order p_1 (considering different values of p_1 for each input could eventually be done to optimize such a two-scale modeling). Hence, analyzing the optimal values of p_2, u_2 and p_1 can bring information about the structure of y. For instance, if $p_2 = 1$ and $u_2 = d$, then y is just an additive model, up to a transformation of its input parameters. In the same manner, a value of p_1 strictly greater than 1 tends to say that the relation between $\boldsymbol{x}$ and y is multi-scale.

Another interesting property of this nested structure comes from the fact that, for all $\boldsymbol{x}$ in $\mathbb{R}^d$:

$$\begin{aligned}\boldsymbol{m}^{(p_2,u_2)}(\boldsymbol{A}\boldsymbol{m}^{(p_1,u_1)}(\boldsymbol{x}))^T\boldsymbol{\beta}_2 &= \sum_{0\leq|\alpha_1|+\cdots+|\alpha_{u_2}|\leq p_2} (\boldsymbol{\beta}_2)_{(\alpha_1,\ldots,\alpha_{u_2})} \times \prod_{i=1}^{u_2}\left(\sum_{k=1}^{\mathcal{C}(p_1,u_1)} \boldsymbol{A}_{ik} m_k^{(p_1,u_1)}(\boldsymbol{x})\right)^{\alpha_i}, \\ &= \sum_{0\leq|\widetilde{\alpha}_1|+\cdots+|\widetilde{\alpha}_{u_1}|\leq p_2\times p_1} x_1^{\widetilde{\alpha}_1} \times \cdots \times x_{u_1}^{\widetilde{\alpha}_{u_1}}\ \widetilde{c}_{\widetilde{\boldsymbol{\alpha}}}(\boldsymbol{A}, \boldsymbol{\beta}_2; u_2),\end{aligned} \tag{3.3.4}$$

where $\widetilde{c}_{\widetilde{\boldsymbol{\alpha}}}(\boldsymbol{A}, \boldsymbol{\beta}_2; u_2)$ is the projection coefficient of $\boldsymbol{m}^{(p_2,u_2)}(\boldsymbol{A}\boldsymbol{m}^{(p_1,u_1)}(\boldsymbol{x}))^T\boldsymbol{\beta}_2$ on $x_1^{\widetilde{\alpha}_1} \times \cdots \times x_{u_1}^{\widetilde{\alpha}_{u_1}}$. Hence, function $\boldsymbol{x} \mapsto \boldsymbol{m}^{(p_2,u_2)}(\boldsymbol{A}\boldsymbol{m}^{(p_1,u_1)}(\boldsymbol{x}))^T\boldsymbol{\beta}_2$ is in $\mathrm{Span}\{\mathcal{P}(p_2 \times p_1, u_1)\}$, while being characterized by only $\mathcal{C}(p_2, u_2) + u_2 \times \mathcal{C}(p_1, u_1)$ parameters. Thus, by choosing u_2 such that the ratio $(\mathcal{C}(p_2, u_2) + u_2 \times \mathcal{C}(p_1, u_1))/\mathcal{C}(p_2 \times p_1, u_1)$ is small, it is possible to parametrize polynomial families with very high cardinality, with only a reduced number of parameters. Such a parametrization is however redundant, in the sense that several distinct values of $\boldsymbol{A}$ and $\boldsymbol{\beta}_2$ lead to the same nested representations. From Eq. (3.3.4), it can be seen that some of these redundancies can be avoided by imposing that:

$$\begin{cases} \boldsymbol{A}_{i1} = 0, \\ \displaystyle\sum_{k=1}^{\mathcal{C}(p_1,u_1)} \boldsymbol{A}_{ik}^2 = 1, \end{cases} \quad 1 \leq i \leq u_2. \tag{3.3.5}$$

For fixed values of p_2 and p_1, it is clear that ratio $(\mathcal{C}(p_2, u_2) + u_2 \times \mathcal{C}(p_1, u_1))/\mathcal{C}(p_2 \times p_1, u_1)$ is minimal when $u_2 = 1$. However, considering higher values of u_2 strongly increases the flexibility of the nested representation to approximate function y. In this chapter, as a compromise between flexibility and minimal parametrization, for all $2 \leq k \leq \mathcal{C}(p_1, u_1)$, we thus propose to fix to zero all the components of $(\boldsymbol{A}_{1k}, \ldots, \boldsymbol{A}_{u_2k})$ but one. This means that each component of vector $\boldsymbol{m}^{(p_1,u_1)}(\boldsymbol{x})$ is used only once in the construction of $\boldsymbol{A}\boldsymbol{m}^{(p_1,u_1)}(\boldsymbol{x})$, and that only $\#\mathrm{Coeff}(p_1, p_2, u_1, u_2) = \mathcal{C}(p_2, u_2) + (\mathcal{C}(p_1, u_1) - 1) - u_2$ independent parameters have to be

Values of d	$\mathcal{C}(p_2 \times p_1, u_1)$	#Coeff$(p_1, p_2, u_1, u_2 = 1)$	#Coeff$(p_1, p_2, u_1, u_2 = d)$
1	10	6	6
2	55	12	17
5	2002	58	106
10	92378	288	561
20	10015005	1773	3521

Table 3.1: Comparison between the dimension of the projection set, $\mathcal{C}(p_2 \times p_1, u_1)$, and the number of independent parameters to characterize the associated projection coefficients in the proposed nested approach, #Coeff$(p_1, p_2, u_1, u_2) = \mathcal{C}(p_2, u_2) + (\mathcal{C}(p_1, u_1) - 1) - u_2$, for $p_1 = p_2 = 3$, $u_1 \in \{1, 2, 5, 10, 20\}$ and $u_2 \in \{1, d\}$.

fixed to span a $\mathcal{C}(p_2 \times p_1, u_1)$-dimensional projection set. As it can be seen in Table 3.1 and as it will be shown in Section 3.4, this assumption is indeed very attractive in terms of dimension reduction while being particularly interesting for the modeling of complex phenomena with very limited information.

To simplify the notations of the next sections, these $\mathcal{C}(p_1, u_1) - 1$ non-zero coefficients of $\boldsymbol{A}$ are supposed to be gathered in a vector $\boldsymbol{\beta}_1$, and we introduce the matrices $\boldsymbol{P}^{(p_1,u_1)}(\boldsymbol{x})$ such that for all $\boldsymbol{x} \in \mathbb{X}$:

$$\boldsymbol{P}^{(p_1,u_1)}(\boldsymbol{x})\boldsymbol{\beta}_1 := \boldsymbol{A}\boldsymbol{m}^{(p_1,u_1)}(\boldsymbol{x}). \tag{3.3.6}$$

For given values of $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$, we then denote by $\mu(\cdot; \boldsymbol{\beta}_1, \boldsymbol{\beta}_2)$ the following nested representation:

$$\mu(\boldsymbol{x}; \boldsymbol{\beta}_1, \boldsymbol{\beta}_2) := \boldsymbol{m}^{(p_2,u_2)}(\boldsymbol{P}^{(p_1,u_1)}(\boldsymbol{x})\boldsymbol{\beta}_1)^T\boldsymbol{\beta}_2, \quad \boldsymbol{x} \in \mathbb{X}. \tag{3.3.7}$$

Finally, for given values of u_2, p_2, p_1, the most appropriate nested representation to approximate function y is given by $\mu(\cdot; \boldsymbol{\beta}_1^*, \boldsymbol{\beta}_2^*)$, where $(\boldsymbol{\beta}_1^*, \boldsymbol{\beta}_2^*)$ is the solution of the following optimization problem:

$$(\boldsymbol{\beta}_1^*, \boldsymbol{\beta}_2^*) := \arg \min_{(\boldsymbol{\beta}_1, \boldsymbol{\beta}_2) \in \mathcal{S}^*} \|y - \mu(\cdot; \boldsymbol{\beta}_1, \boldsymbol{\beta}_2)\|_{\mathbb{X}}^2, \tag{3.3.8}$$

and the admissible searching set, $\mathcal{S}^*$, is a subset of $\mathbb{R}^{\mathcal{C}(p_1,u_1)-1} \times \mathbb{R}^{\mathcal{C}(p_2,u_2)}$ that takes into account the constraints on $\boldsymbol{\beta}_1$ defined by Eqs. (3.3.5) and (3.3.6).

Three main difficulties arise from the optimization problem defined by Eq. (3.3.8). First, as the maximal information about y is a n-dimensional set of evaluations, for given values of $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$, the norm $\|y - \mu(\cdot; \boldsymbol{\beta}_1, \boldsymbol{\beta}_2)\|_{\mathbb{X}}^2$ has to be approximated. If the evaluation points $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\}$ are (more or less) uniformly distributed on $\mathbb{X}$, a (rather) good estimation of this norm is given by its least squares approximation,

$$\frac{1}{n}\sum_{i=1}^{n}\left(y(\boldsymbol{x}^{(i)}) - \mu(\boldsymbol{x}^{(i)}; \boldsymbol{\beta}_1, \boldsymbol{\beta}_2)\right)^2 = \frac{1}{n}\left\|\boldsymbol{y}^{\text{obs}} - \boldsymbol{M}(\boldsymbol{\beta}_1)\boldsymbol{\beta}_2\right\|^2, \tag{3.3.9}$$

where the vector $\boldsymbol{y}^{\text{obs}}$ is defined by Eq. (3.2.3), and $\boldsymbol{M}(\boldsymbol{\beta}_1)$ is a $(n \times \mathcal{C}(p_2, u_2))$-dimensional matrix such that:

$$(\boldsymbol{M}(\boldsymbol{\beta}_1))_{nk} = m_k^{(p_2,u_2)}(\boldsymbol{P}^{(p_1,u_1)}(\boldsymbol{x}^{(n)})\boldsymbol{\beta}_1), \quad 1 \leq n \leq n,\ 1 \leq k \leq \mathcal{C}(p_2, u_2). \tag{3.3.10}$$

Noticing that for all $(\boldsymbol{\beta}_1, \boldsymbol{\beta}_2)$ in $\mathcal{S}^*$,

$$\left\| \boldsymbol{y}^{\text{obs}} - \boldsymbol{M}(\boldsymbol{\beta}_1) \left(\boldsymbol{M}(\boldsymbol{\beta}_1)^T \boldsymbol{M}(\boldsymbol{\beta}_1) \right)^{-1} \boldsymbol{M}(\boldsymbol{\beta}_1)^T \boldsymbol{y}^{\text{obs}} \right\|^2 \leq \left\| \boldsymbol{y}^{\text{obs}} - \boldsymbol{M}(\boldsymbol{\beta}_1)\boldsymbol{\beta}_2 \right\|^2, \tag{3.3.11}$$

the solutions, $\boldsymbol{\beta}_1^*$ and $\boldsymbol{\beta}_2^*$, of the minimization problem defined by Eq. (3.3.8) can respectively be approximated by the vectors $\boldsymbol{\beta}_1^{\text{LS}}$ and $\boldsymbol{\beta}_2^{\text{LS}}(\boldsymbol{\beta}_1^{\text{LS}})$, with:

$$\begin{cases} \boldsymbol{\beta}_1^{\text{LS}} = \arg \min_{\boldsymbol{\beta}_1 \in \mathcal{S}^*_{\boldsymbol{\beta}_1}} \left\| \boldsymbol{y}^{\text{obs}} - \boldsymbol{M}(\boldsymbol{\beta}_1)\boldsymbol{\beta}_2^{\text{LS}}(\boldsymbol{\beta}_1) \right\|^2, \\ \boldsymbol{\beta}_2^{\text{LS}}(\boldsymbol{\beta}_1) = \left(\boldsymbol{M}(\boldsymbol{\beta}_1)^T \boldsymbol{M}(\boldsymbol{\beta}_1) \right)^{-1} \boldsymbol{M}(\boldsymbol{\beta}_1)^T \boldsymbol{y}^{\text{obs}}, \end{cases} \tag{3.3.12}$$

where $\mathcal{S}^*_{\boldsymbol{\beta}_1}$ is a subset of $\mathbb{R}^{\mathcal{C}(p_1,u_1)-1}$ that also takes into account the constraints on $\boldsymbol{\beta}_1$ defined by Eqs. (3.3.5) and (3.3.6).

The second difficulty comes from the fact that the minimization of the function $\boldsymbol{\beta}_1 \mapsto \left\| \boldsymbol{y}^{\text{obs}} - \boldsymbol{M}(\boldsymbol{\beta}_1)\boldsymbol{\beta}_2^{\text{LS}}(\boldsymbol{\beta}_1) \right\|^2$ can be complex. This is due to the fact that this mapping is strongly non-linear, leading to a strongly non-convex problem. For high values of p_2, p_1 and u_2, even if non-convex optimization algorithms such as simulated annealing or simplex algorithms [Brent, 1973] are used, there is no guarantee that the global minimum can be found in a reasonable computational time.

At last, there is a risk that $\left\| \boldsymbol{y}^{\text{obs}} - \boldsymbol{M}(\boldsymbol{\beta}_1^{\text{LS}})\boldsymbol{\beta}_2^{\text{LS}}(\boldsymbol{\beta}_1^{\text{LS}}) \right\|^2 / n$ strongly underestimates $\left\| y - \mu(\cdot; \boldsymbol{\beta}_1^{\text{LS}}, \boldsymbol{\beta}_2^{\text{LS}}(\boldsymbol{\beta}_1^{\text{LS}})) \right\|^2_{\mathbb{X}}$, as the same information is used twice: once for the optimization and once for the error estimation. To avoid such an over-fitting, classical Leave-One-Out (LOO) techniques (see Miller [1974]; Blatman and Sudret [2011]; Perrin et al. [2014]) have to be introduced to get a relevant approximation of $\left\| y - \mu(\cdot; \boldsymbol{\beta}_1^{\text{LS}}, \boldsymbol{\beta}_2^{\text{LS}}(\boldsymbol{\beta}_1^{\text{LS}})) \right\|^2_{\mathbb{X}}$.

3.3.2 Coupling nested representations and Gaussian processes

Once vector $\boldsymbol{\beta}_1^{\text{LS}}$ has been identified from the solving of Eq. (3.3.12), the notion of confidence intervals for the prediction of $y(\boldsymbol{x})$ at an unobserved point $\boldsymbol{x}$ can be found back by assuming that y is a particular realization of a Gaussian stochastic process, whose statistical properties are given by:

$$Y(\cdot) \sim \text{GP}\left(\mu\left(\cdot; \boldsymbol{\beta}_1^{\text{LS}}, \boldsymbol{\beta}_2^{\text{LS}}(\boldsymbol{\beta}_1^{\text{LS}})\right), C\left(\cdot, \cdot; \widehat{\boldsymbol{\Theta}}^{LS}\right)\right), \tag{3.3.13}$$

where $\widehat{\boldsymbol{\Theta}}^{LS}$ gathers the $d+1$ parameters of the Matern-5/2 covariance C defined by Eq. (3.2.7), which are solution of the following log-likelihood maximization problem:

$$\widehat{\boldsymbol{\Theta}}^{LS} = \underset{\boldsymbol{\Theta} \in (0,+\infty)^{d+1}}{\text{argmax}} -\frac{1}{2} \left[\begin{array}{c} n \log(2\pi) + \log(\det(\boldsymbol{R}(\boldsymbol{\Theta}))) + \\ (\boldsymbol{y}^{\text{obs}} - \boldsymbol{M}(\boldsymbol{\beta}_1^{\text{LS}})\boldsymbol{\beta}_2^{\text{LS}}(\boldsymbol{\beta}_1^{\text{LS}}))^T \boldsymbol{R}(\boldsymbol{\Theta})^{-1} (\boldsymbol{y}^{\text{obs}} - \boldsymbol{M}(\boldsymbol{\beta}_1^{\text{LS}})\boldsymbol{\beta}_2^{\text{LS}}(\boldsymbol{\beta}_1^{\text{LS}})) \end{array} \right]. \tag{3.3.14}$$

Such a naive coupling is nevertheless sub-optimal, as the values of $\boldsymbol{\beta}_1$ and $\boldsymbol{\Theta}$ are optimized separately: the nested structure does not take advantage of the Bayesian formalism, and reciprocally. Instead of such a two-steps approach, we propose in this chapter to directly adopt a Bayesian formalism for the estimation of $\boldsymbol{\beta}_1$ and $\boldsymbol{\Theta}$. In the plug-in formalism, this means that the statistical properties of Y are now given by:

$$Y(\cdot) \sim \text{GP}\left(\mu\left(\cdot; \widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\beta}}_2\right), C\left(\cdot, \cdot; \widehat{\boldsymbol{\Theta}}\right)\right), \tag{3.3.15}$$

where $(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\beta}}_2, \widehat{\boldsymbol{\Theta}})$ is the solution of the following log-likelihood maximization problem:

$$(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\beta}}_2, \widehat{\boldsymbol{\Theta}}_1) = \underset{(\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \boldsymbol{\Theta}) \in \mathcal{S}^{\mathrm{adm}}}{\operatorname{argmax}} -\frac{1}{2} \left[\begin{array}{c} n \log(2\pi) + \log(\det(\boldsymbol{R}(\boldsymbol{\Theta}))) \\ +(\boldsymbol{y}^{\mathrm{obs}} - \boldsymbol{M}(\boldsymbol{\beta}_1)\boldsymbol{\beta}_2)^T \boldsymbol{R}(\boldsymbol{\Theta})^{-1} (\boldsymbol{y}^{\mathrm{obs}} - \boldsymbol{M}(\boldsymbol{\beta}_1)\boldsymbol{\beta}_2) \end{array} \right], \tag{3.3.16}$$

where the admissible searching set, $\mathcal{S}^{\mathrm{adm}}$, is a subset of $\mathbb{R}^{\mathcal{C}(p_1,u_1)-1} \times \mathbb{R}^{\mathcal{C}(p_2,u_2)} \times \mathbb{R}^{d+1}$ but is not trivial, as it first takes into account the constraints on $\boldsymbol{\beta}_1$ defined by Eqs. (3.3.5) and (3.3.6), but also guarantees that $\boldsymbol{R}(\boldsymbol{\Theta})$ and $\boldsymbol{M}(\boldsymbol{\beta}_1)^T \boldsymbol{R}(\boldsymbol{\Theta})^{-1} \boldsymbol{M}(\boldsymbol{\beta}_1)$ are invertible.

For all $(\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \boldsymbol{\Theta})$ belonging to the admissible set, $\mathcal{S}^{\mathrm{adm}}$, we denote by L the function such that:

$$L(\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \boldsymbol{\Theta}) = \log(\det(\boldsymbol{R}(\boldsymbol{\Theta}))) + (\boldsymbol{y}^{\mathrm{obs}} - \boldsymbol{M}(\boldsymbol{\beta}_1)\boldsymbol{\beta}_2)^T \boldsymbol{R}(\boldsymbol{\Theta})^{-1} (\boldsymbol{y}^{\mathrm{obs}} - \boldsymbol{M}(\boldsymbol{\beta}_1)\boldsymbol{\beta}_2). \tag{3.3.17}$$

It is interesting to notice that, in the same manner as in Section 3.3.1,

$$L(\boldsymbol{\beta}_1, \boldsymbol{\beta}_2^{\mathrm{LS}}(\boldsymbol{\beta}_1, \boldsymbol{\Theta}), \boldsymbol{\Theta}) \leq L(\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \boldsymbol{\Theta}), \tag{3.3.18}$$

$$\boldsymbol{\beta}_2^{\mathrm{LS}}(\boldsymbol{\beta}_1, \boldsymbol{\Theta}) := \left(\boldsymbol{M}(\boldsymbol{\beta}_1)^T \boldsymbol{R}(\boldsymbol{\Theta})^{-1} \boldsymbol{M}(\boldsymbol{\beta}_1)\right)^{-1} \boldsymbol{M}(\boldsymbol{\beta}_1)^T \boldsymbol{R}(\boldsymbol{\Theta}) \boldsymbol{y}^{\mathrm{obs}}. \tag{3.3.19}$$

It comes:

$$\left\{ \begin{array}{l} (\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}}_1) = \arg \min_{(\boldsymbol{\beta}_1, \boldsymbol{\Theta})} \mathcal{L}(\boldsymbol{\beta}_1, \boldsymbol{\Theta}), \\ \widehat{\boldsymbol{\beta}}_2 = \left(\boldsymbol{M}(\widehat{\boldsymbol{\beta}}_1)^T \boldsymbol{R}(\widehat{\boldsymbol{\Theta}})^{-1} \boldsymbol{M}(\widehat{\boldsymbol{\beta}}_1)\right)^{-1} \boldsymbol{M}(\widehat{\boldsymbol{\beta}}_1)^T \boldsymbol{R}(\widehat{\boldsymbol{\Theta}}) \boldsymbol{y}^{\mathrm{obs}}, \end{array} \right. \tag{3.3.20}$$

where:

$$\mathcal{L}(\boldsymbol{\beta}_1, \boldsymbol{\Theta}) := L(\boldsymbol{\beta}_1, \boldsymbol{\beta}_2^{\mathrm{LS}}(\boldsymbol{\beta}_1, \boldsymbol{\Theta}), \boldsymbol{\Theta}). \tag{3.3.21}$$

Function $(\boldsymbol{\beta}_1, \boldsymbol{\Theta}) \mapsto \mathcal{L}(\boldsymbol{\beta}_1, \boldsymbol{\Theta})$ being strongly non-regular and non-convex, it is proposed to work iteratively on the values of $\boldsymbol{\beta}_1$ and $\boldsymbol{\Theta}$. Two reasons motivate this separation. First, the actions of $\boldsymbol{\beta}_1$ and $\boldsymbol{\Theta}$ on $\mathcal{L}(\boldsymbol{\beta}_1, \boldsymbol{\Theta})$ being very different, dividing the optimization problem tends to regularize the mappings on which the minimization is carried out. Second, by reducing each searching set, each minimization is made easier. Therefore, for a given convergence tolerance ε, Algorithm 1 is introduced for the minimization of $\mathcal{L}$. The convergence of such an iterative algorithm to the global minimum of $\mathcal{L}$ is of course not guaranteed, but it appeared on a series of numerical examples that it allowed us to identify good approximations of $(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}})$ at a reasonable computational cost. As the minimization problem defined by Eq. (3.3.20) is not convex, better approximations of $\widehat{\boldsymbol{\beta}}_1$ can be obtained by repeating several times Algorithm 1, with random initialization of vectors $(\boldsymbol{\beta}_1)_0$ in $\mathcal{S}^*_{\boldsymbol{\beta}_1}$.

3.3.3 Linearization of the nested polynomial trend

Even for small values of p_2, p_1 and u_2, the quantity $\mathcal{L}(\boldsymbol{\beta}_1, \boldsymbol{\Theta})$ is sensitive to small changes in the values of $\boldsymbol{\beta}_1$ and $\boldsymbol{\Theta}$, which makes the solving of the optimization problem defined by Eq. (3.3.20) difficult. In that context, it can be interesting to linearize the nested polynomial trend around the solutions given by Algorithm 1, $\widehat{\boldsymbol{\beta}}_1$ and $\widehat{\boldsymbol{\beta}}_2$, and then work on the compensations $(\boldsymbol{\beta}_1 - \widehat{\boldsymbol{\beta}}_1)$ and $(\boldsymbol{\beta}_2 - \widehat{\boldsymbol{\beta}}_2)$ that could make the prediction of function y better. In the vicinity of $\widehat{\boldsymbol{\beta}}_1$ and $\widehat{\boldsymbol{\beta}}_2$, for all $\boldsymbol{x}$ in $\mathbb{X}$, it comes:

$$\mu(\boldsymbol{x}; \boldsymbol{\beta}_1, \boldsymbol{\beta}_2) \approx \left(\boldsymbol{h}_1(\boldsymbol{x}; \widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\beta}}_2), \boldsymbol{h}_2(\boldsymbol{x}; \widehat{\boldsymbol{\beta}}_1)\right)^T (\boldsymbol{\beta}_1 - \widehat{\boldsymbol{\beta}}_1, \boldsymbol{\beta}_2), \tag{3.3.22}$$

Initialization: $L_1 = 0$, $L_2 = +\infty$, $\boldsymbol{\beta}_1^* = (\boldsymbol{\beta}_1)_0 \in \mathcal{S}_{\boldsymbol{\beta}_1}^*$;
while $|L_2 - L_1| > \varepsilon$ **do**
  $L_1 = L_2$;
  $\boldsymbol{\Theta}^* = \arg\max_{\boldsymbol{\Theta}} \mathcal{L}(\boldsymbol{\beta}_1^*, \boldsymbol{\Theta})$;
  $\boldsymbol{\beta}_1^* = \arg\max_{\boldsymbol{\beta}_1} \mathcal{L}(\boldsymbol{\beta}_1, \boldsymbol{\Theta}^*)$;
  $L_2 = \min(L_2, \mathcal{L}(\boldsymbol{\beta}_1^*, \boldsymbol{\Theta}^*))$;
end
$\widehat{\boldsymbol{\beta}}_1 \approx \boldsymbol{\beta}_1^*$, $\widehat{\boldsymbol{\Theta}} \approx \boldsymbol{\Theta}^*$.

Algorithm 1: Iterative minimization of function $\mathcal{L}$.

$$\boldsymbol{h}_1(\boldsymbol{x}; \widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\beta}}_2) = \boldsymbol{P}^{(p_1,u_1)}(\boldsymbol{x})^T \boldsymbol{D}(\boldsymbol{P}^{(p_1,u_1)}(\boldsymbol{x})\widehat{\boldsymbol{\beta}}_1)^T \widehat{\boldsymbol{\beta}}_2, \tag{3.3.23}$$

$$\boldsymbol{h}_2(\boldsymbol{x}; \widehat{\boldsymbol{\beta}}_1) = \boldsymbol{m}^{(p_2,u_2)}(\boldsymbol{P}^{(p_1,u_1)}(\boldsymbol{x})\widehat{\boldsymbol{\beta}}_1), \tag{3.3.24}$$

$$(\boldsymbol{D}(\boldsymbol{z}))_{kj} := \frac{\partial m_k^{(p_2,u_2)}}{\partial z_j}(\boldsymbol{z}), \quad 1 \leq j \leq u_2, \quad 1 \leq k \leq \mathcal{C}(p_2, u_2), \ \boldsymbol{z} \in \mathbb{R}^{u_2}. \tag{3.3.25}$$

Now, let us denote by $\boldsymbol{\beta} := (\boldsymbol{\beta}_1 - \widehat{\boldsymbol{\beta}}_1, \boldsymbol{\beta}_2)$ the new vector of parameters we need to determine, and by $\boldsymbol{h} := \left(\boldsymbol{h}_1(\cdot; \widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\beta}}_2), \boldsymbol{h}_2(\cdot; \widehat{\boldsymbol{\beta}}_1)\right)$ the new set of projection functions. Conditioned by the values of $\widehat{\boldsymbol{\beta}}_1$, $\widehat{\boldsymbol{\beta}}_2$ and $\widehat{\boldsymbol{\Theta}}$, the formalism introduced in Section 3.2.3 is found back:

$$Y(\cdot) \ \sim \ \mathrm{GP}\left(\boldsymbol{h}(\cdot)^T \boldsymbol{\beta}, C(\cdot,\cdot)\right), \tag{3.3.26}$$

such that the distribution of $(Y \mid \boldsymbol{y}^{\mathrm{obs}})$ can be calculated analytically. Its mean value can directly be used to predict the values of y, and its covariance function can allow us to quantify the confidence we can put in these predictions.

We underline at least two advantages for the linearization. First, the distribution of $(Y \mid \boldsymbol{y}^{\mathrm{obs}})$ will be less dependent on the convergence properties of Algorithm 1, which are not easy to control. Secondly, as the covariance function of $(Y \mid \boldsymbol{y}^{\mathrm{obs}})$ integrates the uncertainty associated with the least squares estimation of $\boldsymbol{\beta}$, that is to say the uncertainty associated with the estimation of $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ in the vicinity of $\widehat{\boldsymbol{\beta}}_1$ and $\widehat{\boldsymbol{\beta}}_2$, the confidence intervals associated with these predictions are expected to be more adapted.

3.3.4 Error evaluation

According to the previous Sections and to Eq. (3.2.9), for given values of truncation parameters p_2, p_1 and u_2, we propose to use the deterministic function $\widehat{y}^{\mathrm{nest}}(\boldsymbol{x})$, such that:

$$\widehat{y}^{\mathrm{nest}}(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x}; \widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}})^T \widehat{\boldsymbol{\beta}}(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}}) + \boldsymbol{r}(\boldsymbol{x}; \widehat{\boldsymbol{\Theta}})^T \boldsymbol{R}(\widehat{\boldsymbol{\Theta}})^{-1} \left(\boldsymbol{y}^{\mathrm{obs}} - \boldsymbol{H}(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}}) \widehat{\boldsymbol{\beta}}(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}})\right), \tag{3.3.27}$$

$$\widehat{\boldsymbol{\beta}}(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}}) := (\boldsymbol{H}(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}})^T \boldsymbol{R}(\widehat{\boldsymbol{\Theta}})^{-1} \boldsymbol{H}(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}}))^{-1} \boldsymbol{H}(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}})^T \boldsymbol{R}(\widehat{\boldsymbol{\Theta}})^{-1} \boldsymbol{y}^{\mathrm{obs}}, \tag{3.3.28}$$

to predict the value of $y(\boldsymbol{x})$ for all $\boldsymbol{x}$ in $\mathbb{X}$, where:

- vectors $\widehat{\boldsymbol{\beta}}_1$ and $\widehat{\boldsymbol{\Theta}}$ are the solutions of the optimization problem given by Eq. (3.3.20), under the additional condition that the matrix $\boldsymbol{H}(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}})^T \boldsymbol{R}(\widehat{\boldsymbol{\Theta}})^{-1} \boldsymbol{H}(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}})$ is invertible,

- vector $\boldsymbol{y}^{\text{obs}}$ is defined by Eq. (3.2.3),
- the function $\boldsymbol{x} \mapsto \boldsymbol{h}(\boldsymbol{x}; \widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}})$ gathers the most influential terms of the vector-valued function $\left(\boldsymbol{h}_1(\cdot; \widehat{\boldsymbol{\beta}}_1, \boldsymbol{\beta}_2^{\text{LS}}(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}})), \boldsymbol{h}_2(\cdot; \widehat{\boldsymbol{\beta}}_1)\right)$, which have been identified from a LAR procedure,
- $\boldsymbol{H}(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}}) := [\boldsymbol{h}(\boldsymbol{x}^{(1)}; \widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}}) \ \cdots \ \boldsymbol{h}(\boldsymbol{x}^{(n)}; \widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}})]$ is the matrix that gathers the evaluations of $\boldsymbol{h}(\cdot; \widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}})$ at the available code evaluations,
- and for all $1 \leq i, j \leq n$, $\boldsymbol{R}(\widehat{\boldsymbol{\Theta}})_{ij} = C(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$ and $r_i(\boldsymbol{x}; \widehat{\boldsymbol{\Theta}}) = C(\boldsymbol{x}, \boldsymbol{x}^{(i)})$, with C the Matern-5/2 covariance function of parameters $\widehat{\boldsymbol{\Theta}}$.

In the same manner as in Section 3.2, when function y is only known through a limited number of evaluations, classical Leave-One-Out (LOO) techniques have to be introduced to approximate the relevance of such a predictor:

$$\left\| y - \widehat{y}^{\text{nest}} \right\|_{L_2}^2 \approx \epsilon_{\text{LOO}}^2 := \frac{1}{n} \sum_{i=1}^{n} \left(y(\boldsymbol{x}^{(i)}) - \widehat{y}_{-i}^{\text{nest}}(\boldsymbol{x}^{(i)}) \right)^2, \tag{3.3.29}$$

where, for all $1 \leq i \leq n$, the function $\widehat{y}_{-i}^{\text{nest}}$ has been constructed in the same manner as $\widehat{y}^{\text{nest}}$, but using the $n-1$ evaluations of the code in $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(i-1)}, \boldsymbol{x}^{(i+1)}, \ldots, \boldsymbol{x}^{(n)}\}$ only.

In order to reduce the computational cost associated with the evaluation of ϵ_{LOO}^2, it is interesting to notice (see Dubrule [1983] for further details) that, for all $1 \leq i \leq n$:

$$y(\boldsymbol{x}^{(i)}) - \widehat{y}_{-i}^{\text{nest}}(\boldsymbol{x}^{(i)}) = \frac{(\widehat{\boldsymbol{C}}(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}}) \boldsymbol{y}^{\text{obs}})_i}{\widehat{\boldsymbol{C}}(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}})_{ii}}, \tag{3.3.30}$$

$$\begin{aligned} \widehat{\boldsymbol{C}}(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}}) = \ & \boldsymbol{R}(\widehat{\boldsymbol{\Theta}})^{-1} - \\ & \boldsymbol{R}(\widehat{\boldsymbol{\Theta}})^{-1} \boldsymbol{H}\left(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}}\right) \left(\boldsymbol{H}(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}})^T \boldsymbol{R}(\widehat{\boldsymbol{\Theta}})^{-1} \boldsymbol{H}(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}}) \right)^{-1} \boldsymbol{H}\left(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}}\right)^T \boldsymbol{R}(\widehat{\boldsymbol{\Theta}})^{-1}. \end{aligned} \tag{3.3.31}$$

LOO error ϵ_{LOO}^2 can then be approximated by:

$$\epsilon_{\text{LOO}}^2 \approx \widehat{\epsilon}_{\text{LOO}}^2 := \frac{1}{n} \sum_{i=1}^{n} \widehat{e}_i^2, \quad \widehat{e}_i^2 := \left[\frac{(\widehat{\boldsymbol{C}}(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}}) \boldsymbol{y}^{\text{obs}})_i}{\widehat{\boldsymbol{C}}(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\Theta}})_{ii}} \right]^2. \tag{3.3.32}$$

Such an approximation is however conditioned by the values of $\widehat{\boldsymbol{\beta}}_1$ and $\widehat{\boldsymbol{\Theta}}$, which are computed using all the code evaluations. In order to be more precise, it can be noticed that for all $\boldsymbol{\beta}_1$, $\boldsymbol{\Theta}$, $1 \leq i \leq n$:

$$\mathcal{L}(\boldsymbol{\beta}_1, \boldsymbol{\Theta}) = \mathcal{L}_{-i}(\boldsymbol{\beta}_1, \boldsymbol{\Theta}) + \frac{(\widetilde{\boldsymbol{C}}(\boldsymbol{\beta}_1, \boldsymbol{\Theta}) \boldsymbol{y}^{\text{obs}})_i^2}{\widetilde{\boldsymbol{C}}(\boldsymbol{\beta}_1, \boldsymbol{\Theta})_{ii}}, \tag{3.3.33}$$

$$\widetilde{\boldsymbol{C}}(\boldsymbol{\beta}_1, \boldsymbol{\Theta}) = \boldsymbol{R}(\boldsymbol{\Theta})^{-1} \{ \boldsymbol{I} - \boldsymbol{M}(\boldsymbol{\beta}_1) (\boldsymbol{M}(\boldsymbol{\beta}_1)^T \boldsymbol{R}(\boldsymbol{\Theta})^{-1} \boldsymbol{M}(\boldsymbol{\beta}_1))^{-1} \boldsymbol{M}(\boldsymbol{\beta}_1)^T \boldsymbol{R}(\boldsymbol{\Theta})^{-1} \}, \tag{3.3.34}$$

where $\boldsymbol{I}$ is the identity matrix and $\mathcal{L}_{-i}(\boldsymbol{\beta}_1, \boldsymbol{\Theta})$ is the evaluation of function $\mathcal{L}(\boldsymbol{\beta}_1, \boldsymbol{\Theta})$ based on the $n-1$ evaluations of the code in $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(i-1)}, \boldsymbol{x}^{(i+1)}, \ldots, \boldsymbol{x}^{(n)}\}$ only. Hence, in the optimization process leading us to the identification of $\widehat{\boldsymbol{\beta}}_1$ and $\widehat{\boldsymbol{\Theta}}$, let $\{((\boldsymbol{\beta}_1)_i, \boldsymbol{\Theta}_i),\ 1 \leq i \leq n_{\text{test}}\}$ be the n_{test} values of $\boldsymbol{\beta}_1$ and $\boldsymbol{\Theta}$, in which function $\mathcal{L}$ has been evaluated. With a

very limited additional computational cost, we can then define, for all $1 \leq i \leq n$, the LOO evaluations of $\widehat{\boldsymbol{\beta}}_1$ and $\widehat{\boldsymbol{\Theta}}$, which are denoted by $\left(\widehat{\boldsymbol{\beta}}_1\right)_{-i}$ and $\widehat{\boldsymbol{\Theta}}_{-i}$ respectively, and which are given by:

$$\left(\left(\widehat{\boldsymbol{\beta}}_1\right)_{-i}, \widehat{\boldsymbol{\Theta}}_{-i}\right) = \arg \min_{(\boldsymbol{\beta}_1, \boldsymbol{\Theta}) \in \{((\boldsymbol{\beta}_1)_i, \boldsymbol{\Theta}_i),\ 1 \leq i \leq n_{\text{test}}\}} \mathcal{L}_{-i}(\boldsymbol{\beta}_1, \boldsymbol{\Theta}). \tag{3.3.35}$$

Finally, we can introduce error $\widetilde{\epsilon}_{\text{LOO}}$, such that:

$$\left\|y - \widehat{y}^{\text{nest}}\right\|_{L_2}^2 \approx \widetilde{\epsilon}_{\text{LOO}}^2 := \frac{1}{n}\sum_{i=1}^{n} \widetilde{e}_i^2, \quad \widetilde{e}_i^2 := \left[\frac{\left(\widehat{\boldsymbol{C}}\left(\left(\widehat{\boldsymbol{\beta}}_1\right)_{-i}, \widehat{\boldsymbol{\Theta}}_{-i}\right)\boldsymbol{y}^{\text{obs}}\right)_i}{\widehat{\boldsymbol{C}}\left(\left(\widehat{\boldsymbol{\beta}}_1\right)_{-i}, \widehat{\boldsymbol{\Theta}}_{-i}\right)_{ii}}\right]^2. \tag{3.3.36}$$

3.3.5 Convergence analysis

All the developments presented in Sections 3.3.1 and 3.3.2 are conditioned by the values of three truncation parameters, p_2, p_1 and u_2, which have to be identified from a convergence analysis. As presented in Section 3.3.1, we remind that the roles of p_2, p_1 and u_2 in the modeling of y are different. Whereas p_2 and u_2 are associated with the modeling of the dependency structure between the input parameters, p_1 is associated with the individual transformation of each input. As a consequence, p_1 is strongly dependent on the dimension of vector $\boldsymbol{\beta}_1$, which parametrizes these individual transformations. On the contrary, this dimension of $\boldsymbol{\beta}_1$, which is equal to $\mathcal{C}(p_1, u_1) - 1 - u_2$, does not depend on p_2, but depends only linearly on u_2. Hence, increasing the values of p_2 and u_2 does not really increase the dimension of the search set for the identification of $\widehat{\boldsymbol{\beta}}_1$, but makes the relation between $\boldsymbol{\beta}_1$ and $\mathcal{L}(\boldsymbol{\beta}_1, \boldsymbol{\Theta})$ much more complex.
For the choice of u_2, p_2 and p_1, maximal values $u_2^{\max}$, $p_2^{\max}$ and $q^{\max}$ are *a priori* chosen. In this chapter, since we want to reduce the number of parameters on which the polynomial trend is based, only values of u_2 that are lower than d are considered: $u_2^{\max} = d$. Finally, the optimal value of (u_2, p_1, p_2) is the one that gives the minimum LOO error among all these tested combinations of values:

$$(u_2^\star, p_1^\star, p_2^\star) := \underset{\substack{1 \leq u_2 \leq d,\\ 1 \leq p_2 \leq p_2^{\max},\\ 1 \leq p_1 \leq p_1^{\max}}}{\text{argmin}} \quad \widetilde{\epsilon}_{\text{LOO}}^2(u_2, p_1, p_2), \tag{3.3.37}$$

where error $\widetilde{\epsilon}_{\text{LOO}}^2$ is defined by Eq. (3.3.32).

3.4 Applications

To illustrate the advantages of the nested structure presented in Section 3.3 for the modeling of the quantity of interest y, this section introduces a series of analytic examples, which are sorted with respect to the input set dimension, d. In each case, the proposed approach is compared to the "LAR+UK" approach, which has been described in Section 3.2. For each function y, let $\widehat{y}^{\text{nest}}$ and $\widehat{y}^{\text{LAR+UK}}$ be the best approximations of y we can get from the available information, when considering a nested polynomial trend and a simple polynomial trend, respectively. Let $\varepsilon_{\text{NEST}}^2$ and $\varepsilon_{\text{LAR+UK}}^2$ be the associated normalized errors, such that:

$$\varepsilon_{\text{NEST}}^2 = \left\|y - \widehat{y}^{\text{nest}}\right\|_{\mathbb{X}}^2 / \left\|g\right\|_{\mathbb{X}}^2, \tag{3.4.1}$$

$$\varepsilon^2_{\mathrm{LAR+UK}} = \left\| y - \widehat{y}^{\mathrm{LAR+UK}} \right\|^2_{\mathbb{X}} / \left\| g \right\|^2_{\mathbb{X}}. \tag{3.4.2}$$

When dealing with a simple polynomial trend, it is reminded that the only truncation parameter that needs to be identified is the maximal total polynomial order, which will be denoted in the following by $p^{\mathrm{LAR+UK}}$ for the sake of clarity. On the contrary, three truncation parameters have to be identified for the nested polynomial trends: p_2, u_2 and p_1. As a consequence, the required computational time to identify $\widehat{y}^{\mathrm{nest}}$ can be much higher than the one required to identify $\widehat{y}^{\mathrm{LAR+UK}}$.

3.4.1 $d = 1$

In this part, we suppose that $d = 1$, and we fix $\mathbb{X} = [-1, 1]$. Three analytic expressions for y are then proposed:

- case 1: $y(x) = P_2 \circ P_1(x)$,
- case 2: $y(x) = \sin((x+1)^3)$,
- case 3: $y(x) = \sin(20x)\cos(2x)$,

where, for all x in $[-1, 1]$:

$$\begin{cases} P_1(x) = \displaystyle\sum_{i=1}^{5} c_i^{(1)} x^{i-1}, \quad \boldsymbol{c}^{(1)} = \dfrac{(0, -0.03, 0.5, -0.4, -0.5)}{\sqrt{0.03^2 + 0.5^2 + 0.4^2 + 0.5^2}}, \\ P_2(x) = \displaystyle\sum_{i=1}^{5} c_i^{(2)} x^{i-1}, \quad \boldsymbol{c}^{(2)} = (-0.1, 0.2, 0.7, -0.2, -0.2). \end{cases} \tag{3.4.3}$$

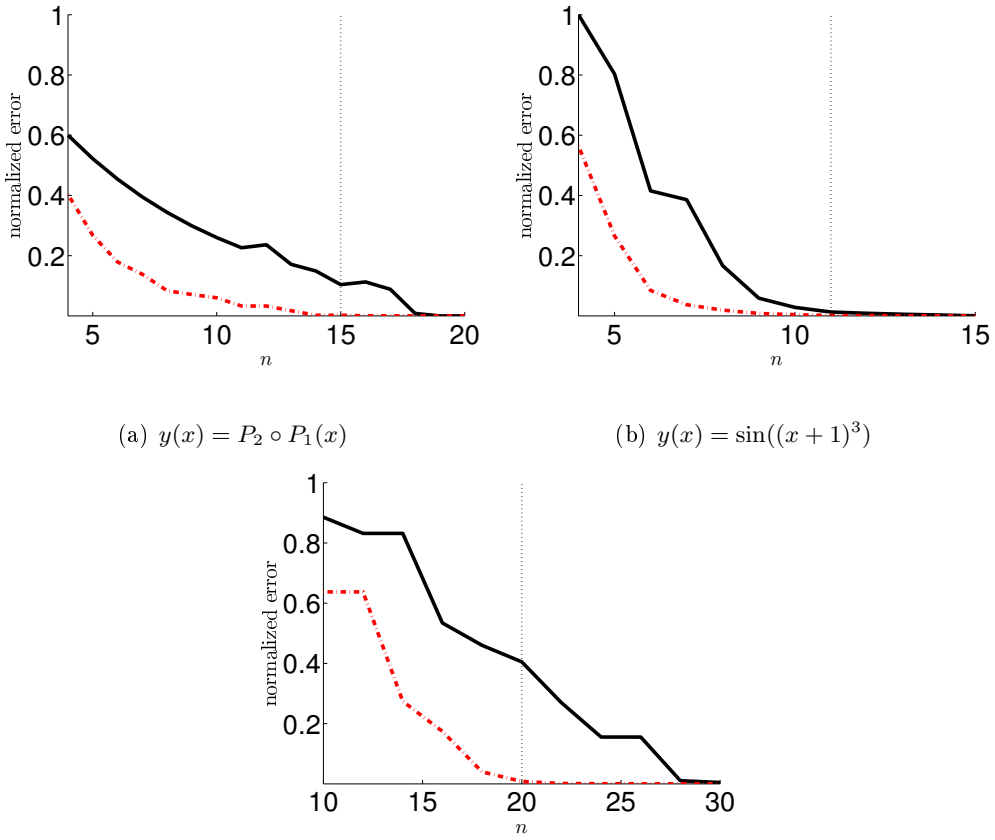
The two first examples are based on chained codes. The third example is introduced to show that this nested structure for the mean can also be interesting for non-chained codes when few code evaluations are available.

For each case, Figure 3.1 compares the evolution of the errors $\varepsilon^2_{\mathrm{NEST}}$ and $\varepsilon^2_{\mathrm{LAR+UK}}$ with respect to n, the number of available evaluations of y. For each value of n, convergence analyses have been performed for both methods. The maximal values for the truncation parameters associated were fixed such that:

$$0 \leq p^{\mathrm{LAR+UK}} \leq 20, \quad 0 \leq p_1, p_2 \leq 10, \quad u_2 = 1. \tag{3.4.4}$$

For the three applications, these convergence analyses lead us to relatively high values for these truncation parameters ($p_1 \geq 4$, $p_2 \geq 4$). As underlined in Section 3.3.1, this can be explained by the ability of the proposed nested structure to parametrize polynomial families with very high cardinality with only few parameters. This is particularly efficient when n is small compared to the number of oscillations of y.

In addition, Figure 3.2 compares the two approaches in terms of prediction for given values of n. In these figures we notice that the proposed method is particularly adapted to the cases when y presents a nested structure or is oscillating. This is particularly true when n is small compared to the complexity of y.

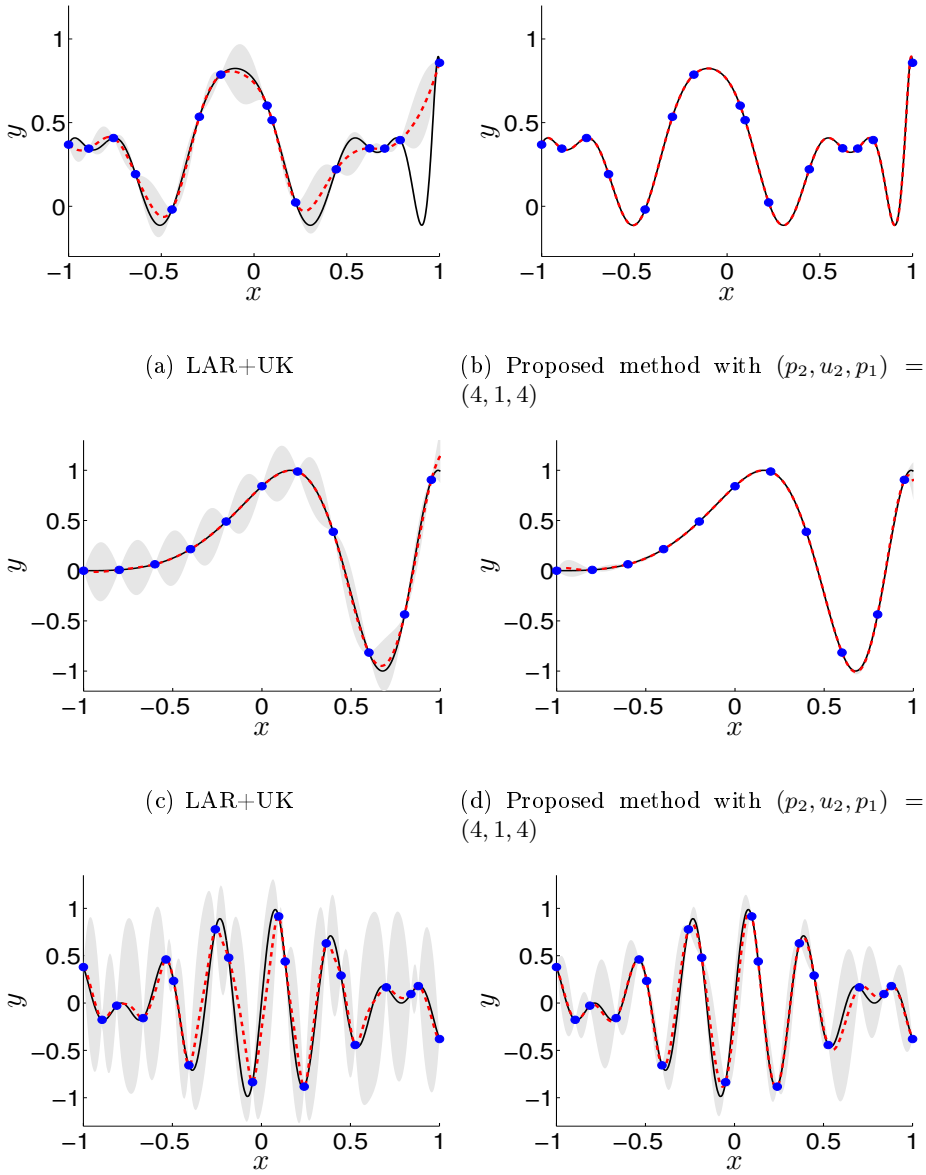


(a) $y(x) = P_2 \circ P_1(x)$

(b) $y(x) = \sin((x+1)^3)$

(c) $y(x) = \sin(20x)\cos(2x)$

Figure 3.1: Evolution of the normalized L^2 errors with respect to n, the number of code evaluations. To be more representative, for each value of n, the LAR+UK and the proposed approaches have been repeated 10 times on randomly chosen learning sets. The curves correspond to the mean value of the errors associated with these 10 repetitions. Solid black line: evolution of the error associated with the LAR+UK approach, $\varepsilon^2_{\text{LAR+UK}}$. Red dotted line: evolution of the error associated with the proposed approach, $\varepsilon^2_{\text{NEST}}$. The vertical bar indicates moreover the value of n on which the results of Figure 3.2 are focused.



(a) LAR+UK

(b) Proposed method with $(p_2, u_2, p_1) = (4, 1, 4)$

(c) LAR+UK

(d) Proposed method with $(p_2, u_2, p_1) = (4, 1, 4)$

(e) LAR+UK

(f) Proposed method with $(p_2, u_2, p_1) = (6, 1, 7)$

Figure 3.2: Efficiency of the proposed method to predict in an unobserved point the value of $y(x) = P_2 \circ P_1(x)$ with $n = 15$ (first row), $y(x) = \sin((x+1)^3)$ with $n = 11$ (second row) and $y(x) = \sin(20x)\cos(2x)$ with $n = 20$ (third row). In each figure, the black solid line is the evolution of the quantity of interest, y, with respect to x, the blue points are the positions of the available observations of y, the red dotted line is the prediction of y based on an optimized LAR+UK approach (left column) or based on the proposed approach associated with optimized values of p_2, u_2 and p_1 (right column). The grey areas correspond to the 95% confidence region for the prediction.

3.4.2 $d > 1$

The idea of this section is to show that the tendencies that were noticed in the one-dimensional cases are found back when considering multidimensional input spaces. To this end, let us consider the three following expressions of y, which can also be seen as particular chained codes, and the associated maximal values for the convergence analyses:

- Case 1: $d = 2$, $0 \leq p^{\text{LAR+UK}} \leq 20$, $0 \leq p_2 \leq 6$, $0 \leq p_1 \leq 10$, $1 \leq u_2 \leq d$.

$$g : \left\{ \begin{array}{ccc} [-1,1]^2 & \to & [-1,1] \\ \boldsymbol{x} & \mapsto & g^{\text{2D}}(\boldsymbol{x}) = (1-x_1^2)\cos(7x_1) \times (1-x_2^2)\sin(5x_2) \end{array} \right. . \tag{3.4.5}$$

- Case 2 (the Ishigami function): $d = 3$, $0 \leq p^{\text{LAR+UK}} \leq 20$, $0 \leq p_2 \leq 3$, $0 \leq p_1 \leq 10$, $1 \leq u_2 \leq d$.

$$g : \left\{ \begin{array}{ccc} [-\pi,\pi]^3 & \to & \mathbb{R} \\ \boldsymbol{x} = (x_1, x_2, x_3) & \mapsto & g^{\text{3D}}(\boldsymbol{x}) = \sin(x_1) + 7\sin(x_2)^2 + 0.1x_3^4\sin(x_1) \end{array} \right. . \tag{3.4.6}$$

- Case 3: $d = 6$, $0 \leq p^{\text{LAR+UK}} \leq 10$, $0 \leq p_2 \leq 3$, $0 \leq p_1 \leq 10$, $1 \leq u_2 \leq d$.

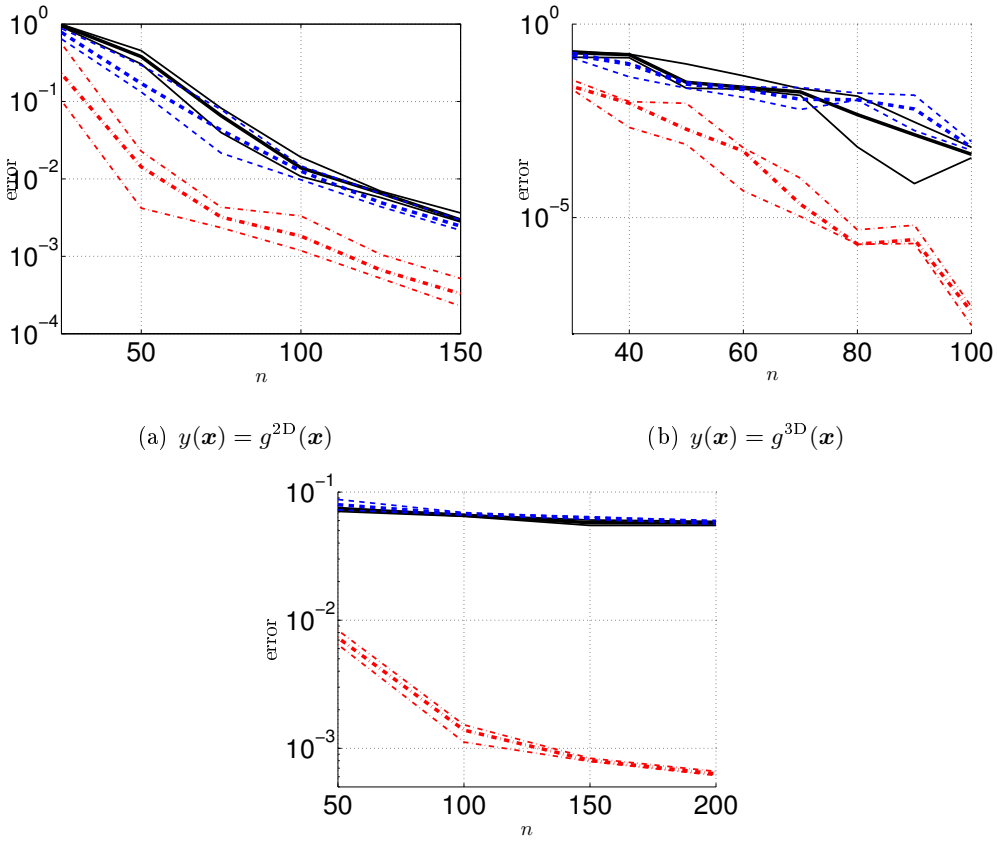
$$g : \left\{ \begin{array}{ccc} [-1,1]^6 & \to & \mathbb{R} \\ \boldsymbol{x} & \mapsto & g^{\text{6D}}(\boldsymbol{x}) = g^{(1)} \circ \boldsymbol{g}^{(2)}(\boldsymbol{x}), \end{array} \right. \tag{3.4.7}$$

$$g^{(1)}(\boldsymbol{z}) = 0.1\cos\left(\sum_{i=1}^{6} z_i\right) + \sum_{i=1}^{6} z_i^2, \quad \boldsymbol{z} \in \mathbb{R}^6, \tag{3.4.8}$$

$$\boldsymbol{g}^{(2)}(\boldsymbol{x}) = (\cos(\pi x_1 + 1), \cos(\pi x_2 + 2), \ldots, \cos(\pi x_6 + 6)) . \tag{3.4.9}$$

In the same manner as in Section 3.4.1, Figure 3.3 compares the evolution of errors $\varepsilon^2_{\text{NEST}}$ and $\varepsilon^2_{\text{LAR+UK}}$ with respect to n. As for the one-dimensional cases, it can be noticed in these figures that, for the studied examples, introducing a nested structure for the polynomial trend can allow us to make the L^2 error decrease by several orders of magnitude, especially when n is low. Moreover, these figures emphasize the interest of optimizing the values of the truncation parameter u_2 when dealing with multidimensional input spaces.

Note that for these examples, there is no information about the structure of the nested code. Adding some information about the relation between the inputs could be very useful to avoid testing too many values of p_1, p_2 and u_2.
As explained in Section 3.3.1, the values of p_2, p_1 and u_2 that were obtained from the convergence analyses can give many information about the unknown structure of the quantity of interest. For the first example, the values $p_2 = 2$, $u_2 = 2$ and $p_1 > 2$ were most of the time chosen, which is coherent with the fact that $g^{\text{2D}}(x_1, x_2)$ is just the product of two functions that depend on x_1 and x_2 only. Hence, a particular attention has to be paid to the modeling of each input, rather than to the modeling of the dependence structure.
In the same manner, for the second example, most of the convergence analyses lead us to $u_2 = 3$ and $p_2 < p_1$, which also shows that the modeling of each input seems to be more important than the characterization of the relation between these modified inputs.
At last, for the third quantity of interest, which is a highly oscillating function in dimension $d = 6$, the convergence analyses seemed to encourage the values of p_2 and p_1 that lead to



(a) $y(\boldsymbol{x}) = g^{2\mathrm{D}}(\boldsymbol{x})$

(b) $y(\boldsymbol{x}) = g^{3\mathrm{D}}(\boldsymbol{x})$

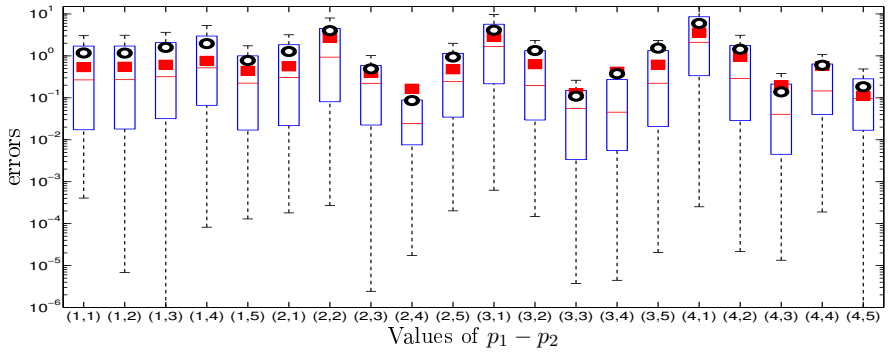
(c) $y(\boldsymbol{x}) = g^{6\mathrm{D}}(\boldsymbol{x})$

Figure 3.3: Evolution of the normalized L^2 errors with respect to n, the number of code evaluations. To be more representative, for each value of n, the LAR+UK and the proposed approaches have been repeated 10 times on randomly chosen learning sets. The curves correspond to the values of the 25% (thin line), the 50% (thick line) and the 75% (thin line) quantiles of the errors associated with these 10 repetitions. Solid black line: evolution of the error associated with the LAR+UK approach, $\varepsilon^2_{\mathrm{LAR+UK}}$. Blue dotted line: evolution of the error associated with the proposed approach, $\varepsilon^2_{\mathrm{NEST}}$, with $u_2 = 1$. Red dashed line: evolution of the error associated with the proposed approach, $\varepsilon^2_{\mathrm{NEST}}$, with $1 \leq u_2 \leq d$.

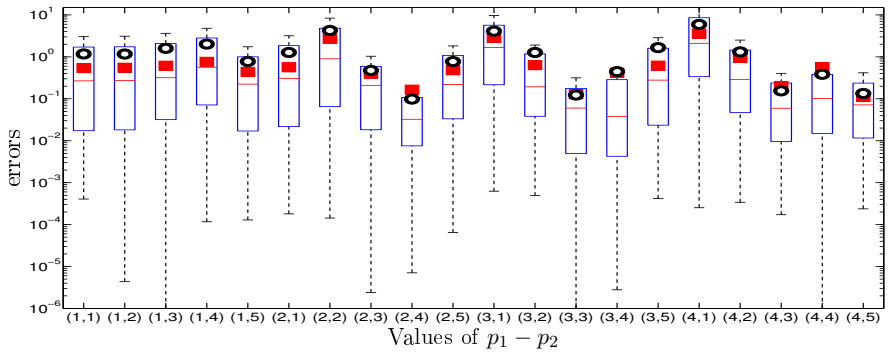
the highest product $p_1 \times p_2$ (before over-fitting). This means that, for this example, it is interesting to approximate quantity of interest y by a complex polynomial representation that is characterized by a small number of parameters.

3.4.3 Relevance of the LOO error

As presented in Section 3.3, when the maximal information about y is a set of code evaluations, the error $\left\|y - \widehat{y}^{\mathrm{nest}}\right\|_{\mathbb{X}}$ can be evaluated by its LOO approximation, $\varepsilon_{\mathrm{LOO}}$. In order to reduce the computational cost associated with the evaluation of $\varepsilon_{\mathrm{LOO}}$, two alternative estimations of error $\left\|y - \widehat{y}^{\mathrm{nest}}\right\|_{\mathbb{X}}$, $\widehat{\varepsilon}_{\mathrm{LOO}}$ and $\widetilde{\varepsilon}_{\mathrm{LOO}}$, have been proposed. In order to underline the relevance of these two LOO errors, Figure 3.4 compares these three errors in the case when $n = 100$ and y is the Ishigami function, whose expression is given by Eq. (3.4.6) (the same kinds of results would have been obtained for other values of n and other expressions of y). In this figure, it can thus be noticed that both approximations $\widehat{\varepsilon}_{\mathrm{LOO}}$ and $\widetilde{\varepsilon}_{\mathrm{LOO}}$ are very close to $\left\|y - \widehat{y}^{\mathrm{nest}}\right\|_{\mathbb{X}}$. In general, the approximation $\widetilde{\varepsilon}_{\mathrm{LOO}}$ is more conservative, in the sense that



(a) Case 1: $\widehat{\varepsilon}_{\mathrm{LOO}}$



(b) Case 2: $\widetilde{\varepsilon}_{\mathrm{LOO}}$

Figure 3.4: Comparisons between error $\left\|y - \widehat{y}^{\mathrm{nest}}\right\|_{\mathbb{X}}$ and its LOO approximations $\widehat{\varepsilon}_{\mathrm{LOO}}$ and $\widetilde{\varepsilon}_{\mathrm{LOO}}$ for the modeling of the Ishigami function from $n = 100$ code evaluations, for $u_2 = d$, $1 \leq p_2 \leq 4$ and $1 \leq p_1 \leq 5$. Red squares: the true values of $\left\|Y - \widehat{y}^{\mathrm{nest}}\right\|_{\mathbb{X}}$. Black circles: the approximated values. In each case, the box-plots correspond to the distributions of $(\widehat{e}_n^2,\ 1 \leq i \leq n)$ and $(\widetilde{e}_n^2,\ 1 \leq i \leq n)$, whose expressions are given by Eqs. (3.3.32) and (3.3.36).

it is less likely that it underestimates $\left\|y - \widehat{y}^{\mathrm{nest}}\right\|_{\mathbb{X}}$. However, as explained in Section 3.3, introducing a linearization around $\widehat{\boldsymbol{\beta}}_1$ reduces the risk of being too dependent on $\widehat{\boldsymbol{\beta}}_1$, which explains the fact that only small differences can be noticed between $\widehat{\varepsilon}_{\mathrm{LOO}}$ and $\widetilde{\varepsilon}_{\mathrm{LOO}}$.

3.5 Conclusions

One of the main objectives of this part was to propose an alternative parametrization of the polynomial trends for the Gaussian process regression. This parametrization, which is based on the composition of two polynomials, allows us to span high dimensional polynomial spaces with a reduced number of parameters. Hence, it has been shown on a series of examples that this approach can be very useful, especially when confronted to the modeling of complex functions with very little information.
In particular, this approach can allow us to find back (or take into account) a potential nested structure of the code.
However, identifying relevant values for these parameters is not easy. In this chapter, these parameters are identified from a two-steps approach. First, their maximum-likelihood estimates are searched from the resolution of the optimization problem. An iterative algorithm has been proposed to approximate the solutions of this problem. Then, a linearization around these values is carried out, in order to find back the usual formalism of GPR, and to minimize the sensitivity of the results to these values.
In spite of all these adaptations, when the input dimension becomes high ($d > 10$), and when a lot of code evaluations are available ($n > 100d$), it appears that the value of p_1 is often equal to 1. Such a value for p_1 corresponds to the "LAR+UK" configuration, which would mean that, in that case, the nested structure is not necessary. This can be due to the fact that the considered quantity of interest does not present a nested structure, or to the fact that the numerical complexity of the optimization problems associated with the nested representation is too high. Increasing the robustness of the proposed iterative algorithm, as well as proposing more efficient methods to solve the introduced optimization problems are thus possible extensions of the present chapter.
Trying to increase the sparsity of the proposed nested representation could also be a good idea, especially to enable the proposed method to deal with systems with higher values of d. Coupling the proposed nested representation to dedicated penalization techniques seems promising for future work.

Chapter 4

Gaussian process regression of two nested codes with scalar output

In this chapter, we focus on the case of two nested codes with scalar outputs. We now assume that observations of the intermediary variable are available. We therefore consider the following system:

$$\begin{array}{ccccc} & & \boldsymbol{x}_2 & \searrow & \\ & & & & y_{\text{nest}}(\boldsymbol{x}_{\text{nest}}) := y_2(y_1(\boldsymbol{x}_1), \boldsymbol{x}_2), \\ \boldsymbol{x}_1 & \rightarrow & y_1(\boldsymbol{x}_1) & \nearrow & \end{array} \tag{4.0.1}$$

where $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ and $\boldsymbol{x}_{\text{nest}}$ are low dimensional vectors and y_1, y_2 and y_{nest} are scalars.
The work presented in this chapter has been published in Marque-Pucheu et al. [2018]. The framework of the Gaussian process regression is considered (see Chapter 1 for further details). We propose an innovative Gaussian process based sequential design for the case of two nested code with scalar outputs.

4.1 Introduction

Thanks to computing power increase, the certification and the design of complex systems rely more and more on simulation. To this end, predictive codes are needed, which have generally to be evaluated at a large number of input points. When the computational cost of these codes is high, surrogate models are introduced to emulate their responses. A lot of industrial issues involve multi-physics phenomena, which can be associated with a series of computer codes. However, when these code networks are used for optimization, uncertainty quantification, or risk analysis purposes, they are generally considered a single code. In that case, all the inputs characterizing the system of interest are gathered in a single input vector, and little attention is paid to the potential intermediate results. When trying to emulate such code networks, this is clearly sub-optimal, as much information is lost in the statistical learning, so that too many evaluations of each code are likely to be required to get a satisfying prediction precision.
In this chapter, we focus on the case of two nested computer codes, where the output of the first code is one of the inputs of the second code. We assume that these two computer codes are deterministic, but expensive to evaluate. To predict the value of this nested code at an unobserved point, a Bayesian formalism [Robert, 2007] is adopted in the following. Each computer code is *a priori* modeled by a Gaussian process, and the idea is to identify the posterior distribution of the combination of these two processes given a limited number of evaluations of the two codes. The Gaussian process hypothesis is widely used in computer experiments ([Sacks et al., 1989; Santner et al., 2003; Rasmussen and Williams, 2006; Kennedy and O'Hagan, 2000, 2001; Berger et al., 2001; Paulo, 2005; Kleijnen, 2017]), as it allows a very

good trade-off between error control, complexity, and efficiency. The two main issues of this approach, also called Kriging, concern the choice of the statistical properties of the Gaussian processes that are used, and the choice of the points where to evaluate the codes. When a single computer code is considered, several methods exist to add one new point or a batch of new points sequentially to an already existing Design of Experiments. Depending on the purpose, optimization or reconstruction of the objective function on its whole input set, the criteria are based on the mean, variance or covariance of the predictor ([Sacks et al., 1989; Santner et al., 2003; Bect et al., 2012; Echard et al., 2011; Chevalier et al., 2014]). Given that our aim is to predict the output of the nested code on its whole input set, sequential designs based on a reduction of the integrated prediction variance (IMSE) are an appropriate choice. In the case of a single code, the variance expression can be explicitly derived under mild restrictive conditions on the mean and the covariance of the prior Gaussian distribution. The adaptation of these selection criteria to the case of two nested codes is not direct. Indeed, the combination of two Gaussian processes is not Gaussian, so that the prediction variance is much more complicated to estimate. The challenges posed by the composition of two Gaussian processes have been studied in the Deep Gaussian processes literature and the proposed methods are based on the Monte-Carlo computation of the likelihood of the nested Gaussian processes [Perdikaris et al., 2017] or on the computation of a lower bound of this likelihood [Damianou and Lawrence, 2013]. The composition of Gaussian processes can also be used in the multi-fidelity framework [Perdikaris et al., 2017]. This framework enables to use several levels of convergence of a simulator (for example in a finite element model a coarse mesh corresponds to the low fidelity simulator and the finer mesh corresponds to the high-fidelity simulator) and therefore to have a trade-off between accuracy and computation time [Kennedy and O'Hagan, 2000; Le Gratiet, 2013; Le Gratiet and Garnier, 2014; Picheny and Ginsbourger, 2013; Tuo et al., 2014].
Moreover, if the two codes can be launched separately, the selection criterion has also to indicate which one of the two codes to launch. The sequential designs are based on the prediction variance, which has to be computed at a large number of points. To reduce the computational cost associated with these computations, we propose several adaptations of the Gaussian Process formalism to the nested case. These adaptations make it possible to compute the two first statistical moments of the nested code output predictor exactly or quickly. Then, original sequential selection criteria are introduced, which try to exploit as much as possible the nested structure of the studied codes. In particular, these criteria are able to integrate the fact that the computational costs associated with the evaluation of each code can be different.
The outline of this chapter is the following. Section 4.2 presents the theoretical framework of the Gaussian process-based surrogate models, its generalization to the nested case, and introduces two selection criteria based on the prediction variance to reduce the prediction uncertainty sequentially. Section 4.3 introduces a series of simplifications to allow a quick computation of the prediction variance. In Section 4.4, the presented methods are applied to two examples.

The technical proofs of the results presented in the following sections are given in Section 4.6.

4.2 Surrogate modeling for two nested computer codes

4.2.1 General framework

Let $\mathcal{S}$ be a system which is characterized by a vector of input parameters, $\boldsymbol{x}_{\text{nest}} \in \mathbb{X}_{\text{nest}}$. Let $y_{\text{nest}} : \mathbb{X}_{\text{nest}} \to \mathbb{R}$ be a deterministic mapping that is used to analyze the studied system. In this chapter, we focus on the case where the function $\boldsymbol{x}_{\text{nest}} \mapsto y_{\text{nest}}(\boldsymbol{x}_{\text{nest}})$ can be modeled by

two nested codes. Two quantities of interest, y_1 and y_2, are thus introduced to characterize these two codes, which are supposed to be two real-valued continuous functions on their respective definition domains $\mathbb{X}_1$ and $\mathbb{R} \times \mathbb{X}_2$. Given these two functions, the nested code is defined as follows:

$$\begin{array}{ccccc} & & \boldsymbol{x}_2 \in \mathbb{X}_2 & & \\ & & & \searrow & \\ \boldsymbol{x}_1 \in \mathbb{X}_1 & \rightarrow & y_1(\boldsymbol{x}_1) \in \mathbb{R} & \nearrow & y_{\text{nest}}(\boldsymbol{x}_{\text{nest}}) := y_2(y_1(\boldsymbol{x}_1), \boldsymbol{x}_2) \in \mathbb{R}, \end{array} \tag{4.2.1}$$

where $\boldsymbol{x}_{\text{nest}} := (\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathbb{X}_{\text{nest}} = \mathbb{X}_1 \times \mathbb{X}_2$. The sets $\mathbb{X}_1$ and $\mathbb{X}_2$ are moreover supposed to be two compact subsets of $\mathbb{R}^{d_1}$ and $\mathbb{R}^{d_2}$ respectively, where d_1 and d_2 are two positive integers. In theory, the definition domains may be unbounded, but the reduction to compact sets enables the square integrability of y_{nest} on $\mathbb{X}_{\text{nest}}$. If they are unbounded, it is possible to introduce weighted L_2 spaces.
Given a limited number of evaluations of y_1 and y_2, the objective is to accurately predict y_{nest} on the whole input set.

4.2.2 Gaussian process-based surrogate models

4.2.2.1 Background

The Gaussian process regression (GPR), or Kriging, is a technique that is widely used to replace an expensive computer code by a surrogate model, that is to say a fast to evaluate mathematical function. The GPR is based on the assumption that the two code outputs, y_1 and y_2, can be seen as the sample paths of two stochastic processes, Y_1 and Y_2, which are supposed to be Gaussian for the sake of tractability:

$$Y_i(\cdot) \sim \text{GP}\left(\mu_i(\cdot), C_i(\cdot,\cdot)\right), \quad i \in \{1,2\}, \tag{4.2.2}$$

where for all $1 \leq i \leq 2$, μ_i and C_i denote respectively the mean and the covariance functions of Y_i.
Let $\overline{\boldsymbol{X}}_1^{\text{obs}} := \left(\bar{\boldsymbol{x}}_1^{(1)} = \boldsymbol{x}_1^{(1)}, \ldots, \bar{\boldsymbol{x}}_1^{(n_1)} = \boldsymbol{x}_1^{(n_1)}\right)$ be a $(n_1 \times d_1)$-dimensional matrix that gathers n_1 elements of $\mathbb{X}_1$ and $\overline{\boldsymbol{X}}_2^{\text{obs}} := \left(\bar{\boldsymbol{x}}_2^{(1)} = \left(\varphi_1^{(1)}, \boldsymbol{x}_2^{(1)}\right), \ldots, \bar{\boldsymbol{x}}_2^{(n_2)} = (\varphi_1^{(n_2)}, \boldsymbol{x}_2^{(n_2)})\right)$ be a $(n_2 \times d_2)$-dimensional matrix that gathers n_2 elements of $\mathbb{R} \times \mathbb{X}_2$. Denoting by

$$\boldsymbol{y}_1^{\text{obs}} := (y_1(\boldsymbol{x}_1^{(1)}), \ldots, y_1(\boldsymbol{x}_1^{(n_1)})), \text{ and } \boldsymbol{y}_2^{\text{obs}} := (y_2(\varphi_1^{(1)}, \boldsymbol{x}_2^{(1)}), \ldots, y_2(\varphi_1^{(n_2)}, \boldsymbol{x}_2^{(n_2)})), \tag{4.2.3}$$

the vectors that gather the evaluations of y_1 and y_2 at these points, it can be shown that:

$$Y_i^c(\cdot) := Y_i(\cdot) \mid \boldsymbol{y}_i^{\text{obs}} \sim \text{GP}\left(\mu_i^c(\cdot), C_i^c(\cdot,\cdot)\right), \tag{4.2.4}$$

and the detailed expressions of the conditioned mean functions, μ_i^c, and the conditioned covariance functions, C_i^c are presented in Eqs. (4.2.11) and (4.2.13) for the "Universal Kriging" framework. For further details on these expressions in other frameworks, the interested reader may refer to Section 1.4.
The relevance of the Gaussian process predictor strongly depends on the definitions of μ_i and C_i. When the only information about y_i is a finite set of evaluations, these functions are generally chosen in general parametric families. In this chapter, functions C_i are chosen in the squared exponential and Matérn-5/2 classes (see Section 1.4 for further details about classical parametric expressions for C_i).
The squared exponential class defines a parametric family of covariance functions that can be written in the form:

$$K_i\left(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_i'\right) = \exp\left(-d\left(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_i'\right)^2\right), \tag{4.2.5}$$

where:

$$\bar{\boldsymbol{x}}_i := \begin{cases} \boldsymbol{x}_1 \text{ if } i = 1, \\ (\varphi_1, \boldsymbol{x}_2) \text{ if } i = 2, \end{cases} \tag{4.2.6}$$

and $d\left(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_i^{'}\right) = \left\| \text{diag}\left(\boldsymbol{\ell}_i\right)^{-1} \left(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}}_i^{'}\right) \right\|$, $\text{diag}\left(\boldsymbol{\ell}_i\right)$ denotes a square matrix whose diagonal is equal to the vector $\boldsymbol{\ell}_i$ of correlation lengths.
Regarding the Matérn kernel, we consider the radial Matérn kernel, obtained by substituting the (weighted) Euclidean distance into the 1-dimensional Matérn kernel, and not the tensor product kernel obtained by multiplication of 1-dimensional kernels. So, the covariance functions of the Matérn $\frac{5}{2}$ class can be written in the form:

$$K_i\left(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_i^{'}\right) = \left(1 + \sqrt{5} d\left(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_i^{'}\right) + \frac{5}{3} d\left(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_i^{'}\right)^2\right) \exp\left(-\sqrt{5} d\left(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_i^{'}\right)\right). \tag{4.2.7}$$

Linear representations are considered for the mean functions:

$$\mu_i\left(\cdot\right) = \boldsymbol{h}_i\left(\cdot\right)^T \boldsymbol{\beta}_i, \tag{4.2.8}$$

where $\boldsymbol{h}_i$ is a given p_i-dimensional vector of functions (see Chapter 3 for further details on the choice of the basis functions). In the following, the framework of the "Universal Kriging" is adopted, which consists in:

- assuming an (improper) uniform distribution for $\boldsymbol{\beta}_i$,
- conditioning all the results by an estimator of the hyper-parameters that characterize the covariance functions C_i (obtained by cross-validation, as explained below),
- integrating over $\boldsymbol{\beta}_i$ the conditioned distribution of Y_i.

In that case, the distribution of Y_i^c, which is defined by Eq. (4.2.4), is Gaussian, and its statistical moments can explicitly be derived (see Sacks et al. [1989]; Bichon et al. [2008]; Helbert et al. [2009]; Bect et al. [2012]; Perrin et al. [2017]).
If we denote the posterior mean of $\widehat{\boldsymbol{\beta}}_i$ by:

$$\widehat{\boldsymbol{\beta}}_i := \left[\boldsymbol{h}_i\left(\overline{\boldsymbol{X}}_i^{\text{obs}}\right)\left(C_i\left(\overline{\boldsymbol{X}}_i^{\text{obs}}, \overline{\boldsymbol{X}}_i^{\text{obs}}\right)\right)^{-1} \boldsymbol{h}_i\left(\overline{\boldsymbol{X}}_i^{\text{obs}}\right)^T\right]^{-1} \boldsymbol{h}_i\left(\overline{\boldsymbol{X}}_i^{\text{obs}}\right)\left(C_i\left(\overline{\boldsymbol{X}}_i^{\text{obs}}, \overline{\boldsymbol{X}}_i^{\text{obs}}\right)\right)^{-1} \boldsymbol{y}_i^{\text{obs}}, \tag{4.2.9}$$

where $\boldsymbol{h}_i\left(\overline{\boldsymbol{X}}_i^{\text{obs}}\right)$ is a $(p_i \times n_i)$-dimensional matrix, whose j-th column is $\boldsymbol{h}_i\left(\bar{\boldsymbol{x}}_i^{(j)}\right)$, and $C_i\left(\overline{\boldsymbol{X}}_i^{\text{obs}}, \overline{\boldsymbol{X}}_i^{\text{obs}}\right)$ is a $(n_i \times n_i)$-dimensional matrix, such that:

$$\left(C_i\left(\overline{\boldsymbol{X}}_i^{\text{obs}}, \overline{\boldsymbol{X}}_i^{\text{obs}}\right)\right)_{jk} = C_i\left(\bar{\boldsymbol{x}}_i^{(j)}, \bar{\boldsymbol{x}}_i^{(k)}\right), \tag{4.2.10}$$

then the posterior prediction mean and variance can be written:

$$\mu_i^c\left(\bar{\boldsymbol{x}}_i\right) = \boldsymbol{h}_i\left(\bar{\boldsymbol{x}}_i\right)^T \widehat{\boldsymbol{\beta}}_i + C_i\left(\bar{\boldsymbol{x}}_i, \overline{\boldsymbol{X}}_i^{\text{obs}}\right)\left(C_i\left(\overline{\boldsymbol{X}}_i^{\text{obs}}, \overline{\boldsymbol{X}}_i^{\text{obs}}\right)\right)^{-1}\left[\boldsymbol{y}_i^{\text{obs}} - \boldsymbol{h}_i\left(\overline{\boldsymbol{X}}_i^{\text{obs}}\right)^T \widehat{\boldsymbol{\beta}}_i\right], \tag{4.2.11}$$

and:

$$\left(\sigma_i^c\left(\bar{\boldsymbol{x}}_i\right)\right)^2 = C_i^c\left(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_i\right), \tag{4.2.12}$$

$$
\begin{aligned}
C_i^c\left(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_i'\right) = \; & C_i\left(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_i'\right) - C_i\left(\bar{\boldsymbol{x}}_i, \overline{\boldsymbol{X}}_i^{\text{obs}}\right)\left(C_i\left(\overline{\boldsymbol{X}}_i^{\text{obs}}, \overline{\boldsymbol{X}}_i^{\text{obs}}\right)\right)^{-1} C_i\left(\overline{\boldsymbol{X}}_i^{\text{obs}}, \bar{\boldsymbol{x}}_i'\right) \\
& + \left[\boldsymbol{h}_i\left(\bar{\boldsymbol{x}}_i\right)^T - C_i\left(\bar{\boldsymbol{x}}_i, \overline{\boldsymbol{X}}_i^{\text{obs}}\right)\left(C_i\left(\overline{\boldsymbol{X}}_i^{\text{obs}}, \overline{\boldsymbol{X}}_i^{\text{obs}}\right)\right)^{-1} \boldsymbol{h}_i\left(\overline{\boldsymbol{X}}_i^{\text{obs}}\right)^T\right] \\
& \left[\boldsymbol{h}_i\left(\overline{\boldsymbol{X}}_i^{\text{obs}}\right)\left(C_i\left(\overline{\boldsymbol{X}}_i^{\text{obs}}, \overline{\boldsymbol{X}}_i^{\text{obs}}\right)\right)^{-1} \boldsymbol{h}_i\left(\overline{\boldsymbol{X}}_i^{\text{obs}}\right)^T\right]^{-1} \\
& \left[\boldsymbol{h}_i\left(\bar{\boldsymbol{x}}_i'\right) - \boldsymbol{h}_i\left(\overline{\boldsymbol{X}}_i^{\text{obs}}\right)\left(C_i\left(\overline{\boldsymbol{X}}_i^{\text{obs}}, \overline{\boldsymbol{X}}_i^{\text{obs}}\right)\right)^{-1} C_i\left(\overline{\boldsymbol{X}}_i^{\text{obs}}, \bar{\boldsymbol{x}}_i'\right)\right],
\end{aligned}
\tag{4.2.13}
$$

where $C_i\left(\bar{\boldsymbol{x}}_i, \overline{\boldsymbol{X}}_i^{\text{obs}}\right)$ is a n_i-dimensional vector and $\left(C_i\left(\bar{\boldsymbol{x}}_i, \overline{\boldsymbol{X}}_i^{\text{obs}}\right)\right)_k = C_i\left(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_i^{(k)}\right)$.

In this chapter, the hyperparameters of the covariance functions (see Section 1.4) are estimated for each set of observations by maximizing the Leave-One-Out log predictive probability (see Rasmussen and Williams [2006], Chapter 5, and Bachoc [2013a,b]).

4.2.2.2 Coupling the surrogate models of the two codes

According to Eq. (4.2.1), the nested code, $\boldsymbol{x}_{\text{nest}} \mapsto y_{\text{nest}}(\boldsymbol{x}_{\text{nest}})$, can thus be seen as a particular realization of the conditioned process Y_{nest}^c, so that for all $(\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathbb{X}_1 \times \mathbb{X}_2$,

$$
Y_{\text{nest}}^c(\boldsymbol{x}_1, \boldsymbol{x}_2) := Y_2^c(Y_1^c(\boldsymbol{x}_1), \boldsymbol{x}_2). \tag{4.2.14}
$$

Under this Gaussian formalism, the best prediction of y_{nest} at any unobserved point $\boldsymbol{x}_{\text{nest}} = (\boldsymbol{x}_1, \boldsymbol{x}_2)$ in $\mathbb{X}_1 \times \mathbb{X}_2$ is given by the mean value of $Y_{\text{nest}}^c(\boldsymbol{x}_1, \boldsymbol{x}_2)$, whereas its variance can be used to characterize the confidence in the prediction. As explained in Section 4.1, there is no reason for Y_{nest}^c to be Gaussian, but according to Proposition 4.2.1, the first- and second-order moments at a given input point can be obtained by computing two one-dimensional integrals with respect to a Gaussian measure.

Proposition 4.2.1. *For all $(\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathbb{X}_1 \times \mathbb{X}_2$, if $\xi \sim \mathcal{N}(0,1)$, then:*

$$
\mathbb{E}\left[Y_{nest}^c(\boldsymbol{x}_1, \boldsymbol{x}_2)\right] = \mathbb{E}\left[\mu_2^c(\mu_1^c(\boldsymbol{x}_1) + \sigma_1^c(\boldsymbol{x}_1)\xi, \boldsymbol{x}_2)\right], \tag{4.2.15}
$$

$$
\mathbb{E}\left[\left(Y_{nest}^c(\boldsymbol{x}_1, \boldsymbol{x}_2)\right)^2\right] = \mathbb{E}\left[\begin{array}{l}\left\{\mu_2^c(\mu_1^c(\boldsymbol{x}_1) + \sigma_1^c(\boldsymbol{x}_1)\xi, \boldsymbol{x}_2)\right\}^2 \\ + \left\{\sigma_2^c(\mu_1^c(\boldsymbol{x}_1) + \sigma_1^c(\boldsymbol{x}_1)\xi, \boldsymbol{x}_2)\right\}^2\end{array}\right]. \tag{4.2.16}
$$

The proof of this Proposition can be found in Section 4.6.

The computation of these moments can be done by quadrature rules or by Monte-Carlo methods ([Baker, 1977]). However, the computation time can be expensive, especially if the moments have to be computed at a large number of points.

Note that the proposed predictor for y_{nest} can be built using observations of y_1 or y_2 alone and not only observations of y_{nest}. It can take into account the partial information. If the two codes can be launched separately, this property will be particularly useful for the sequential enrichment of the initial design of experiments, since the variance of Y_{nest}^c can be reduced by evaluating y_1 or y_2 alone.

4.2.3 Sequential designs for the improvement of Gaussian process predictors

The relevance of the predictor Y^c_{nest} strongly depends on the space filling properties of the sets gathering the inputs of the available observations of y_1 and y_2, which are generally called Designs of Experiments (DoE). Space-filling Latin hypercube sampling (LHS) or quasi-Monte-Carlo sampling are generally chosen to define such *a priori* DoE ([Fang and Lin, 2003; Fang et al., 2006; Perrin and Cannamela, 2017]). The relevance of the predictor can then be improved by adding new points to an already existing DoE, as the higher the values of n_1 and n_2, the more chance there is for $\|\mathbb{E}\left[Y^c_{\text{nest}}\right] - y_{\text{nest}}\|^2_{\mathbb{X}_{\text{nest}}}$ to be small.

In the case of a single code, the existing selection criteria are based on the prediction variance [Sacks et al., 1989; Santner et al., 2003; Bect et al., 2012; Gramacy and Lian, 2012], the prediction mean [Hu and Ludkovski, 2017] or both [Echard et al., 2011] or the covariance between the observations [Sacks et al., 1989; Santner et al., 2003] and depend on the goal of the experiments: optimization, or reconstruction of the objective function on its whole input domain.
In this chapter the objective is to predict the output of the nested code on its whole input domain. So, a stepwise uncertainty reduction (SUR) [Chevalier et al., 2014] strategy is adopted in order to define criteria to add a new point. The proposed criteria are based on a minimization of the IMSE (integral of the prediction variance over the input domain) or on a maximization of the reduction of IMSE per unit of computational time. Some criteria that enable to take into account the different costs of several computer codes exist, for example in the multi-fidelity framework [Stroh et al., 2017] or multi-objective constraints [Perrin, 2016], but their adaptation to the case of two nested codes is not direct.
The use of IMSE is simplified by some properties of the Gaussian processes. Indeed, if Z is a Gaussian process that is indexed by $\boldsymbol{x}$ in $\mathbb{X}$, the variance of the conditioned random variable $Z(\boldsymbol{x}) \mid Z(\boldsymbol{x}^{\text{new}})$, where $\boldsymbol{x}$ and $\boldsymbol{x}^{\text{new}}$ are any elements of $\mathbb{X}$, does not depend on the (unknown) value of $Z(\boldsymbol{x}^{\text{new}})$. So, this variance can be denoted by abuse of notation $\mathbb{V}\left[Z(\boldsymbol{x}) \mid \boldsymbol{x}^{\text{new}}\right]$. To minimize the global uncertainty over Z at a reduced computational cost, a natural approach would consist in searching the value of $\boldsymbol{x}^{\text{new}}$ so that

$$\int_{\mathbb{X}} \mathbb{V}\left[Z(\boldsymbol{x}) \mid \boldsymbol{x}^{\text{new}}\right] d\boldsymbol{x} \tag{4.2.17}$$

is minimal (under the condition that this integral exists).

In the nested case, we also have to choose to which code to add a new observation point. To this end, let τ_1 and τ_2 be the numerical costs (in CPU time for instance) that are associated with the evaluations of y_1 and y_2 respectively. For the sake of simplicity, we assume that these numerical costs are independent on the value of the input parameters, and that they are *a priori* known. Two selection criteria are eventually proposed to optimize the relevance of the predictor of the nested code output sequentially. To simplify the reading, the following notation is proposed:

$$(\widetilde{\boldsymbol{x}}_i, \widetilde{\mathbb{X}}_i) := \begin{cases} (\boldsymbol{x}_1^*, \mathbb{X}_1) \text{ if } i = 1, \\ \left((\varphi_1^*, \boldsymbol{x}_2^*), \mu_1^c\left(\mathbb{X}_1\right) \times \mathbb{X}_2\right) \text{ if } i = 2, \\ \left((\boldsymbol{x}_1^*, \boldsymbol{x}_2^*), \mathbb{X}_1 \times \mathbb{X}_2\right) \text{ if } i = 3, \end{cases} \tag{4.2.18}$$

where $\boldsymbol{x}_1^* \in \mathbb{X}_1$, $\varphi_1^* \in \mu_1^c\left(\mathbb{X}_1\right)$ and $\boldsymbol{x}_2^* \in \mathbb{X}_2$ and we denote by $\mathbb{V}(Y^c_{\text{nest}}(\boldsymbol{x}_{\text{nest}})|\widetilde{\boldsymbol{x}}_i)$ the variance of $Y^c_{\text{nest}}(\boldsymbol{x}_{\text{nest}})$ under the hypothesis that the code(s) corresponding to the new point $\widetilde{\boldsymbol{x}}_i$ is (are) evaluated at this point (in practice, we remind that these code evaluations are not required

for the estimation of this variance). This variance can be defined by:

$$\mathbb{V}(Y^c_{\text{nest}}(\boldsymbol{x}_{\text{nest}})|\widetilde{\boldsymbol{x}}_i) := \begin{cases} \mathbb{V}(Y_2\left(Y_1\left(\boldsymbol{x}_1\right),\boldsymbol{x}_2\right)|\boldsymbol{y}_1^{\text{obs}},\boldsymbol{y}_2^{\text{obs}},y_i\left(\widetilde{\boldsymbol{x}}_i\right)),\ i\in\{1,2\}, \\ \mathbb{V}(Y_2\left(Y_1\left(\boldsymbol{x}_1\right),\boldsymbol{x}_2\right)|\boldsymbol{y}_1^{\text{obs}},\boldsymbol{y}_2^{\text{obs}},y_{\text{nest}}\left(\widetilde{\boldsymbol{x}}_i\right)),\ i=3, \end{cases} \tag{4.2.19}$$

with $\boldsymbol{x}_{\text{nest}} := (\boldsymbol{x}_1, \boldsymbol{x}_2)$.

- First, the chained I-optimal criterion selects the best point in $\mathbb{X}_1 \times \mathbb{X}_2$ to minimize the integrated variance of the predictor of the nested code:

$$\widetilde{\boldsymbol{x}}_3^{\text{new}} = \underset{\widetilde{\boldsymbol{x}}_3\in\widetilde{\mathbb{X}}_3}{\text{argmin}} \int_{\mathbb{X}_{\text{nest}}} \mathbb{V}(Y^c_{\text{nest}}(\boldsymbol{x}_{\text{nest}})|\widetilde{\boldsymbol{x}}_3)d\boldsymbol{x}_{\text{nest}}. \tag{4.2.20}$$

  Such a criterion is *a priori* adapted to the case where it is not possible to run independently the codes 1 and 2.

- Secondly, the best I-optimal criterion selects the best among the candidates in $\mathbb{X}_1$ and $\mathbb{X}_2$ in order to maximize the decrease per unit of computational cost of the integrated prediction variance of the nested code:

$$(i^{\text{new}}, \widetilde{\boldsymbol{x}}_{i^{\text{new}}}^{\text{new}}) = \underset{\widetilde{\boldsymbol{x}}_i\in\widetilde{\mathbb{X}}_i,\ i\in\{1,2\}}{\text{argmax}} \frac{1}{\tau_i} \times \int_{\mathbb{X}_{\text{nest}}} \left[\mathbb{V}\left(Y^c_{\text{nest}}(\boldsymbol{x}_{\text{nest}})\right) - \mathbb{V}\left(Y^c_{\text{nest}}(\boldsymbol{x}_{\text{nest}})|\widetilde{\boldsymbol{x}}_i\right)\right] d\boldsymbol{x}_{\text{nest}}. \tag{4.2.21}$$

  In that case, the difference in the computational costs is taken into account, and a linear expected improvement per unit of computational cost is assumed for the sake of simplicity.

For each new observation of the first code, the hyperparameters of the covariance function C_1 are re-estimated. In the same way, for each new observation of the second code, the hyperparameters of the covariance function C_2 are re-estimated.
An initial set of observations is necessary to estimate the hyperparameters of the covariance functions C_1 and C_2 and therefore to compute the prediction variance and the proposed sequential design criteria. This initial set will be chosen as a maximin LHS design on $\mathbb{X}_{\text{nest}}$.

4.3 Fast computation of the variance of the predictor of the nested code

As explained in Section 4.2.3, choosing the position of the new point requires to compute the value of $\text{Var}(Y^c_{\text{nest}}(\boldsymbol{x}_{\text{nest}})|\widetilde{\boldsymbol{x}}_i)$ for each potential value of $\widetilde{\boldsymbol{x}}_i$ in $\widetilde{\mathbb{X}}_i$ and for a grid or a sample of $\boldsymbol{x}_{\text{nest}}$ used in a quadrature formula or an empirical average to approximate the integral in $\boldsymbol{x}_{\text{nest}}$ of Eqs. (4.2.21) and (4.2.20).
For a given $\boldsymbol{x}_{\text{nest}}$, the variance is theoretically given by Eqs. (4.2.15) and (4.2.16). If a quadrature rule or a Monte Carlo approach is used to approximate the variance, then the optimization procedure becomes prohibitively expensive from the computational point of view. To circumvent this problem, we present in this section several approaches to make the computation of $\text{Var}(Y^c_{\text{nest}}(\boldsymbol{x}_{\text{nest}})|\widetilde{\boldsymbol{x}}_i)$ explicit, and therefore extremely fast to compute.

4.3.1 Explicit derivation of the two first statistical moments of the predictor

Lemma 4.3.1. *If* $X \sim \mathcal{N}(\mu, \sigma^2)$ *and* $g(x, a, b, c) := x^a \exp\left(bx + cx^2\right)$, $(a, b, c) \in \mathbb{N} \times \mathbb{R}^2$, *then, under the condition that* $1 - 2c\sigma^2 > 0$, *the mean of* $g(X, a, b, c)$ *can be computed analytically, and its expression is given by Eq.* (4.6.1).

Lemma 4.3.2. *If* $g(x, a, b, c) := x^a \exp\left(bx + cx^2\right)$, $(a, b, c) \in \mathbb{N} \times \mathbb{R}^2$, *then*

$$g(x, a_i, b_i, c_i)\, g(x, a_j, b_j, c_j) = g(x, a_i + a_j, b_i + b_j, c_i + c_j), \tag{4.3.1}$$

where $(a_i, b_i, c_i) \in \mathbb{N} \times \mathbb{R}^2$ *and* $(a_j, b_j, c_j) \in \mathbb{N} \times \mathbb{R}^2$.

Proposition 4.3.1. *Using the notations of the Universal Kriging framework that is introduced in Section 4.2.2, if:*

1. *for* $1 \leq k \leq p_2$ *the mean function* $(\boldsymbol{h}_2)_k$ *is of the form:*

$$(\boldsymbol{h}_2(\varphi_1, \boldsymbol{x}_2)_k = m_k(\boldsymbol{x}_2)\ \varphi_1^{a_k}, \tag{4.3.2}$$

   where m_k *is a deterministic function from* $\mathbb{X}_2$ *to* $\mathbb{R}$ *and* $a_k \in \mathbb{N}$,

2. *the covariance function* C_2 *is squared exponential, i.e. an element of the squared exponential class,*

then the conditional moments of order 1 and 2 of $Y^c_{nest}(\boldsymbol{x}_1, \boldsymbol{x}_2)$, *which are defined by Eqs.* (4.2.15) *and* (4.2.16) *can be calculated analytically using Lemmas 4.3.1 and 4.3.2. Moreover, the expression of the first order moment is given by Eqs.* (4.6.5) *and* (4.6.1) *and the one of the second order moment is given by Eqs.* (4.6.8) *and* (4.6.1).

The proof of this Proposition can be found in Section 4.6.

In other words, if the prior of the Gaussian process modeling the function y_2 has a trend which is a polynomial of φ_1, with coefficients as functions of $\boldsymbol{x}_2$, and a covariance function of the squared exponential class, then the moments of order 1 and 2 of the coupling of the predictors of the two codes can be computed explicitly.
In particular, if the process associated with y_2 has a constant or zero mean and a squared exponential (i.e. Gaussian) covariance, then the mean and the variance of the coupling of the predictors of y_1 and y_2 can be computed analytically.

However, the use of a squared exponential covariance function is based on the assumption of infinite differentiability of the second code. This assumption is not necessarily verified.
Besides, the method cannot be applied to the case of more than two codes. Indeed, in the case of three codes, the coupling of the Gaussian predictors of the two first codes is no longer Gaussian. Even if the Gaussian process modeling the third code has a squared exponential covariance and a polynomial trend with respect to the output of the second code, the analytical method cannot be applied because the predictor of the output of the chain of the two first codes is not Gaussian.

4.3.2 Linearized approach

In the cases where the conditions for Proposition 4.3.1 are not fulfilled (or if more than two codes are considered), another approach is proposed in this section, which is based on a linearization of the process modeling the nested code. Indeed, for $i \in \{1,2\}$, let ε_i^c be the Gaussian process so that:

$$Y_i^c = \mu_i^c + \varepsilon_i^c. \tag{4.3.3}$$

By construction, ε_i^c is the residual prediction uncertainty once Y_i has been conditioned by n_i evaluations of y_i. We remind that the two Gaussian processes Y_i are statistically independent, so Y_i^c and therefore ε_i^c are statistically independent. Under the condition that n_1 is large enough for Y_1^c being a reliable statistical model for y_1, then ε_1^c is small.

Proposition 4.3.2. *If:*

1. *the predictor of a nested computer code can be written $Y_{nest}^c(\boldsymbol{x}_1, \boldsymbol{x}_2) := Y_2^c(Y_1^c(\boldsymbol{x}_1), \boldsymbol{x}_2)$, where Y_i^c are independent Gaussian processes which can be written as $Y_i^c = \mu_i^c + \varepsilon_i^c$, where $\varepsilon_i^c \sim GP(0, C_i^c)\,,\ i \in \{1,2\}$,*
2. *and ε_1^c is small enough for the linearization to be valid,*

then the predictor of the nested computer code can be defined as a Gaussian process with the following mean and covariance functions:

$$\begin{aligned}
\mu_{nest}^c(\boldsymbol{x}_1, \boldsymbol{x}_2) &= \mu_2^c(\mu_1^c(\boldsymbol{x}_1), \boldsymbol{x}_2), \\
C_{nest}^c((\boldsymbol{x}_1, \boldsymbol{x}_2), (\boldsymbol{x}_1', \boldsymbol{x}_2')) &= C_2^c((\mu_1^c(\boldsymbol{x}_1), \boldsymbol{x}_2), (\mu_1^c(\boldsymbol{x}_1'), \boldsymbol{x}_2')) \\
&\quad + \frac{\partial \mu_2^c}{\partial \varphi_1}(\mu_1^c(\boldsymbol{x}_1), \boldsymbol{x}_2) \frac{\partial \mu_2^c}{\partial \varphi_1}(\mu_1^c(\boldsymbol{x}_1'), \boldsymbol{x}_2') C_1^c(\boldsymbol{x}_1, \boldsymbol{x}_1'),
\end{aligned} \tag{4.3.4}$$

where μ_i^c, $i \in 1,2$ is given by Eq. (4.2.11) and C_i^c, $i \in 1,2$ is given by Eq. (4.2.13) and $\dfrac{\partial \mu_2^c}{\partial \varphi_1}(\mu_1^c(\boldsymbol{x}_1), \boldsymbol{x}_2)$ is given by Eq. (4.6.13).
It can also be written that $Y_{nest}^c = \mu_{nest}^c + \varepsilon_{nest}^c$, with:

$$\varepsilon_{nest}^c(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{\partial \mu_2^c}{\partial \varphi_1}(\mu_1^c(\boldsymbol{x}_1), \boldsymbol{x}_2)\varepsilon_1^c(\boldsymbol{x}_1) + \varepsilon_2^c(\mu_1^c(\boldsymbol{x}_1), \boldsymbol{x}_2). \tag{4.3.5}$$

The proof of this Proposition can be found in Section 4.6.

Corollary 4.3.3. *In the framework of Universal Kriging for Y_1^c and Y_2^c with explicit basis functions $\boldsymbol{h}_i$ and covariance functions C_i, $i \in \{1,2\}$, if the derivatives $\dfrac{\partial \boldsymbol{h}_2}{\partial \varphi_1}(\varphi_1, \boldsymbol{x}_2)$ and $\dfrac{\partial C_2}{\partial \varphi_1}\left((\varphi_1, \boldsymbol{x}_2), \overline{\boldsymbol{X}}_2^{obs}\right)$ can be computed explicitly, then the predictor of the nested computer code can be defined, thanks to a linearization, as a Gaussian process with explicit mean and covariance functions. In particular, if the covariance function C_2 is in the Matérn $\dfrac{5}{2}$ or squared exponential classes, the derivative $\dfrac{\partial C_2}{\partial \varphi_1}\left((\varphi_1, \boldsymbol{x}_2), \overline{\boldsymbol{X}}_2^{obs}\right)$ can be computed analytically, and the associated expressions are given in Eqs. (4.6.18) and (4.6.21).*

The proof of this Corollary can be found in Section 4.6.

Corollary 4.3.4. *According to Eqs. (4.3.5), (4.2.21) and (4.2.20), if the predictor of the nested code is obtained with the linearized method, then, thanks to the independence between ε_1^c and ε_2^c, the selection criteria of the sequential designs can be written:*

- *for the chained I-optimal design:*

$$(\boldsymbol{x}_1^{new}, \boldsymbol{x}_2^{new}) = \underset{(\boldsymbol{x}_1^*, \boldsymbol{x}_2^*) \in \mathbb{X}_1 \times \mathbb{X}_2}{\operatorname{argmin}} \int_{\mathbb{X}_{nest}} \left(\frac{\partial \mu_2^c}{\partial \varphi_1}(\mu_1^c(\boldsymbol{x}_1), \boldsymbol{x}_2) \right)^2 \mathbb{V}\left[\varepsilon_1^c(\boldsymbol{x}_1) | \boldsymbol{x}_1^*\right] d\boldsymbol{x}_1 d\boldsymbol{x}_2, + \int_{\mathbb{X}_{nest}} \mathbb{V}\left[\varepsilon_2^c(\mu_1^c(\boldsymbol{x}_1), \boldsymbol{x}_2) | \mu_1^c(\boldsymbol{x}_1^*), \boldsymbol{x}_2^*\right] d\boldsymbol{x}_1 d\boldsymbol{x}_2, \tag{4.3.6}$$

  where $\dfrac{\partial \mu_2^c}{\partial \varphi_1}(\mu_1^c(\boldsymbol{x}_1), \boldsymbol{x}_2)$ *is given by Eq.* (4.6.13),

- *for the best I-optimal design:*

$$(i^{new}, \boldsymbol{x}_i^{new}) = \underset{\widetilde{\boldsymbol{x}}_i \in \widetilde{\mathbb{X}}_i,\ i \in \{1,2\}}{\operatorname{argmax}} \frac{1}{\tau_i} \mathcal{V}_i\left(\tilde{\boldsymbol{x}}_i\right), \tag{4.3.7}$$

where:

$$\mathcal{V}_1\left(\tilde{\boldsymbol{x}}_1\right) = \int_{\mathbb{X}_{nest}} \left(\frac{\partial \mu_2^c}{\partial \varphi_1}(\mu_1^c(\boldsymbol{x}_1), \boldsymbol{x}_2) \right)^2 \left(\mathbb{V}\left[\varepsilon_1^c(\boldsymbol{x}_1)\right] - \mathbb{V}\left[\varepsilon_1^c(\boldsymbol{x}_1) | \tilde{\boldsymbol{x}}_1\right] \right) d\boldsymbol{x}_1 d\boldsymbol{x}_2, \tag{4.3.8}$$

$$\mathcal{V}_2\left(\tilde{\boldsymbol{x}}_2\right) = \int_{\mathbb{X}_{nest}} \left(\mathbb{V}\left[\varepsilon_2^c(\mu_1^c(\boldsymbol{x}_1), \boldsymbol{x}_2)\right] - \mathbb{V}\left[\varepsilon_2^c(\mu_1^c(\boldsymbol{x}_1), \boldsymbol{x}_2) | \tilde{\boldsymbol{x}}_2\right] \right) d\boldsymbol{x}_1 d\boldsymbol{x}_2. \tag{4.3.9}$$

The proof of this Corollary can be found in Section 4.6.

Hence, thanks to the proposed linearization, and the fact that the conditional distribution of a Gaussian process is still Gaussian with updated first and second order moments, the variance of $Y_{\text{nest}}^c(\boldsymbol{x}_{\text{nest}})$ and the one of $Y_{\text{nest}}^c(\boldsymbol{x}_{\text{nest}})|\widetilde{\boldsymbol{x}}_i$ can be explicitly computed for all $(\boldsymbol{x}_{\text{nest}}, \widetilde{\boldsymbol{x}}_i)$ in $\mathbb{X}_{\text{nest}} \times \widetilde{\mathbb{X}}_i$. Under the condition that the linearization is valid, this approach can be applied to configurations with more than two nested codes.
However, it can be inferred from equation (4.3.4) that the variance depends on $\boldsymbol{y}_1^{\text{obs}}$ through μ_1^c and $\boldsymbol{y}_2^{\text{obs}}$ through μ_2^c. To circumvent this problem for the computation of the forward variance in the sequential designs, we assume that for a candidate $\tilde{\boldsymbol{x}}_1$, μ_1^c corresponds to $\mathbb{E}\left[Y_1 | \boldsymbol{y}_1^{\text{obs}}\right]$ and by abuse of notation, that $(\sigma_1^c)^2 = C_1^c$ corresponds to $\mathbb{V}\left[Y_1 | \overline{\boldsymbol{X}}_1^{\text{obs}}, \tilde{\boldsymbol{x}}_1\right]$. In the same way, for a candidate $\tilde{\boldsymbol{x}}_2$, we assume that μ_2^c corresponds to $\mathbb{E}\left[Y_2 | \boldsymbol{y}_2^{\text{obs}}\right]$ and by abuse of notation, that $(\sigma_2^c)^2 = C_2^c$ corresponds to $\mathbb{V}\left[Y_2 | \overline{\boldsymbol{X}}_2^{\text{obs}}, \tilde{\boldsymbol{x}}_2\right]$. So, by doing this, we suppose that the estimate of $y_i\left(\tilde{\boldsymbol{x}}_i\right)$ can be replaced by its prediction mean $\mathbb{E}\left[Y_i\left(\tilde{\boldsymbol{x}}_i\right) | \boldsymbol{y}_i^{\text{obs}}\right]$, in accordance with the Kriging Believer strategy proposed in Ginsbourger et al. [2010].

4.4 Applications

In this section, the proposed methods are applied to two examples: an analytical one-dimensional one and a multidimensional one.
In particular, the linearized method of Proposition 4.3.2 is compared with the analytical method of Proposition 4.3.1 in terms of prediction accuracy.
The linearized method is compared with the so-called "blind box" method. The blind box method corresponds to the case where the nested computer code is considered as a single computer code. In that case, only the inputs $\boldsymbol{x}_{\text{nest}}$ and the output y_{nest} are taken into account and a Gaussian process regression of this equivalent computer code is done. The intermediary information φ_1 is not taken into account. The Gaussian process Y_{bb} can therefore be defined as follows (see also Perrin et al. [2017]):

$$Y_{bb}(\cdot) \sim GP\left(\boldsymbol{h}_{bb}(\cdot)^T \boldsymbol{\beta}_{bb},\ C_{bb}(\cdot,\cdot)\right), \tag{4.4.1}$$

where

$$\boldsymbol{h}_{bb}(\boldsymbol{x}_1,\boldsymbol{x}_2) = \left(\frac{\partial \boldsymbol{h}_2}{\partial \varphi_1}\left(\boldsymbol{h}_1(\boldsymbol{x}_1)^T \boldsymbol{\beta}_1^*, \boldsymbol{x}_2\right)^T \boldsymbol{\beta}_2^* \boldsymbol{h}_1(\boldsymbol{x}_1), \boldsymbol{h}_2\left(\boldsymbol{h}_1(\boldsymbol{x}_1)^T \boldsymbol{\beta}_1^*, \boldsymbol{x}_2\right)\right), \tag{4.4.2}$$

$$\boldsymbol{\beta}_{bb} = (\boldsymbol{\beta}_1 - \boldsymbol{\beta}_1^*, \boldsymbol{\beta}_2), \tag{4.4.3}$$

$$(\boldsymbol{\beta}_1^*, \boldsymbol{\beta}_2^*) = \underset{(\boldsymbol{\beta}_1,\boldsymbol{\beta}_2)}{\operatorname{argmin}} \sum_{i=1}^{n} \left[y_2\left(y_1\left(\boldsymbol{x}_1^{(i)}\right), \boldsymbol{x}_2^{(i)}\right) - \boldsymbol{h}_2\left(\boldsymbol{h}_1\left(\boldsymbol{x}_1^{(i)}\right)^T \boldsymbol{\beta}_1, \boldsymbol{x}_2^{(i)}\right)^T \boldsymbol{\beta}_2 \right]^2, \tag{4.4.4}$$

$n = n_1 = n_2$ and C_{bb} is a stationary covariance function chosen in a parametric family and defined on $\mathbb{X}_{\text{nest}} \times \mathbb{X}_{\text{nest}}$. In order to make the comparison between the blind box and the other methods easier, the mean function is defined as a linearization of the coupling of the mean functions used in the linearized method.
Finally, the performances of the sequential designs are compared with a space filling design (maximin LHS) on $\mathbb{X}_{\text{nest}}$.

4.4.1 Characteristics of the examples

4.4.1.1 Analytical example

In the analytical example, the properties of the mean functions of the Gaussian processes and of the codes are:

$$\boldsymbol{h}_1(x_1) = \begin{bmatrix} 1 \\ x_1 \\ x_1^2 \end{bmatrix}, \quad \boldsymbol{\beta}_1 = \begin{bmatrix} -2 \\ 0.25 \\ 0.0625 \end{bmatrix}, \quad y_1(x_1) = \boldsymbol{h}_1(x_1)^T \boldsymbol{\beta}_1 - 0.25\cos(2\pi x_1), \tag{4.4.5}$$

$$\boldsymbol{h}_2(\varphi_1) = \begin{bmatrix} 1 \\ \varphi_1 \\ \varphi_1^2 \\ \varphi_1^3 \end{bmatrix}, \qquad \boldsymbol{\beta}_2 = \begin{bmatrix} 6 \\ -5 \\ -2 \\ 1 \end{bmatrix}, \qquad y_2(\varphi_1) = \boldsymbol{h}_2(\varphi_1)^T \boldsymbol{\beta}_2 - 0.25\cos(2\pi\varphi_1), \tag{4.4.6}$$

where $x_1 \in [-7, 7]$. In this example $\mathbb{X}_2 = \emptyset$.

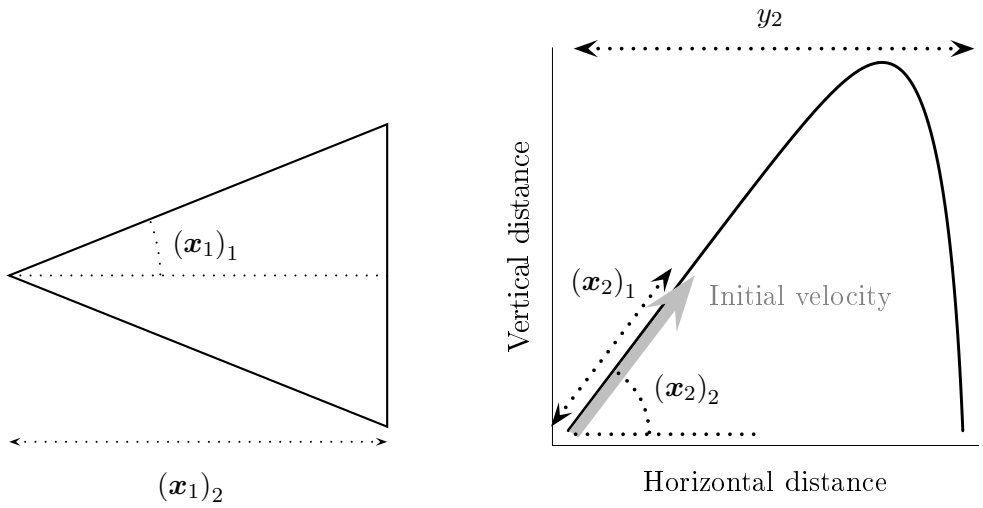
In the analytical example, the covariance functions are squared exponential (i.e. Gaussian). This implies that the Gaussian processes associated with the codes are mean square infinitely differentiable. This enables to apply Proposition 4.3.1 and Proposition 4.3.2 to this example.

4.4.1.2 Hydrodynamic example

In this example, the coupling of two computer codes is considered. The objective is to determine the impact point of a conical projectile.

The first code computes the drag coefficient of a cone divided by the height of the cone. Its inputs are the height and the half-angle of the cone, so the dimension of $\boldsymbol{x}_1$ is 2 and $\boldsymbol{x}_1 \in \left[\frac{\pi}{36}, \frac{\pi}{4}\right] \times [0.2, 2]$.

The second code computes the range of the ballistic trajectory of a cone. Its inputs are the output of the first code, associated with φ_1, and the initial velocity and angle of the



(a) Code 1: drag coefficient / height of the cone

(b) Code 2: range of a ballistic trajectory

Figure 4.1: Hydrodynamic example: Inputs and outputs of the two codes.

ballistic trajectory of the cone, gathered in $\boldsymbol{x}_2$. The dimension of $\boldsymbol{x}_2$ is therefore 2 and $\boldsymbol{x}_2 \in [1500, 3000] \times \left[\frac{\pi}{12}, \frac{7\pi}{36}\right]$.

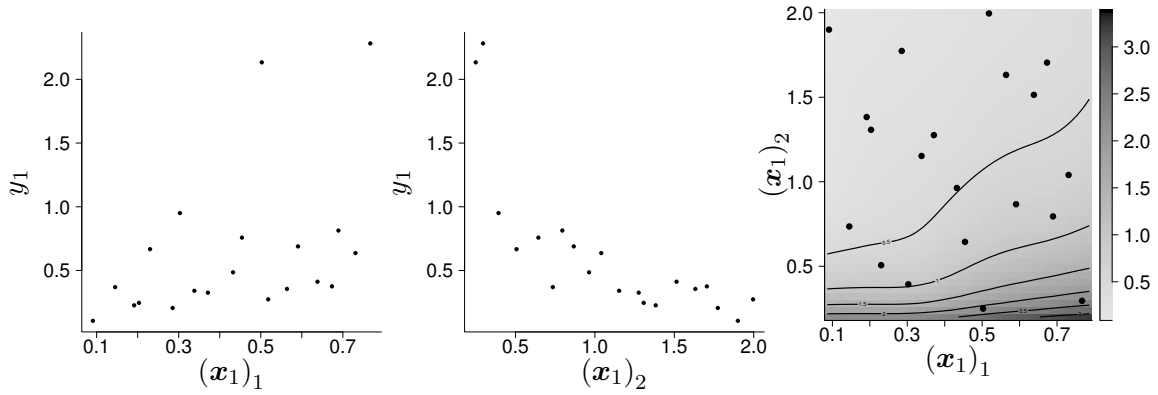
Figure 4.1 illustrates the two codes inputs and outputs.
Figure 4.2 presents, for each code, the scatter plots of the variations of the output with respect to the most sensitive components of their inputs. The inputs correspond to a set of 20 points drawn according to a maximin LHS design on $\mathbb{X}_{\text{nest}}$. These figures enable to propose a basis of functions for the prior mean of the processes associated with the two codes.
For the first code, the scatter plots highlight a linear variation with respect to $(\boldsymbol{x}_1)_1$ and a multiplicative inverse variation with respect to $(\boldsymbol{x}_1)_2$, so the proposed basis functions are:

$$\boldsymbol{h}_1(\boldsymbol{x}_1) = \left(1\ ,\ (\boldsymbol{x}_1)_1\ ,\ \frac{1}{(\boldsymbol{x}_1)_2}\right)^T. \tag{4.4.7}$$

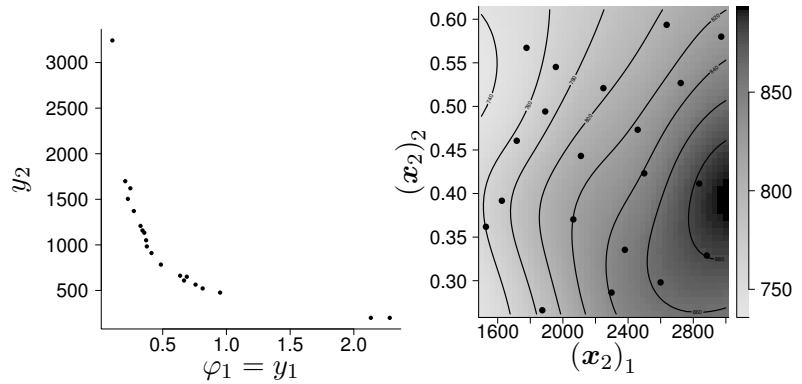
For the second code, only a multiplicative inverse variation with respect to φ_1 is evident, so the proposed basis functions are:

$$\boldsymbol{h}_2(\varphi_1, \boldsymbol{x}_2) = \left(\frac{1}{\max(\varphi_1, \varphi_{1_{\min}})},\ 1\ ,\ 1\right)^T. \tag{4.4.8}$$

The denominator has a lower bound $\varphi_{1_{\min}}$ in order to avoid any inversion problem around zero. $\varphi_{1_{\min}}$ is set to the small arbitrary value 0.1.
The image plot 4.2(c) represents the UK prediction mean of the first code, obtained with the proposed basis functions. The predicted value of y_1 for the maximum value of $(\boldsymbol{x}_1)_1$ and the minimum value of $(\boldsymbol{x}_1)_2$ is high compared with the values of the observations. So, the first code has been evaluated at this input point and gives the value of 3.4, which is consistent with the prediction. This illustrates the relevance of the proposed basis, that is used to extrapolate the prediction at a point with no observations around. The image plot 4.2(e) represents the



(a) Code 1 (b) Code 1 (c) UK prediction of code 1



(d) Code 2 (e) UK prediction of code 2 ($\varphi_1 = 0.5$)

Figure 4.2: Hydrodynamic example: variation of the outputs y_1 and y_2 of the two codes with respect to the most sensitive components of their inputs $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ for a set of 20 input points drawn according to a maximin LHS design on $\mathbb{X}_{\text{nest}}$. The image plots present the UK prediction (conditional mean of the GP) of y_1 and y_2 for the same set of observations.

UK prediction mean of the second code, obtained with the proposed basis at a value of 0.5 for φ_1.

In the hydrodynamic example, the covariance functions are in the Matérn $\frac{5}{2}$ class. This enables to perform the linearization of Proposition 4.3.2 and Corollary 4.3.3.

In both examples, the covariance functions include a non-zero nugget term (see Gramacy and Lee [2012] for further details), that means that they can be written as:

$$C_i\left(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_i^{'}\right) = \sigma_i^2\left[K_i\left(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_i^{'}\right) + g\delta_{\bar{\boldsymbol{x}}_i=\bar{\boldsymbol{x}}_i^{'}}\right], \tag{4.4.9}$$

where $\sigma_i \in \mathbb{R}_+$, K_i is chosen in a parametric family (squared exponential or Matérn $\frac{5}{2}$), g is the nugget term whose value is 10^{-6}, and δ is the Kronecker delta function. This non-zero nugget term is used for reasons of numerical stability.

4.4.2 Prediction performance for a given set of observations

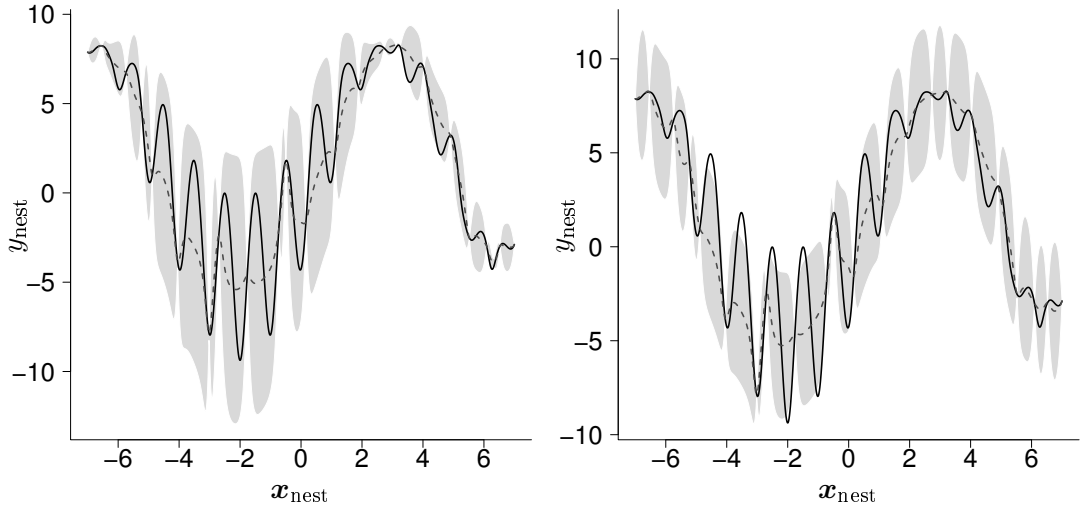
A set of validation observations is available. Let $\boldsymbol{x}_{\text{nest}}^{(1)} \dots \boldsymbol{x}_{\text{nest}}^{(N_{\text{test}})}$ be N_{test} elements of $\mathbb{X}_{\text{nest}}$. Denoting by $y_{\text{nest}}\left(\boldsymbol{x}_{\text{nest}}^{(1)}\right) \dots y_{\text{nest}}\left(\boldsymbol{x}_{\text{nest}}^{(N_{\text{test}})}\right)$ the evaluations of the nested code at these points, the performance criterion of the nested predictor mean, also called error on the mean can be defined as:

$$\text{Error on the mean} = \frac{\sum_{i=1}^{N_{\text{test}}}\left(y_{\text{nest}}\left(\boldsymbol{x}_{\text{nest}}^{(i)}\right) - \widehat{y}_{\text{nest}}\left(\boldsymbol{x}_{\text{nest}}^{(i)}\right)\right)^2}{\sum_{i=1}^{N_{\text{test}}}\left(y_{\text{nest}}\left(\boldsymbol{x}_{\text{nest}}^{(i)}\right) - \frac{1}{N_{\text{test}}}\sum_{j=1}^{N_{\text{test}}} y_{\text{nest}}\left(\boldsymbol{x}_{\text{nest}}^{(j)}\right)\right)^2}, \tag{4.4.10}$$

where $\widehat{y}_{\text{nest}}$ denotes a prediction of the nested code, which can be obtained with the analytical, linearized or blind-box method.
For both examples, the validation set of 150 points is drawn according to a maximin LHS on $\mathbb{X}_{\text{nest}}$.

Figure 4.3 presents, for the analytical example, an example of the prediction mean and 95% prediction interval computed with the linearized and the blind box methods. The two predictors are built with the same set of 20 observation points drawn according to a maximin LHS design on $\mathbb{X}_{\text{nest}}$. It can be seen that, with the blind box method, the magnitude of the prediction interval is the same across the input domain and depends only on the distance to the observation points. The prediction interval is too big in the area with small variations and too small in the area with larger variations. On the contrary, taking into account the intermediary observations (with the linearized method here) enables to better take into account the non-stationarity of the variations of the nested code output.

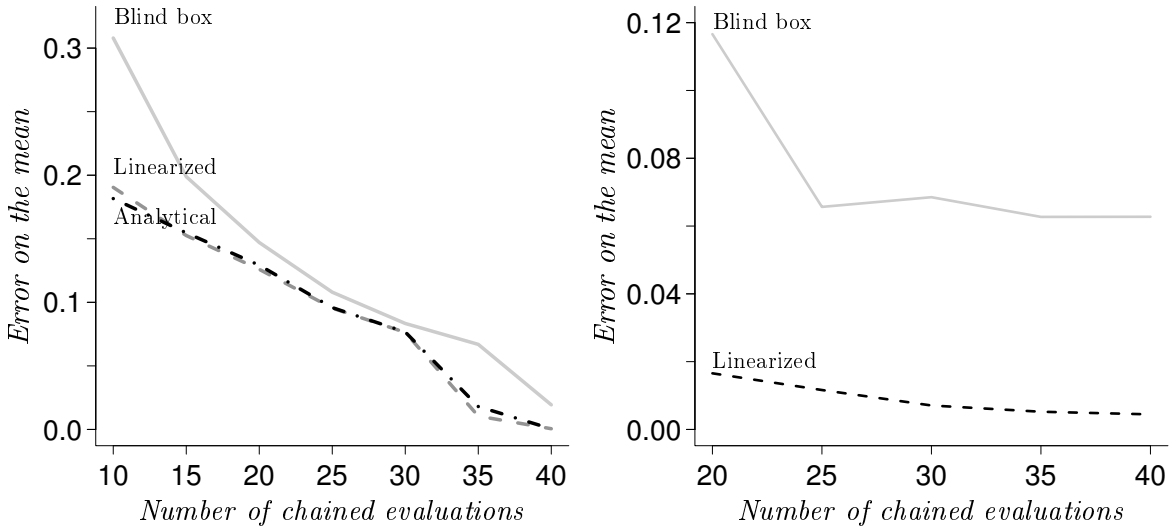
Figure 4.4 presents the error on the mean with the blind box and the linearized methods for both examples, and the analytical method for the analytical example. For all methods, the predictors are built with the same learning sets drawn according to maximin LHS designs on $\mathbb{X}_{\text{nest}}$ of increasing size.
The left figure, corresponding to the analytical example, shows the similar accuracies of the prediction means computed with the analytical and linearized methods proposed in Proposition 4.3.1 and Proposition 4.3.2.



(a) Linearized method

(b) Blind box method

Figure 4.3: Analytical example: Predictors of the nested code obtained with the linearized and the blind box methods. The set of 20 observations is drawn according to a maximin LHS on $\mathbb{X}_{\text{nest}}$. Actual values shown by a continuous line, the prediction mean by a dotted line and the 95% prediction interval by a grey area.



(a) Analytical: squared exponential covariance

(b) Hydrodynamic example: Matérn $\frac{5}{2}$ covariance

Figure 4.4: Comparison of the prediction mean accuracy for the blind box and the linearized (Proposition 4.3.2) methods, and, in the case of a squared exponential covariance function, the analytical method (Proposition 4.3.1). The curves correspond to the median of 50 draws of maximin LHS designs on $\mathbb{X}_1 \times \mathbb{X}_2$ of increasing size.

For both examples, the precision of the prediction mean is better with the linearized method than with the blind box method, showing the interest of taking into account the intermediary information.

The results show that the analytical and linearized methods lead to the same prediction mean accuracy. As a reminder, the analytical method requires the infinite differentiability of the second code. This assumption is correct for the analytical example but not necessarily for the hydrodynamic example. The linearized method requires the prediction error of the first code to be small enough for the linearization to be valid. Since the prediction error of the first code can be reduced thanks to a sequential enrichment of the initial design, the required assumption of the analytical method is stronger than the one of the linearized method.
Consequently, the linearized method will be used in the remainder of the numerical applications.

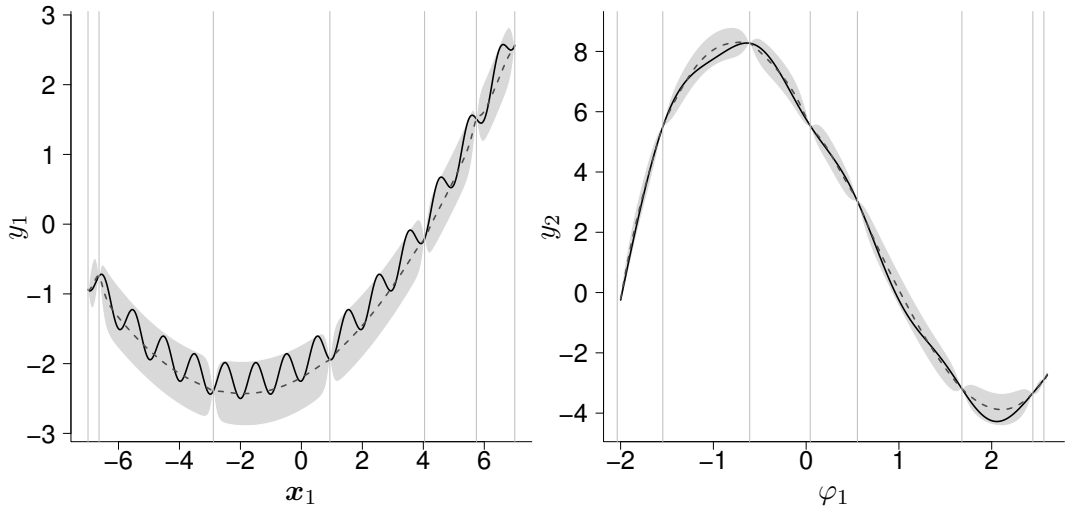
4.4.3 Performances of the sequential designs

Figure 4.5 shows an example of the prediction mean and 95% prediction interval of the predictors Y_1^c, Y_2^c and Y_{nest}^c. The predictors Y_1^c and Y_2^c are not built with the same number of observations, so the predictor Y_{nest}^c is built with a different number of observations of the codes 1 and 2. The fact that the number of observations of the two codes can be different will be useful for the sequential designs. Moreover, the estimation of the prediction variance of the nested code is accurate, and that will also be useful for the choice of the new observation point in the sequential designs.

4.4.3.1 With identical computational costs for both codes

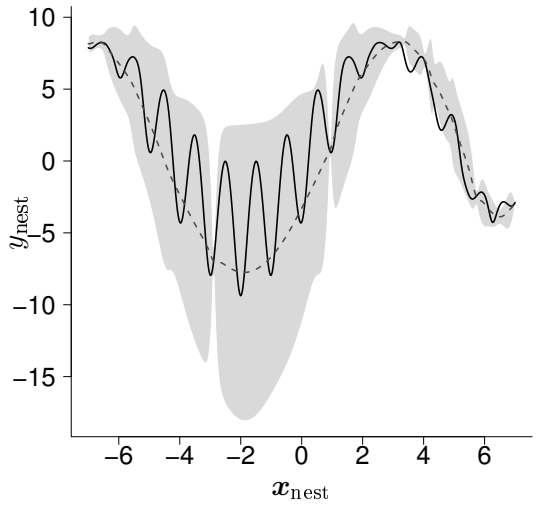
Figure 4.6 presents the error on the mean of the linearized predictor for the proposed sequential designs and for maximin LHS designs of increasing size. The initial designs of the sequential strategies are the same maximin LHS designs on $\mathbb{X}_{\text{nest}}$ with 10 points for the analytical example and 20 points for the hydrodynamic example. That is why the initial point of the three curves is the same on both line plots. The costs of the two codes are considered to be the same, that is to say $\tau_1 = \tau_2 = 1$. The figure shows the relevance of the proposed sequential designs for improving the prediction mean of the linearized nested predictor, compared with the maximin LHS designs on $\mathbb{X}_{\text{nest}}$.
In the analytical example, the best I-optimal sequential design enables to obtain the most accurate prediction mean at a given computational cost. In the hydrodynamic example, in the first 10 iterations, the best I-optimal design outperforms the chained I-optimal design. After this initial stage, the best I-optimal design calls alternately code 1 and code 2 and becomes equivalent to the chained I-optimal design.

Figure 4.7 shows to which of the two codes the new observations points are added for the best I-optimal sequential design. In both examples, new observation points of the first code are first added.
It seems that the uncertainty propagated from the first code into the second code is predominant at the beginning. The best I-optimal sequential design aims therefore at reducing this uncertainty by first adding new observation points of the first code. Then new observations of both codes are added.



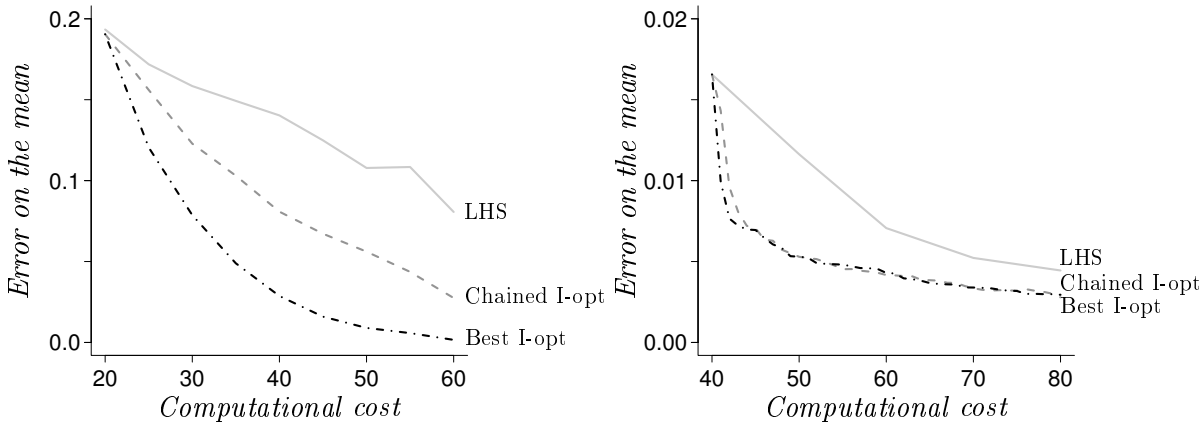
(a) Code 1

(b) Code 2



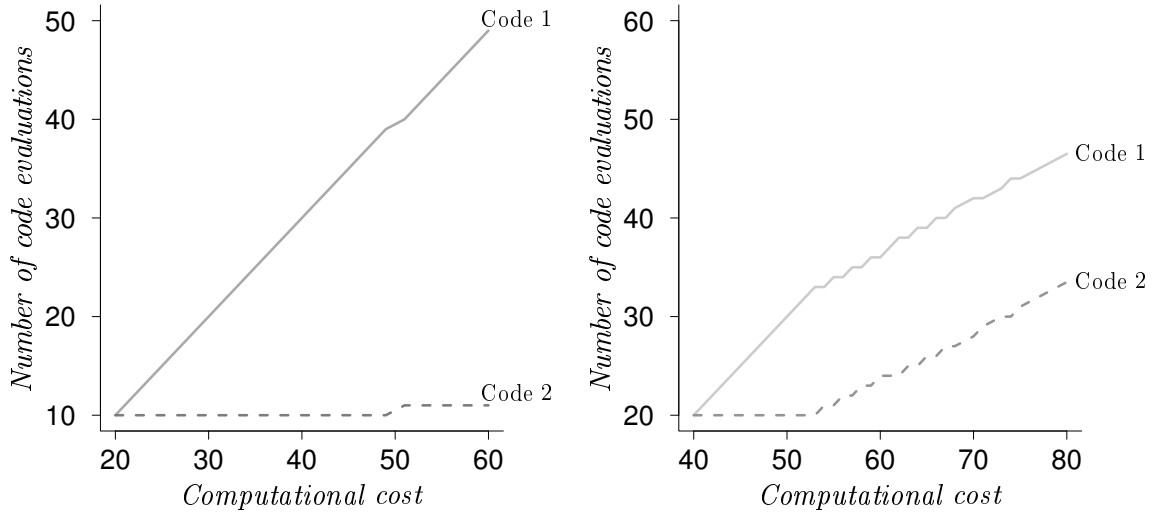
(c) Nested code

Figure 4.5: Analytical example: an example of the predictors Y_1^c, Y_2^c and Y_{nest}^c. The black line represents the real values of y_1, y_2 and y_{nest}, the grey area, the 95% prediction interval and the grey dotted line, the prediction mean. The mean and prediction interval of Y_{nest}^c are computed thanks to the linearized method. The vertical lines of the two top plots represent the observations of the two codes, which are drawn according to LHS designs on $\mathbb{X}_1$ and $\mu_1^c(\mathbb{X}_1)$ of sizes 7 and 8. The number of observations is not the same for both codes.



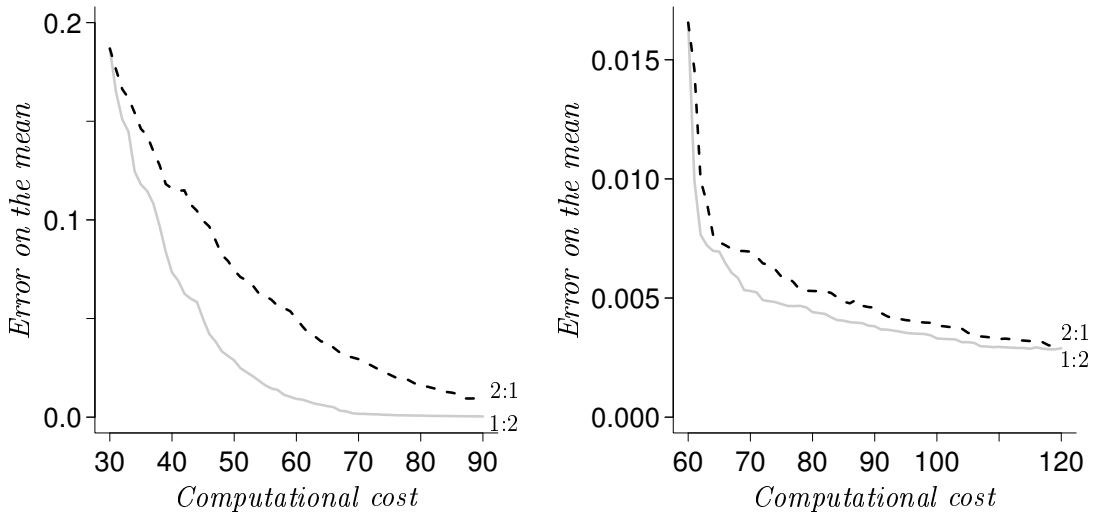
(a) Analytical example (b) Hydrodynamic example

Figure 4.6: Comparison of the prediction mean accuracy of the linearized predictor with the maximin LHS design on $\mathbb{X}_{\text{nest}}$ and the sequential designs, for both examples. In the hydrodynamic example, the two curves representing the sequential designs are almost superposed. The initial designs are the same for the three curves, with a size of 10 points for the analytical example and 20 points for the hydrodynamical example. The draw of the maximin LHS design on $\mathbb{X}_{\text{nest}}$ is repeated 50 times and the curves present the median of the associated results. The costs of the two codes are assumed to be the same.



(a) Analytical example (b) Hydrodynamic example

Figure 4.7: Comparison of the number of evaluations of each code in the case of a sequential best I-optimal design applied to both examples. The curves correspond to the median of 50 draws of the initial design. The costs of the two codes are assumed to be the same.



(a) Analytical example

(b) Hydrodynamic example

Figure 4.8: Performances of the best I-optimal sequential design in terms of prediction mean accuracy with different computational costs for the two codes. 1:2 $\leftrightarrow \tau_1 = 1$ and $\tau_2 = 2$, 2:1 $\leftrightarrow \tau_1 = 2$ and $\tau_2 = 1$. The curves correspond to the median of 50 draws of the initial maximin LHS design on $\mathbb{X}_{\text{nest}}$. The initial designs are the same for the two curves corresponding to each example and contain 15 observations and 30 observations on both codes for the analytical and the hydrodynamical example.

4.4.3.2 With different computational costs

Figure 4.8 shows the prediction mean accuracy with the best I-optimal sequential design when the costs of the two codes are different. Two cases are presented. The first one corresponds to the case where the cost associated with the first code is twice the one associated with the second code, that is to say $\tau_1 = 2$ and $\tau_2 = 1$, the second one corresponds to the case where the cost associated with the second code is twice the one associated with the first code, that is to say $\tau_1 = 1$ and $\tau_2 = 2$.
It can be seen that for both examples, the prediction accuracy at a given total computational cost is better when the cost of the first code is lower, that is to say when more observation points of the first code can be added for the same computational budget. These results are consistent with those of Figure 4.7.

4.5 Conclusions

In this chapter the formalism of Universal Kriging is adapted to the case of two nested computer codes.
Two methods to compute quickly the mean and variance of the nested code predictor have been proposed. The first one, called "analytical" computes the exact value of the two first moments of the predictor. But it cannot be applied to the coupling of more than two codes. The second one, called "linearized", enables to obtain a Gaussian predictor of the nested code, with mean and variance that can be instantly computed. The approach could be generalized to the coupling of more than two codes.
Both proposed methods take into account the intermediary information, that is to say the output of the first code. A comparison with the reference method, called "blind box", is made. In this method a Gaussian process regression of the block of the two codes is made without considering the intermediary observations. The numerical examples illustrate the interest of taking into account the intermediary information in terms of prediction mean accuracy.

Moreover, two sequential designs are proposed in order to improve the prediction accuracy of the nested predictor. The first one, the "chained" I-optimal sequential design, corresponds to the case where the two codes cannot be launched separately. The second one, the "best" I-optimal sequential design, allows to choose to which of the two codes to add a new observation point and to take into account the different computational costs of the two codes.
The numerical applications show the interest of the sequential designs compared with a space-filling design (maximin LHS). Furthermore, they illustrate the advantage, in terms of prediction mean accuracy, of choosing to which code to add a new observation point compared with simply adding new observation points of the nested code. The results show an amplification of the uncertainties in the chain of codes, leading to the addition of observation points of the first code firstly in the best I-optimal sequential design. It can be assumed that this should be similar with the coupling of more than two codes. In other words, the uncertainty of the beginning of the chain should be reduced as a priority.

4.6 Proofs

4.6.1 Proof of Proposition 4.2.1

According to Eq (4.2.4), one can write:

$$Y_i^c(\boldsymbol{x_i}) \overset{d}{=} \mu_i^c(\boldsymbol{x_i}) + \sigma_i^c(\boldsymbol{x_i})\,\xi_i, \quad \xi_i \sim \mathcal{N}(0,1), \quad i \in \{1,2\},$$

where ξ_1 and ξ_2 are independent according to the independence of the initial processes Y_1 and Y_2 and the fact that $Y_i^c := Y_i|\boldsymbol{y}_i^{\text{obs}}$.
Therefore, the process modeling the nested code can be written:

$$\begin{aligned} Y^c_{\text{nest}}(\boldsymbol{x}_1,\boldsymbol{x}_2) = &\quad Y_2^c(Y_1^c(\boldsymbol{x}_1),\boldsymbol{x}_2) \\ = &\quad \mu_2^c\left(\mu_1^c(\boldsymbol{x_1}) + \sigma_1^c(\boldsymbol{x_1})\,\xi_1, \boldsymbol{x_2}\right) + \sigma_2^c\left(\mu_1^c(\boldsymbol{x_1}) + \sigma_1^c(\boldsymbol{x_1})\,\xi_1, \boldsymbol{x_2}\right)\xi_2. \end{aligned}$$

Given the independence of ξ_1 and ξ_2 and the fact that $\mathbb{E}(\xi_2) = 0$, it can be inferred that the first moment of Y^c_{nest} can be written:

$$\mathbb{E}\left(Y^c_{\text{nest}}(\boldsymbol{x}_1,\boldsymbol{x}_2)\right) = \mathbb{E}\left(\mu_2^c\left(\mu_1^c(\boldsymbol{x_1}) + \sigma_1^c(\boldsymbol{x_1})\,\xi_1, \boldsymbol{x_2}\right)\right).$$

By noting that:

- $$\begin{aligned} \left(Y^c_{\text{nest}}(\boldsymbol{x}_1,\boldsymbol{x}_2)\right)^2 = &\quad \left(Y_2^c(Y_1^c(\boldsymbol{x}_1),\boldsymbol{x}_2)\right)^2 \\ = &\quad \left(\mu_2^c\left(\mu_1^c(\boldsymbol{x_1}) + \sigma_1^c(\boldsymbol{x_1})\,\xi_1, \boldsymbol{x_2}\right) + \sigma_2^c\left(\mu_1^c(\boldsymbol{x_1}) + \sigma_1^c(\boldsymbol{x_1})\,\xi_1, \boldsymbol{x_2}\right)\xi_2\right)^2 \\ = &\quad \left(\mu_2^c\left(\mu_1^c(\boldsymbol{x_1}) + \sigma_1^c(\boldsymbol{x_1})\,\xi_1, \boldsymbol{x_2}\right)\right)^2 + \left(\sigma_2^c\left(\mu_1^c(\boldsymbol{x_1}) + \sigma_1^c(\boldsymbol{x_1})\,\xi_1, \boldsymbol{x_2}\right)\right)^2 \xi_2^2 \\ &\quad + 2\mu_2^c\left(\mu_1^c(\boldsymbol{x_1}) + \sigma_1^c(\boldsymbol{x_1})\,\xi_1, \boldsymbol{x_2}\right)\sigma_2^c\left(\mu_1^c(\boldsymbol{x_1}) + \sigma_1^c(\boldsymbol{x_1})\,\xi_1, \boldsymbol{x_2}\right)\xi_2, \end{aligned}$$

- ξ_1 and ξ_2 are independent,
- $\mathbb{E}(\xi_2) = 0$ and $\mathbb{E}\left(\xi_2^2\right) = 1$,

the second moment of Y^c_{nest} can be written:

$$\mathbb{E}\left(\left(Y^c_{\text{nest}}(\boldsymbol{x}_1,\boldsymbol{x}_2)\right)^2\right) = \mathbb{E}\left[\begin{array}{l} \left(\mu_2^c\left(\mu_1^c(\boldsymbol{x_1}) + \sigma_1^c(\boldsymbol{x_1})\,\xi_1, \boldsymbol{x_2}\right)\right)^2 \\ + \left(\sigma_2^c\left(\mu_1^c(\boldsymbol{x_1}) + \sigma_1^c(\boldsymbol{x_1})\,\xi_1, \boldsymbol{x_2}\right)\right)^2 \end{array}\right].$$

4.6.2 Proof of Lemma 4.3.1

If $X \sim \mathcal{N}(\mu, \sigma^2)$ and $g(x,a,b,c) := x^a \exp\left[bx + cx^2\right]$, then the mean of $g(x,a,b,c)$ is equal to:

$$\mathbb{E}\left[g(X,a,b,c)\right] = \int_{\mathbb{R}} g(x,a,b,c)\,\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right) dx.$$

It follows that:

$$\begin{aligned}
\mathbb{E}\left[g\left(X,a,b,c\right)\right] = & \int_{\mathbb{R}} x^a \exp\left(bx+cx^2\right) \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right) dx \\
= & \exp\left(-\frac{1}{2\sigma^2}\left(\frac{\left(\sigma^2 b+\mu\right)^2}{2c\sigma^2-1}+\mu^2\right)\right) \\
& \times \int_{\mathbb{R}} x^a \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{1-2c\sigma^2}{\sigma^2}\left(x-\frac{\sigma^2 b+\mu}{1-2c\sigma^2}\right)^2\right) dx \\
= & \exp\left(-\frac{1}{2\sigma^2}\left(\frac{\left(\sigma^2 b+\mu\right)^2}{2c\sigma^2-1}+\mu^2\right)\right) \frac{1}{\sqrt{1-2c\sigma^2}} \mathbb{E}\left[X_g^a\right],
\end{aligned}$$

where $X_g \sim \mathcal{N}\left(\frac{\sigma^2 b+\mu}{1-2c\sigma^2}, \frac{\sigma^2}{1-2c\sigma^2}\right)$, under the condition that $1-2c\sigma^2 > 0$.
Moreover, for $Y \sim \mathcal{N}\left(\mu_Y, \sigma_Y^2\right)$, any moment of order k, $k \in \mathbb{N}$, of Y can be computed analytically ([Papoulis and Pillai, 2002]):

$$\mathbb{E}\left[Y^k\right] = \sum_{i=0}^{\lfloor \frac{k}{2} \rfloor} \binom{k}{2i} \mu_Y^{k-2i} \frac{(2i)!}{2^i i!} \sigma_Y^{2i}.$$

Hence, given that all the moments of a Gaussian variable can be computed analytically, the mean $\mathbb{E}\left[g\left(X,a,b,c\right)\right]$ can be computed analytically, and its expression is:

$$\begin{aligned}
\mathbb{E}\left[g\left(X,a,b,c\right)\right] = & \exp\left(-\frac{1}{2\sigma^2}\left(\frac{\left(\sigma^2 b+\mu\right)^2}{2c\sigma^2-1}+\mu^2\right)\right) \frac{1}{\sqrt{1-2c\sigma^2}} \\
& \times \sum_{i=0}^{\lfloor \frac{a}{2} \rfloor} \binom{a}{2i} \left(\frac{\sigma^2 b+\mu}{1-2c\sigma^2}\right)^{a-2i} \frac{(2i)!}{2^i i!} \left(\frac{\sigma^2}{1-2c\sigma^2}\right)^i.
\end{aligned} \tag{4.6.1}$$

4.6.3 Proof of Lemma 4.3.2

One has:

$$\begin{aligned}
g\left(x,a_i,b_i,c_i\right) g\left(x,a_j,b_j,c_j\right) = & \; x^{a_i} x^{a_j} \exp\left(b_i x + c_i x^2 + b_j x + c_j x^2\right) \\
= & \; x^{a_i+a_j} \exp\left((b_i+b_j)x + (c_i+c_j)x^2\right) \\
= & \; g\left(x, a_i+a_j, b_i+b_j, c_i+c_j\right).
\end{aligned}$$

4.6.4 Proof of Proposition 4.3.1

First moment

In the framework of Universal Kriging, according to equation (4.2.11) the conditional mean function of the process modeling the second code can be written:

$$\begin{aligned}\mu_2^c(\varphi_1, \boldsymbol{x}_2) = &\ \boldsymbol{h_2}(\varphi_1, \boldsymbol{x}_2)^T \widehat{\boldsymbol{\beta}}_2 + C_2\left((\varphi_1, \boldsymbol{x}_2), \bar{\boldsymbol{x}}_2^{\text{obs}}\right) \boldsymbol{v}_c \\ = &\ \sum_{i=1}^{p_2} \left(\boldsymbol{h_2}(\varphi_1, \boldsymbol{x}_2)\right)_i \left(\widehat{\boldsymbol{\beta}}_2\right)_i + \sum_{i=1}^{n_2} C_2\left((\varphi_1, \boldsymbol{x}_2), \left(\varphi_1^{(i)}, \boldsymbol{x}_2^{(i)}\right)\right) (\boldsymbol{v}_c)_i \\ = &\ (1) + (2),\end{aligned} \tag{4.6.2}$$

where $\varphi_1 \sim \mathcal{N}\left(\mu_1^c, (\sigma_1^c)^2\right)$, and

$$\boldsymbol{v}_c = \left(C_2\left(\overline{\boldsymbol{X}}_2^{\text{obs}}, \overline{\boldsymbol{X}}_2^{\text{obs}}\right)\right)^{-1} \left[\boldsymbol{y}_2^{\text{obs}} - \boldsymbol{h}_2\left(\overline{\boldsymbol{X}}_2^{\text{obs}}\right)^T \widehat{\boldsymbol{\beta}}_2\right]. \tag{4.6.3}$$

According to the assumptions of Proposition 4.3.1 the i-th, $i \in \{1, \ldots, p_2\}$, component of basis function $\boldsymbol{h}_2$ can be written:

$$\left(\boldsymbol{h}_2(\varphi_1, \boldsymbol{x}_2)\right)_i = m_i(\boldsymbol{x}_2)\ g(\varphi_1, a_i, 0, 0),$$

with m_i deterministic functions and $g(x, a, b, c) := x^a \exp\left(bx + cx^2\right),\ (a, b, c) \in \mathbb{N} \times \mathbb{R}^2$.
In the same way, the covariance function C_2 is in the squared exponential class, so according to Eq. (4.2.5), it can be written:

$$C_2\left((\varphi_1, \boldsymbol{x}_2), \left(\varphi_1', \boldsymbol{x}_2'\right)\right) = \sigma_2^2 k\left(\frac{\varphi_1 - \varphi_1'}{\ell_{\varphi_1}}\right) \prod_{i=1}^{d_2} k\left(\frac{(\boldsymbol{x}_2)_i - (\boldsymbol{x}_2')_i}{\ell_i}\right),$$

with $k : x \mapsto \exp\left(-x^2\right)$. So, one can write that:

$$\begin{aligned}C_2\left((\varphi_1, \boldsymbol{x}_2), \left(\varphi_1', \boldsymbol{x}_2'\right)\right) &= k\left(\frac{\varphi_1 - \varphi_1'}{\ell_{\varphi_1}}\right) \ell(\boldsymbol{x}_2 - \boldsymbol{x}_2'), \\ &= \exp\left(-\left(\frac{\varphi_1'}{\ell_{\varphi_1}}\right)^2\right) g\left(\varphi_1, 0, \frac{2\varphi_1'}{\ell_{\varphi_1}^2}, \frac{-1}{\ell_{\varphi_1}^2}\right) \ell(\boldsymbol{x}_2 - \boldsymbol{x}_2'),\end{aligned}$$

where ℓ is a deterministic function defined by:

$$\ell\left(\boldsymbol{x}_2 - \boldsymbol{x}_2'\right) = \sigma_2^2 \prod_{i=1}^{d_2} \exp\left(-\left(\frac{(\boldsymbol{x}_2)_i - (\boldsymbol{x}_2')_i}{\ell_i}\right)^2\right), \tag{4.6.4}$$

with $\ell_i, 1 \leq i \leq d_2$ the correlation lengths associated with $\boldsymbol{x}_2$.

So, the terms (1) and (2) of the equation (4.6.2) can be written:

$$(1) = \sum_{i=1}^{p_2} g(\varphi_1, a_i, 0, 0)\ m_i(\boldsymbol{x}_2) \left(\widehat{\boldsymbol{\beta}}_2\right)_i,$$

$$(2) = \sum_{i=1}^{n_2} (\boldsymbol{v}_c)_i\, \ell\left(\boldsymbol{x}_2 - \boldsymbol{x}_2^{(i)}\right) \exp\left(-\left(\frac{\varphi_1^{(i)}}{\ell_{\varphi_1}}\right)^2\right) g\left(\varphi_1, 0, \frac{2\varphi_1^{(i)}}{\ell_{\varphi_1}^2}, \frac{-1}{\ell_{\varphi_1}^2}\right).$$

Given that m_i and ℓ are deterministic functions, $\widehat{\boldsymbol{\beta}}_2$, $\boldsymbol{v}_c$, $\boldsymbol{x}_2^{(i)}$ and $\boldsymbol{x}_2$ are deterministic vectors, and the $\varphi_1^{(i)}$ are deterministic real numbers, one has:

$$\mathbb{E}\left[(1)\right] = \sum_{i=1}^{p_2} \mathbb{E}\left[g\left(\varphi_1, a_i, 0, 0\right)\right] \; m_i(\boldsymbol{x}_2) \left(\widehat{\boldsymbol{\beta}}_2\right)_i,$$

$$\mathbb{E}\left[(2)\right] = \sum_{i=1}^{n_2} \left(\boldsymbol{v}_c\right)_i \ell\left(\boldsymbol{x}_2 - \boldsymbol{x}_2^{(i)}\right) \exp\left(-\left(\frac{\varphi_1^{(i)}}{\ell_{\varphi_1}}\right)^2\right) \mathbb{E}\left[g\left(\varphi_1, 0, \frac{2\varphi_1^{(i)}}{\ell_{\varphi_1}^2}, \frac{-1}{\ell_{\varphi_1}^2}\right)\right].$$

According to Lemma 4.3.1, and the fact that $1 - 2\left(\frac{-1}{\ell_{\varphi_1}^2}\right)(\sigma_1^c)^2 > 0$, the means $\mathbb{E}\left[(1)\right]$ and $\mathbb{E}\left[(2)\right]$ can be calculated analytically, and consequently, the mean $\mathbb{E}\left[\mu_2^c\left(\varphi_1, \boldsymbol{x}_2\right)\right]$ can be calculated analytically, and its expression is:

$$\begin{aligned}\mathbb{E}\left[\mu_2^c\left(\varphi_1, \boldsymbol{x}_2\right)\right] = \; & \sum_{i=1}^{p_2} \mathbb{E}\left[g\left(\varphi_1, a_i, 0, 0\right)\right] \; m_i(\boldsymbol{x}_2) \left(\widehat{\boldsymbol{\beta}}_2\right)_i \\ & + \sum_{i=1}^{n_2} \left(\boldsymbol{v}_c\right)_i \ell\left(\boldsymbol{x}_2 - \boldsymbol{x}_2^{(i)}\right) \exp\left(-\left(\frac{\varphi_1^{(i)}}{\ell_{\varphi_1}}\right)^2\right) \mathbb{E}\left[g\left(\varphi_1, 0, \frac{2\varphi_1^{(i)}}{\ell_{\varphi_1}^2}, \frac{-1}{\ell_{\varphi_1}^2}\right)\right],\end{aligned} \tag{4.6.5}$$

where $\boldsymbol{v}_c$ is defined by Eq. (4.6.3), $\ell\left(\boldsymbol{x}_2 - \boldsymbol{x}_2'\right)$ is defined by Eq. (4.6.4), ℓ_{φ_1} is the correlation length associated with φ_1 and $\widehat{\boldsymbol{\beta}}_2$ is given by Eq. (4.2.9).

Second moment

From Eq. (4.2.11), (4.2.13) and (4.6.3), one has:

$$\mu_2^c\left(\varphi_1, \boldsymbol{x}_2\right) = \boldsymbol{h}_2\left(\varphi_1, \boldsymbol{x}_2\right)^T \widehat{\boldsymbol{\beta}}_2 + C_2\left(\left(\varphi_1, \boldsymbol{x}_2\right), \overline{\boldsymbol{X}}_2^{\text{obs}}\right) \boldsymbol{v}_c,$$

and:

$$\begin{aligned}\left(\sigma_2^c\left(\varphi_1, \boldsymbol{x}_2\right)\right)^2 = C_2\left(\left(\varphi_1, \boldsymbol{x}_2\right), \left(\varphi_1, \boldsymbol{x}_2\right)\right) - C_2\left(\left(\varphi_1, \boldsymbol{x}_2\right), \overline{\boldsymbol{X}}_2^{\text{obs}}\right) \left(\boldsymbol{R}_2^{\text{obs}}\right)^{-1} C_2\left(\overline{\boldsymbol{X}}_2^{\text{obs}}, \left(\varphi_1, \boldsymbol{x}_2\right)\right) + \\ \left[\boldsymbol{h}_2\left(\varphi_1, \boldsymbol{x}_2\right)^T - C_2\left(\left(\varphi_1, \boldsymbol{x}_2\right), \overline{\boldsymbol{X}}_2^{\text{obs}}\right) \left(\boldsymbol{R}_2^{\text{obs}}\right)^{-1} \boldsymbol{h}_2\left(\overline{\boldsymbol{X}}_2^{\text{obs}}\right)^T\right] \left[\boldsymbol{h}_2\left(\overline{\boldsymbol{X}}_2^{\text{obs}}\right) \left(\boldsymbol{R}_2^{\text{obs}}\right)^{-1} \boldsymbol{h}_2\left(\overline{\boldsymbol{X}}_2^{\text{obs}}\right)^T\right]^{-1} \\ \left[\boldsymbol{h}_2\left(\varphi_1, \boldsymbol{x}_2\right) - \boldsymbol{h}_2\left(\overline{\boldsymbol{X}}_2^{\text{obs}}\right) \left(\boldsymbol{R}_2^{\text{obs}}\right)^{-1} C_2\left(\overline{\boldsymbol{X}}_2^{\text{obs}}, \left(\varphi_1, \boldsymbol{x}_2\right)\right)\right],\end{aligned}$$

where $\boldsymbol{R}_2^{\text{obs}} = C_2\left(\overline{\boldsymbol{X}}_2^{\text{obs}}, \overline{\boldsymbol{X}}_2^{\text{obs}}\right)$.
Hence, it can be written that:

$$\begin{aligned}\left(\mu_2^c\left(\varphi_1, \boldsymbol{x}_2\right)\right)^2 + \left(\sigma_2^c\left(\varphi_1, \boldsymbol{x}_2\right)\right)^2 = \; & \sigma_2^2 + \underbrace{\boldsymbol{h_2}\left(\varphi_1, \boldsymbol{x}_2\right)^T \boldsymbol{A}_h \boldsymbol{h_2}\left(\varphi_1, \boldsymbol{x}_2\right)}_{(1)} \\ & + \underbrace{C_2\left(\left(\varphi_1, \boldsymbol{x}_2\right), \bar{\boldsymbol{x}}_2^{\text{obs}}\right) \boldsymbol{A}_c \; C_2\left(\bar{\boldsymbol{x}}_2^{\text{obs}}, \left(\varphi_1, \boldsymbol{x}_2\right)\right)}_{(2)} \\ & + \underbrace{C_2\left(\left(\varphi_1, \boldsymbol{x}_2\right), \bar{\boldsymbol{x}}_2^{\text{obs}}\right) \boldsymbol{A}_{ch} \; \boldsymbol{h_2}\left(\varphi_1, \boldsymbol{x}_2\right)}_{(3)},\end{aligned} \tag{4.6.6}$$

where:

$$\begin{aligned}
\boldsymbol{A}_h = &\ \widehat{\boldsymbol{\beta}}_2\widehat{\boldsymbol{\beta}}_2^T + \left(\boldsymbol{h}_2\left(\overline{\boldsymbol{X}}_2^{\text{obs}}\right)\left(\boldsymbol{R}_2^{\text{obs}}\right)^{-1}\boldsymbol{h}_2\left(\overline{\boldsymbol{X}}_2^{\text{obs}}\right)^T\right)^{-1}, \\
\boldsymbol{A}_c = &\ \boldsymbol{v}_c\boldsymbol{v}_c^T - \left(\boldsymbol{R}_2^{\text{obs}}\right)^{-1} + \left(\boldsymbol{R}_2^{\text{obs}}\right)^{-1}\boldsymbol{h}_2\left(\overline{\boldsymbol{X}}_2^{\text{obs}}\right)^T\left[\boldsymbol{h}_2\left(\overline{\boldsymbol{X}}_2^{\text{obs}}\right)\left(\boldsymbol{R}_2^{\text{obs}}\right)^{-1}\boldsymbol{h}_2\left(\overline{\boldsymbol{X}}_2^{\text{obs}}\right)^T\right]^{-1} \\
&\ \boldsymbol{h}_2\left(\overline{\boldsymbol{X}}_2^{\text{obs}}\right)\left(\boldsymbol{R}_2^{\text{obs}}\right)^{-1}, \\
\boldsymbol{A}_{ch} = &\ 2\boldsymbol{v}_c\widehat{\boldsymbol{\beta}}_2^T - 2\left(\boldsymbol{R}_2^{\text{obs}}\right)^{-1}\boldsymbol{h}_2\left(\overline{\boldsymbol{X}}_2^{\text{obs}}\right)^T\left[\boldsymbol{h}_2\left(\overline{\boldsymbol{X}}_2^{\text{obs}}\right)\left(\boldsymbol{R}_2^{\text{obs}}\right)^{-1}\boldsymbol{h}_2\left(\overline{\boldsymbol{X}}_2^{\text{obs}}\right)^T\right]^{-1}.
\end{aligned} \tag{4.6.7}$$

According to the assumptions of Proposition 4.3.1 and to lemma 4.3.2, the terms (1), (2) and (3) of the equation (4.6.6) can be rewritten:

$$\begin{aligned}
(1) = &\ \sum_{i=1}^{p_2}\sum_{j=1}^{p_2}\left(\boldsymbol{A}_h\right)_{ij}\left(\boldsymbol{h_2}\left(\varphi_1,\boldsymbol{x}_2\right)\right)_i\left(\boldsymbol{h_2}\left(\varphi_1,\boldsymbol{x}_2\right)\right)_j \\
= &\ \sum_{i=1}^{p_2}\sum_{j=1}^{p_2}\left(\boldsymbol{A}_h\right)_{ij} m_i\left(\boldsymbol{x}_2\right) m_j\left(\boldsymbol{x}_2\right) g\left(\varphi_1,a_i,0,0\right) g\left(\varphi_1,a_j,0,0\right) \\
= &\ \sum_{i=1}^{p_2}\sum_{j=1}^{p_2}\left(\boldsymbol{A}_h\right)_{ij} m_i\left(\boldsymbol{x}_2\right) m_j\left(\boldsymbol{x}_2\right) g\left(\varphi_1,a_i+a_j,0,0\right),
\end{aligned}$$

$$\begin{aligned}
(2) = &\ \sum_{i=1}^{n_2}\sum_{j=1}^{n_2}\left(\boldsymbol{A}_c\right)_{ij} C_2\left(\left(\varphi_1,\boldsymbol{x}_2\right),\left(\varphi_1^{(i)},\boldsymbol{x}_2^{(i)}\right)\right) C_2\left(\left(\varphi_1,\boldsymbol{x}_2\right),\left(\varphi_1^{(j)},\boldsymbol{x}_2^{(j)}\right)\right) \\
= &\ \sum_{i=1}^{n_2}\sum_{j=1}^{n_2}\left(\boldsymbol{A}_c\right)_{ij}\ell\left(\boldsymbol{x}_2-\boldsymbol{x}_2^{(i)}\right)\ell\left(\boldsymbol{x}_2-\boldsymbol{x}_2^{(j)}\right)\exp\left(-\frac{\left(\varphi_1^{(i)}\right)^2+\left(\varphi_1^{(j)}\right)^2}{\ell_{\varphi_1}^2}\right) \\
&\ \times g\left(\varphi_1,0,\frac{2\varphi_1^{(i)}}{\ell_{\varphi_1}^2},\frac{-1}{\ell_{\varphi_1}^2}\right) g\left(\varphi_1,0,\frac{2\varphi_1^{(j)}}{\ell_{\varphi_1}^2},\frac{-1}{\ell_{\varphi_1}^2}\right) \\
= &\ \sum_{i=1}^{n_2}\sum_{j=1}^{n_2}\left(\boldsymbol{A}_c\right)_{ij}\ell\left(\boldsymbol{x}_2-\boldsymbol{x}_2^{(i)}\right)\ell\left(\boldsymbol{x}_2-\boldsymbol{x}_2^{(j)}\right)\exp\left(-\frac{\left(\varphi_1^{(i)}\right)^2+\left(\varphi_1^{(j)}\right)^2}{\ell_{\varphi_1}^2}\right) \\
&\ \times g\left(\varphi_1,0,2\frac{\varphi_1^{(i)}+\varphi_1^{(j)}}{\ell_{\varphi_1}^2},\frac{-2}{\ell_{\varphi_1}^2}\right),
\end{aligned}$$

$$\begin{aligned}
(3) = &\ \sum_{i=1}^{n_2}\sum_{j=1}^{p_2}\left(\boldsymbol{A}_{ch}\right)_{ij} C_2\left(\left(\varphi_1,\boldsymbol{x}_2\right),\left(\varphi_1^{(i)},\boldsymbol{x}_2^{(i)}\right)\right)\left(\boldsymbol{h_2}\left(\varphi_1,\boldsymbol{x}_2\right)\right)_j \\
= &\ \sum_{i=1}^{n_2}\sum_{j=1}^{p_2}\left(\boldsymbol{A}_{ch}\right)_{ij}\ell\left(\boldsymbol{x}_2-\boldsymbol{x}_2^{(i)}\right)\exp\left(-\left(\frac{\varphi_1^{(i)}}{\ell_{\varphi_1}}\right)^2\right) m_j\left(\boldsymbol{x}_2\right) g\left(\varphi_1,0,\frac{2\varphi_1^{(i)}}{\ell_{\varphi_1}^2},\frac{-1}{\ell_{\varphi_1}^2}\right) \\
&\ \times g\left(\varphi_1,a_j,0,0\right) \\
= &\ \sum_{i=1}^{n_2}\sum_{j=1}^{p_2}\left(\boldsymbol{A}_{ch}\right)_{ij}\ell\left(\boldsymbol{x}_2-\boldsymbol{x}_2^{(i)}\right)\exp\left(-\left(\frac{\varphi_1^{(i)}}{\ell_{\varphi_1}}\right)^2\right) m_j\left(\boldsymbol{x}_2\right) g\left(\varphi_1,a_j,\frac{2\varphi_1^{(i)}}{\ell_{\varphi_1}^2},\frac{-1}{\ell_{\varphi_1}^2}\right).
\end{aligned}$$

Given that m_i and ℓ are deterministic functions, $\boldsymbol{x}_2$ and $\boldsymbol{x}_2^{(i)}$ are deterministic vectors, $\boldsymbol{A}_h$, $\boldsymbol{A}_c$ and $\boldsymbol{A}_{ch}$ deterministic matrices, and $\varphi_1^{(i)}$ and ℓ_{φ_1} are deterministic real numbers, one can write:

$$\mathbb{E}\left[(1)\right] = \sum_{i=1}^{p_2}\sum_{j=1}^{p_2} \left(\boldsymbol{A}_h\right)_{ij} m_i\left(\boldsymbol{x}_2\right) m_j\left(\boldsymbol{x}_2\right) \mathbb{E}\left[g\left(\varphi_1, a_i + a_j, 0, 0\right)\right],$$

$$\mathbb{E}\left[(2)\right] = \sum_{i=1}^{n_2}\sum_{j=1}^{n_2} \left(\boldsymbol{A}_c\right)_{ij} \ell\left(\boldsymbol{x}_2 - \boldsymbol{x}_2^{(i)}\right) \ell\left(\boldsymbol{x}_2 - \boldsymbol{x}_2^{(j)}\right) \exp\left(-\frac{\left(\varphi_1^{(i)}\right)^2 + \left(\varphi_1^{(j)}\right)^2}{\ell_{\varphi_1}^2}\right) \mathbb{E}\left[g\left(\varphi_1, 0, 2\frac{\varphi_1^{(i)} + \varphi_1^{(j)}}{\ell_{\varphi_1}^2}, \frac{-2}{\ell_{\varphi_1}^2}\right)\right],$$

$$\mathbb{E}\left[(3)\right] = \sum_{i=1}^{n_2}\sum_{j=1}^{p_2} \left(\boldsymbol{A}_{ch}\right)_{ij} \ell\left(\boldsymbol{x}_2 - \boldsymbol{x}_2^{(i)}\right) \exp\left(-\left(\frac{\varphi_1^{(i)}}{\ell_{\varphi_1}}\right)^2\right) m_j\left(\boldsymbol{x}_2\right) \mathbb{E}\left[g\left(\varphi_1, a_j, \frac{2\varphi_1^{(i)}}{\ell_{\varphi_1}^2}, \frac{-1}{\ell_{\varphi_1}^2}\right)\right].$$

Hence, according to the lemma 4.3.1, the mean $\mathbb{E}\left[(1)\right]$ can be computed analytically. In the same way, according to the lemma 4.3.1, and the fact that $1 - 4\left(\frac{-1}{\ell_{\varphi_1}^2}\right)\left(\sigma_1^c\right)^2 > 0$ and $1 - 2\left(\frac{-1}{\ell_{\varphi_1}^2}\right)\left(\sigma_1^c\right)^2 > 0$, the means $\mathbb{E}\left[(2)\right]$ and $\mathbb{E}\left[(3)\right]$ can be calculated analytically. Consequently, the mean $\mathbb{E}\left[\left(\mu_2^c\left(\varphi_1, \boldsymbol{x}_2\right)\right)^2 + \left(\sigma_2^c\left(\varphi_1, \boldsymbol{x}_2\right)\right)^2\right]$ can be calculated analytically, and its expression is:

$$\begin{aligned}
\mathbb{E}\left[\left(\mu_2^c\left(\varphi_1, \boldsymbol{x}_2\right)\right)^2 + \left(\sigma_2^c\left(\varphi_1, \boldsymbol{x}_2\right)\right)^2\right] &= \sigma_2^2 + \sum_{i=1}^{p_2}\sum_{j=1}^{p_2} \left(\boldsymbol{A}_h\right)_{ij} m_i\left(\boldsymbol{x}_2\right) m_j\left(\boldsymbol{x}_2\right) \mathbb{E}\left[g\left(\varphi_1, a_i + a_j, 0, 0\right)\right] \\
&+ \sum_{i=1}^{n_2}\sum_{j=1}^{n_2} \left(\boldsymbol{A}_c\right)_{ij} \ell\left(\boldsymbol{x}_2 - \boldsymbol{x}_2^{(i)}\right) \ell\left(\boldsymbol{x}_2 - \boldsymbol{x}_2^{(j)}\right) \exp\left(-\frac{\left(\varphi_1^{(i)}\right)^2 + \left(\varphi_1^{(j)}\right)^2}{\ell_{\varphi_1}^2}\right) \\
&\times \mathbb{E}\left[g\left(\varphi_1, 0, 2\frac{\varphi_1^{(i)} + \varphi_1^{(j)}}{\ell_{\varphi_1}^2}, \frac{-2}{\ell_{\varphi_1}^2}\right)\right] \\
&+ \sum_{i=1}^{n_2}\sum_{j=1}^{p_2} \left(\boldsymbol{A}_{ch}\right)_{ij} \ell\left(\boldsymbol{x}_2 - \boldsymbol{x}_2^{(i)}\right) m_j\left(\boldsymbol{x}_2\right) \exp\left(-\left(\frac{\varphi_1^{(i)}}{\ell_{\varphi_1}}\right)^2\right) \mathbb{E}\left[g\left(\varphi_1, a_j, \frac{2\varphi_1^{(i)}}{\ell_{\varphi_1}^2}, \frac{-1}{\ell_{\varphi_1}^2}\right)\right],
\end{aligned} \tag{4.6.8}$$

where $\boldsymbol{A}_h$, $\boldsymbol{A}_c$ and $\boldsymbol{A}_{ch}$ are defined in Eq. (4.6.7), $\boldsymbol{v}_c$ is defined in Eq. (4.6.3), $\ell\left(\boldsymbol{x}_2 - \boldsymbol{x}_2'\right)$ is defined by Eq. (4.6.4), ℓ_{φ_1} is the correlation length associated with φ_1 and $\widehat{\boldsymbol{\beta}}_2$ is given by Eq. (4.2.9).

From the two previous paragraphs and Proposition 4.2.1, it can be inferred that, if verifying the assumptions of Proposition 4.3.1, then the first and the second moments of $Y_{\text{nest}}^c(\boldsymbol{x}_1, \boldsymbol{x}_2)$ can be calculated analytically.

4.6.5 Proof of Proposition 4.3.2

If $Y^c_{\text{nest}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = Y^c_2(Y^c_1(\boldsymbol{x}_1), \boldsymbol{x}_2)$ where $Y^c_i = \mu^c_i + \varepsilon^c_i$, $\varepsilon^c_i \sim \text{GP}\left(0, C^c_i\right)$, $i \in \{1, 2\}$, then if ε^c_1 is small enough, the process $Y^c_{\text{nest}}(\boldsymbol{x}_1, \boldsymbol{x}_2)$ can be linearized:

$$\begin{aligned} Y^c_{\text{nest}}(\boldsymbol{x}_1, \boldsymbol{x}_2) &= \mu^c_2(\mu^c_1(\boldsymbol{x}_1) + \varepsilon^c_1(\boldsymbol{x}_1), \boldsymbol{x}_2) + \varepsilon^c_2(\mu^c_1(\boldsymbol{x}_1) + \varepsilon^c_1(\boldsymbol{x}_1), \boldsymbol{x}_2), \\ &\approx \mu^c_2(\mu^c_1(\boldsymbol{x}_1), \boldsymbol{x}_2) + \frac{\partial \mu^c_2}{\partial \varphi_1}(\mu^c_1(\boldsymbol{x}_1), \boldsymbol{x}_2)\varepsilon^c_1(\boldsymbol{x}_1) + \varepsilon^c_2(\mu^c_1(\boldsymbol{x}_1), \boldsymbol{x}_2). \end{aligned}$$

So, one can write:

$$Y^c_{\text{nest}}(\boldsymbol{x}_1, \boldsymbol{x}_2) \approx \mu^c_{\text{nest}}(\boldsymbol{x}_1, \boldsymbol{x}_2) + \varepsilon^c_{\text{nest}}(\mu^c_1(\boldsymbol{x}_1), \boldsymbol{x}_2), \tag{4.6.9}$$

with

$$\mu^c_{\text{nest}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \mu^c_2(\mu^c_1(\boldsymbol{x}_1), \boldsymbol{x}_2), \tag{4.6.10}$$

and

$$\varepsilon^c_{\text{nest}} = \frac{\partial \mu^c_2}{\partial \varphi_1}(\mu^c_1(\boldsymbol{x}_1), \boldsymbol{x}_2)\varepsilon^c_1(\boldsymbol{x}_1) + \varepsilon^c_2(\mu^c_1(\boldsymbol{x}_1), \boldsymbol{x}_2). \tag{4.6.11}$$

ε^c_1 and ε^c_2 are independent centered Gaussian processes, so $\varepsilon^c_{\text{nest}}$ is a centered Gaussian process, whose covariance function, C^c_{nest}, is given by:

$$\begin{aligned} C^c_{\text{nest}}((\boldsymbol{x}_1, \boldsymbol{x}_2), (\boldsymbol{x}'_1, \boldsymbol{x}'_2)) &= C^c_2((\mu^c_1(\boldsymbol{x}_1), \boldsymbol{x}_2), (\mu^c_1(\boldsymbol{x}'_1), \boldsymbol{x}'_2)) \\ &+ \frac{\partial \mu^c_2}{\partial \varphi_1}\left((\mu^c_1(\boldsymbol{x}_1), \boldsymbol{x}_2)\right) \frac{\partial \mu^c_2}{\partial \varphi_1}\left((\mu^c_1(\boldsymbol{x}'_1), \boldsymbol{x}'_2)\right) C^c_1(\boldsymbol{x}_1, \boldsymbol{x}'_1). \end{aligned} \tag{4.6.12}$$

From Eqs (4.6.9), (4.6.10), (4.6.11) and (4.6.12), it can be inferred that the predictor of the nested code can be defined as a Gaussian process with mean function μ^c_{nest} defined by Eq. (4.6.10), and covariance function C^c_{nest} defined by Eq. (4.6.12).

Moreover, it follows from Eq. (4.2.11) that:

$$\begin{aligned} \frac{\partial \mu^c_2}{\partial \varphi_1}(\varphi_1, \boldsymbol{x}_2) &= \left(\frac{\partial \boldsymbol{h}_2}{\partial \varphi_1}(\varphi_1, \boldsymbol{x}_2)\right)^T \widehat{\boldsymbol{\beta}}_2 \\ &+ \frac{\partial C^c_2}{\partial \varphi_1}\left((\varphi_1, \boldsymbol{x}_2), \overline{\boldsymbol{X}}^{\text{obs}}_2\right)\left(C_2\left(\overline{\boldsymbol{X}}^{\text{obs}}_2, \overline{\boldsymbol{X}}^{\text{obs}}_2\right)\right)^{-1}\left[\boldsymbol{y}^{\text{obs}}_2 - \boldsymbol{h}_2\left(\overline{\boldsymbol{X}}^{\text{obs}}_2\right)^T \widehat{\boldsymbol{\beta}}_2\right]. \end{aligned} \tag{4.6.13}$$

4.6.6 Proof of Corollary 4.3.3

Explicit mean

According to Eq. (4.2.11), if $\boldsymbol{h}_i$ and C_i can be computed explicitly, then μ^c_i can be computed explicitly. Therefore, according to Eq. (4.3.4), the mean of the Gaussian linearized predictor can be computed explicitly.

Explicit variance

According to Eq. (4.2.13), if $\boldsymbol{h}_i$ and C_i can be computed explicitly, then C^c_i can be computed explicitly.

According to Eq. (4.6.13), if $\boldsymbol{h}_2$, C_2 and the derivatives $\dfrac{\partial \boldsymbol{h}_2}{\partial \varphi_1}(\varphi_1, \boldsymbol{x}_2)$ and $\dfrac{\partial C_2}{\partial \varphi_1}\left((\varphi_1, \boldsymbol{x}_2), \overline{\boldsymbol{X}}^{\text{obs}}_2\right)$ can be computed explicitly, then the derivative of μ^c_2 with respect to φ_1 can be computed explicitly.

Therefore, according to Eq. (4.3.4), the variance of the Gaussian linearized predictor can be computed explicitly.

Hence it follows that, if $\boldsymbol{h}_i$ and C_i and the derivatives $\dfrac{\partial \boldsymbol{h}_2}{\partial \varphi_1}(\varphi_1, \boldsymbol{x}_2)$ and $\dfrac{\partial C_2}{\partial \varphi_1}\left((\varphi_1, \boldsymbol{x}_2), \overline{\boldsymbol{X}}_2^{\text{obs}}\right)$ can be computed explicitly, then the mean and the variance of the Gaussian linearized predictor of the nested code can be computed explicitly.

Moreover, the derivative $\dfrac{\partial C_2}{\partial \varphi_1}\left((\varphi_1, \boldsymbol{x}_2), \overline{\boldsymbol{X}}_2^{\text{obs}}\right)$ can be computed explicitly if C_2 is in the squared exponential or the Matérn $\dfrac{5}{2}$ class, and the associated explicit formulas are given in what follows.

Matérn $\dfrac{5}{2}$ class

If one denotes by:

$$\begin{aligned} \delta = & \ d\left((\varphi_1, \boldsymbol{x}_2), \left(\varphi_1', \boldsymbol{x}_2'\right)\right) \\ = & \ \sqrt{\frac{\left(\varphi_1 - \varphi_1'\right)^2}{\ell_{\varphi_1}^2} + \sum_{i=1}^{d_2} \frac{\left((\boldsymbol{x}_2)_i - (\boldsymbol{x}_2)_i\right)^2}{\ell_i^2}}, \end{aligned} \tag{4.6.14}$$

then, according to Eq. (4.2.7), the Matérn kernel can be rewritten:

$$K_{\frac{5}{2}}(\delta) = \left(1 + \sqrt{5}\delta + \frac{5}{3}\delta^2\right) \exp\left(-\sqrt{5}\delta\right). \tag{4.6.15}$$

Moreover, one has:

$$\frac{\partial \delta}{\partial \varphi_1} = \frac{\varphi_1 - \varphi_1'}{\ell_{\varphi_1}^2} \frac{1}{\delta}, \tag{4.6.16}$$

and

$$\begin{aligned} \frac{\partial K_{\frac{5}{2}}}{\partial \delta}(\delta) = & \ \exp\left(-\sqrt{5}\delta\right)\left[-\sqrt{5}\left(1 + \sqrt{5}\delta + \frac{5}{3}\delta^2\right) + \sqrt{5} + \frac{10}{3}\delta\right] \\ = & \ \exp\left(-\sqrt{5}\delta\right)\left[-5\delta - \sqrt{5}\frac{5}{3}\delta^2 + \frac{10}{3}\delta\right] \\ = & \ -\frac{5}{3}\delta\left(1 + \sqrt{5}\delta\right)\exp\left(-\sqrt{5}\delta\right), \end{aligned} \tag{4.6.17}$$

By noting that in the case of a Matérn $\dfrac{5}{2}$ kernel:

$$\frac{\partial C_2}{\partial \varphi_1} = \frac{\partial K_{\frac{5}{2}}}{\partial \delta} \frac{\partial \delta}{\partial \varphi_1},$$

the derivative of C_2 with respect to φ_1 is:

$$\begin{aligned} \frac{\partial C_2}{\partial \varphi_1}\left((\varphi_1, \boldsymbol{x}_2), \left(\varphi_1', \boldsymbol{x}_2'\right)\right) = & \ -\frac{5}{3}\frac{\varphi_1 - \varphi_1'}{\ell_{\varphi_1}^2}\left[1 + \sqrt{5}\, d\left((\varphi_1, \boldsymbol{x}_2), \left(\varphi_1', \boldsymbol{x}_2'\right)\right)\right] \\ & \exp\left[-\sqrt{5}\, d\left((\varphi_1, \boldsymbol{x}_2), \left(\varphi_1', \boldsymbol{x}_2'\right)\right)\right]. \end{aligned} \tag{4.6.18}$$

Squared exponential class

According to Eq. (4.2.5), the squared exponential kernel can be rewritten:

$$K_{\text{Gauss}}(\delta) = \exp\left(-\delta^2\right). \tag{4.6.19}$$

Hence, we have:

$$\frac{\partial K_{\text{Gauss}}}{\partial \delta}(\delta) = -2\delta \exp\left(-\delta^2\right). \tag{4.6.20}$$

By noting that, in the case of a squared exponential kernel:

$$\frac{\partial C_2}{\partial \varphi_1} = \frac{\partial K_{\text{Gauss}}}{\partial \delta}\frac{\partial \delta}{\partial \varphi_1},$$

the derivative of C_2 with respect to φ_1 is:

$$\frac{\partial C_2}{\partial \varphi_1}\left((\varphi_1, \boldsymbol{x}_2), \left(\varphi_1^{'}, \boldsymbol{x}_2^{'}\right)\right) = -2\frac{\varphi_1 - \varphi_1^{'}}{\ell_{\varphi_1}^2} \exp\left[-d\left((\varphi_1, \boldsymbol{x}_2), \left(\varphi_1^{'}, \boldsymbol{x}_2^{'}\right)\right)^2\right]. \tag{4.6.21}$$

Chapter 5

Gaussian process regression of two nested codes with functional outputs

In this chapter, we focus on the case of two nested codes with high dimensional vectorial outputs. Moreover, we assume that observations of the intermediary variable are available. We therefore consider the following system:

$$\begin{array}{ccccc} & & \boldsymbol{x}_2 & \searrow & \\ \boldsymbol{x}_1 & \rightarrow & \boldsymbol{y}_1(\boldsymbol{x}_1) & \nearrow & \boldsymbol{y}_{\text{nest}}(\boldsymbol{x}_{\text{nest}}) := \boldsymbol{y}_2(\boldsymbol{y}_1(\boldsymbol{x}_1), \boldsymbol{x}_2), \end{array} \tag{5.0.1}$$

where $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ and $\boldsymbol{x}_{\text{nest}}$ are low dimensional vectors and $\boldsymbol{y}_1$, $\boldsymbol{y}_2$ and $\boldsymbol{y}_{\text{nest}}$ are high dimensional vectors.
The work presented in this chapter will be published in the near future. An innovative dimension reduction method of the intermediary high dimensional vectorial variable is presented. Moreover, a Gaussian predictor of $\boldsymbol{y}_{\text{nest}}$, which takes into account the intermediary observations of $\boldsymbol{y}_1$, is proposed.

5.1 Introduction

The role of simulation for the design and the certification of complex systems is increasing. The design and the certification of complex systems involve methods like risk analysis or sensitivity analysis [Sobol, 2001]. However, such methods require the evaluation of the output of the studied system at a large number of input points. Therefore, if the computer codes are costly, the use of surrogate models becomes necessary.
In this chapter we focus on a chain of two codes with functional outputs, which are functions of time. By functional output, we mean high dimensional vectorial output and not infinite dimensional output [Pinski et al., 2015]. The functional output of the first code is one of the inputs of the second code. Several challenges are posed by such an application:

- there are two codes,
- the two codes are computationally expensive, with different computational costs,
- the second code has a functional input and a functional output,
- the two codes are coupled by a functional variable.

We want to perform a sensitivity analysis of the nested code output with respect to its inputs. Given the high computational cost of the codes, our first objective is to emulate the nested code output.
In the same manner as in the previous chapters (see Chapters 3 and 4), we focus on the Gaussian Process regression for the construction of this emulator. However, the existing works on Gaussian process regression generally deal with a single code or regard a chain or a network of codes as a single code. Besides, Chapter 4 only treats the case of two nested codes which are coupled by a scalar intermediary variable.

Regarding the Gaussian process surrogate modeling of a code with a functional output, the existing works generally deal with low dimensional vectorial inputs. Two approaches exist. The first one is based on a dimension reduction of the functional output (through a Principal Components Analysis for example) and the independent surrogate modeling of the projected components [Fricker et al., 2013; Higdon et al., 2008]. The second one is based on a tensorized (Kronecker) structure of the covariance function of the Gaussian process modeling the code, which means a separation between the time (or the indices of the functional output) and the other inputs [Hung et al., 2015; Rougier, 2008; Williams et al., 2006; Conti et al., 2009; Conti and O'Hagan, 2010]. Following Perrin [2018], the second approach will be chosen in this chapter.

Besides, concerning the Gaussian process surrogate modeling of a code with a functional input, the existing approaches generally rely on a projection of the functional input on a basis and consider a scalar output. The projection can be based on Partial Least Squares [Nanty et al., 2017] or Active Subspaces [Russi, 2010; Constantine et al., 2014]. However, the first approach cannot take into account additional information about the code, like causality. Indeed, in this chapter, the second code is causal, which means that the output at a given time depends only on the functional input at previous times. The second approach requires the knowledge of the derivatives of the code's output, but this information is not always available. In this chapter, we propose a method for the dimension reduction of the functional input of a code which is adapted to the output of the code. The proposed method enables to take into account additional information about the code, like causality, and does not require the knowledge of the derivatives of the code.

The contributions of this chapter are the following. First, we propose a method for the dimension reduction of the functional input of a code which is adapted to the output of the code. It does not require the knowledge of the derivatives of the code and can take into account additional information about the code, like causality. Second, we propose an extension of the work presented in Chapter 4 to the case of two codes coupled by a functional intermediary variable. The proposed method enables to obtain a predictor of the nested code which can take into account observations of the functional intermediary variable or observations of one of the two codes only. Moreover, the obtained predictor of the nested code is Gaussian with fast to compute conditioned mean and variance. Finally, we propose sequential design criteria [Jones et al., 1998; Picheny et al., 2010; Bect et al., 2012] which are suited for the improvement of the accuracy of the predictor of two nested codes with a functional intermediary variable. In particular, the nested structure of the codes is exploited by adding observations of one or the other code.

The proposed dimension reduction method is presented in Section 5.2 and compared to other existing dimension reduction techniques. Section 5.3 details the chosen approach for the surrogate modeling of a code with low dimensional vectorial inputs and a functional output. Section 5.4 contains the description of the Gaussian process predicting the nested code output and the associated sequential designs. Finally, the methods are applied to a numerical example

in Section 5.5.

Note that the proofs of the results presented in the remainder of this chapter are given in Section 5.8.

5.2 Dimension reduction of the functional input of a code

In this section, we study the dimension reduction of the functional input of a code with functional input and output. First, some existing methods are outlined. Then a dimension reduction technique based on a linear model between the functional input and output of the code is proposed.
In this chapter, the following additional notation will be used:

- $\boldsymbol{X}^c = \boldsymbol{I}_{N_t} - \boldsymbol{X}$ where $\boldsymbol{I}_{N_t}$ denotes the identity matrix and $\boldsymbol{X}$ is a $(N_t \times N_t)$-dimensional matrix.

5.2.1 Formalism

In this chapter, a system $\mathcal{S}$, characterized by low dimensional vectorial inputs $\boldsymbol{x}_{\text{nest}} \in \mathbb{X}_{\text{nest}}$, is considered. The output of the system is a time varying function and it is numerically represented by a very high dimensional vector. If we denote by N_t the dimension of the vector, the output of the system can be associated with the deterministic function $\boldsymbol{y}_{\text{nest}} : \mathbb{X}_{\text{nest}} \to \mathbb{R}^{N_t}$. Moreover, we focus on the case where the deterministic function $\boldsymbol{y}_{\text{nest}}$ can be broken down into two nested computer codes with functional outputs, which are characterized by the deterministic functions $\boldsymbol{y}_1 : \mathbb{X}_1 \to \mathbb{R}^{N_t}$ and $\boldsymbol{y}_2 : \mathbb{R}^{N_t} \times \mathbb{X}_2 \to \mathbb{R}^{N_t}$. The nested code can therefore be defined as follows:

$$\begin{array}{ccccc} & & \boldsymbol{x}_2 & \searrow & \\ & & & & \boldsymbol{y}_{\text{nest}}(\boldsymbol{x}_{\text{nest}}) := \boldsymbol{y}_2(\boldsymbol{y}_1(\boldsymbol{x}_1), \boldsymbol{x}_2), \\ \boldsymbol{x}_1 & \to & \boldsymbol{y}_1(\boldsymbol{x}_1) & \nearrow & \end{array} \tag{5.2.1}$$

where $\boldsymbol{x}_{\text{nest}} := (\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathbb{X}_{\text{nest}} = \mathbb{X}_1 \times \mathbb{X}_2$, $\mathbb{X}_1$ and $\mathbb{X}_2$ are subsets of $\mathbb{R}^{d_1}$ and $\mathbb{R}^{d_2}$, and d_1 and d_2 are two non-negative integers. Besides, we denote by $\mu_{\mathbb{X}_1}$ the probability measure associated with $\mathbb{X}_1$ and $\mu_{\mathbb{X}_2}$ the probability measure associated with $\mathbb{X}_2$.
The two codes are computationally expensive. Since we want to perform a sensitivity analysis of the nested code output with respect to its inputs, we aim therefore at constructing an emulator of the output of the nested code, which has to be accurate on the most likely regions of the input domain.
Besides, a set of n observations of the nested code is available. The observations of the inputs are drawn according to $\mu_{\mathbb{X}_1} \times \mu_{\mathbb{X}_2}$. The sets of observations of the inputs and output of the first code are denoted by:

$$\begin{aligned} \boldsymbol{X}_1^{\text{init}} &:= \left(\boldsymbol{x}_1^{(1)}, \ldots, \boldsymbol{x}_1^{(n)}\right), \\ \boldsymbol{Y}_1^{\text{init}} &:= \left(\boldsymbol{y}_1^{(1)} = \boldsymbol{y}_1\left(\boldsymbol{x}_1^{(1)}\right); \ldots; \boldsymbol{y}_1^{(n)} = \boldsymbol{y}_1\left(\boldsymbol{x}_1^{(n)}\right)\right), \end{aligned} \tag{5.2.2}$$

where $\boldsymbol{X}_1^{\text{init}}$ is a $(n \times d_1)$-dimensional matrix and $\boldsymbol{Y}_1^{\text{init}}$ is a $(N_t \times n)$-dimensional matrix. The observations $\boldsymbol{X}_1^{\text{init}}$ are independently drawn according to $\mu_{\mathbb{X}_1}$.
The sets of observations of the inputs and output of the second code are denoted by:

$$\begin{aligned} \boldsymbol{X}_2^{\text{init}} &:= \left(\boldsymbol{x}_2^{(1)}, \ldots, \boldsymbol{x}_2^{(n)}\right), \\ \boldsymbol{Y}_2^{\text{init}} &:= \left(\boldsymbol{y}_2^{(1)} = \boldsymbol{y}_2\left(\boldsymbol{y}_1^{(1)}, \boldsymbol{x}_2^{(1)}\right); \ldots; \boldsymbol{y}_2^{(n)} = \boldsymbol{y}_2\left(\boldsymbol{y}_1^{(n)}, \boldsymbol{x}_2^{(n)}\right)\right), \end{aligned} \tag{5.2.3}$$

where $\boldsymbol{X}_2^{\text{init}}$ is a $(n \times d_2)$-dimensional matrix and $\boldsymbol{Y}_2^{\text{init}}$ is a $(N_t \times n)$-dimensional matrix. The observations $\boldsymbol{X}_2^{\text{init}}$ are independently drawn according to $\mu_{\mathbb{X}_2}$.

As mentioned in 5.1, several challenges are raised by the objective of emulating the studied system. One of these challenges is the Gaussian process regression of the second code, which has a functional input and a functional output. Moreover, the derivatives of the code's output are unknown and few observations are available compared to the dimension of the functional variables. Given that the existing methods for the surrogate modeling of a code with a functional output are generally suited for the case of low dimensional vectorial inputs, our first objective is to reduce the dimension of the functional input of the second code in order to perform a Gaussian process regression of this code.
As explained in Chapter 2, two methods of dimension reduction coupled with a Gaussian process regression are commonly used. The first one is based on Partial Least Squares (see Nanty et al. [2017] and Section 2.1) and the second one is based on Active Subspaces (see [Constantine et al., 2014] and Section 2.1). The first one cannot easily take into account additional information about the code, like causality. The second one requires the knowledge of the first order derivatives of the output of the code, which are, in our case, not available.

Thanks to the dimension reduction of the functional input of the second code, the surrogate modeling of this code can be performed using an existing framework for the surrogate modeling of a code with low dimensional vectorial inputs and a functional output (see Conti et al. [2009] and Section 2.2).
In this section, we first outline the formerly introduced dimension reduction techniques, then we propose an innovative method for the dimension reduction of a functional input which is adapted to the output. It is based on a linear approximation of this output. In other words, we aim at estimating a $(N_t \times N_t)$-dimensional matrix $\boldsymbol{Z}$, with rank $m \ll N_t$, which minimizes:

$$\int_{\mathbb{X}_1 \times \mathbb{X}_2} \left\| \boldsymbol{y}_2\left(\boldsymbol{y}_1\left(\boldsymbol{x}_1\right), \boldsymbol{x}_2\right) - \boldsymbol{y}_2\left(\overline{\boldsymbol{y}}_1 + \boldsymbol{Z}\left(\boldsymbol{y}_1\left(\boldsymbol{x}_1\right) - \overline{\boldsymbol{y}}_1\right), \boldsymbol{x}_2\right) \right\|^2 d\mu_{\mathbb{X}_1}\left(\boldsymbol{x}_1\right) d\mu_{\mathbb{X}_2}\left(\boldsymbol{x}_2\right), \quad (5.2.4)$$

where $\overline{\boldsymbol{y}}_1 = \displaystyle\int_{\mathbb{X}_1} \boldsymbol{y}_1\left(\boldsymbol{x}_1\right) d\mu_{\mathbb{X}_1}\left(\boldsymbol{x}_1\right)$.

5.2.2 Dimension reduction of the functional input only

The Principal Components Analysis (PCA) (see Jackson [2003] and Section 2.1) is widely used for dimension reduction. With this method, the dimension reduction of the high-dimensional variable is based on the eigendecomposition of its covariance matrix.
This dimension reduction method takes into account only the functional input of the code.
In our framework, the output of the first code is also the functional input of the second code. The covariance matrix which has to be computed is thus $\operatorname{cov}\left(\boldsymbol{y}_1\right)$. It can be estimated from the set of n observations of the functional output $\boldsymbol{y}_1$ of the first code:

$$\boldsymbol{R}_{\boldsymbol{Y}_1^{\text{init}}} = \frac{1}{n-1} \sum_{i=1}^{n} \left(\boldsymbol{y}_1^{(i)} - \overline{\boldsymbol{Y}_1^{\text{init}}}\right) \left(\boldsymbol{y}_1^{(i)} - \overline{\boldsymbol{Y}_1^{\text{init}}}\right)^T, \quad (5.2.5)$$

where $\boldsymbol{y}_1^{(i)}$ are introduced in Eq. (5.2.2), and

$$\overline{\boldsymbol{Y}_1^{\text{init}}} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{y}_1^{(i)} \quad (5.2.6)$$

is the mean trajectory of the observations of $\boldsymbol{y}_1$.

However, given the high computational cost of the first code, few observations are available. The number of observations n is therefore small compared to the number of discretization steps N_t. The estimate of the $(N_t \times N_t)$-dimensional covariance matrix with the n observations may be not very accurate. We propose therefore another way to estimate the covariance matrix of the functional output of the first code.

Indeed, if a predictor of the first code output is available, then, following the Kriging Believer approach of Ginsbourger et al. [2010], the covariance matrix can be estimated from the predicted output at a large number N_1 of input points independently drawn according to $\mu_{\mathbb{X}_1}$. The predictor can be constructed from the set of n observations of the first code: $\boldsymbol{X}_1^{\text{init}}$ and $\boldsymbol{Y}_1^{\text{init}}$.
This estimate can be written:

$$\boldsymbol{R}_{\boldsymbol{\mu}_1} = \frac{1}{N_1 - 1} \sum_{i=1}^{N_1} \left(\boldsymbol{\mu}_1^{(i)} - \overline{\boldsymbol{\mu}_1} \right) \left(\boldsymbol{\mu}_1^{(i)} - \overline{\boldsymbol{\mu}_1} \right)^T, \tag{5.2.7}$$

where $\boldsymbol{\mu}_1^{(i)} = \boldsymbol{\mu}_1 \left(\tilde{\boldsymbol{x}}_1^{(i)} \right)$ is the mean of a Gaussian process emulator of $\boldsymbol{y}_1$ at input site $\tilde{\boldsymbol{x}}_1^{(i)}$. The input points $\{\tilde{\boldsymbol{x}}_1^{(i)}, 1 \leq i \leq N_1\}$ are independently drawn according to $\mu_{\mathbb{X}_1}$ and

$$\overline{\boldsymbol{\mu}_1} = \frac{1}{N_1} \sum_{i=1}^{N_1} \boldsymbol{\mu}_1^{(i)} \tag{5.2.8}$$

is the mean trajectory of the associated predicted trajectories.
The estimation of the covariance matrix from an increased number of paths compared to the initial set of observations enables to increase the rank of the estimator of the covariance matrix. We observed on a series of test cases (see Section 5.5.2) that this procedure could lead to projection bases which are more accurate.

5.2.3 Partial Least Squares regression

Partial Least Squares regression was introduced by Wold [1966] (see Section 2.1) and can be used for the reduction of the dimension of a high dimensional input of a code by taking into account the covariance between the high dimensional input and the output. As mentioned in Höskuldsson [1988], the projection basis can be defined using the singular value decomposition of the covariance matrix between the inputs and the output. In other words, if we denote by $\boldsymbol{U}\left(R_{\boldsymbol{y}_1\boldsymbol{y}_2}\right)$ the matrix whose columns gather the left-singular vectors of $R_{\boldsymbol{y}_1\boldsymbol{y}_2} = \text{cov}\left(\boldsymbol{y}_1, \boldsymbol{y}_2\right)$, then the projection basis is defined by the columns of $\boldsymbol{U}\left(R_{\boldsymbol{y}_1\boldsymbol{y}_2}\right)$. By construction, this projection basis is adapted to the output of the code.
Finally, the m first projected variables are given by $\boldsymbol{U}_m\left(R_{\boldsymbol{y}_1\boldsymbol{y}_2}\right)^T \boldsymbol{y}_1$ where $\boldsymbol{U}_m\left(R_{\boldsymbol{y}_1\boldsymbol{y}_2}\right)$ gathers the m left-singular vectors which are associated with the m largest singular values of $R_{\boldsymbol{y}_1\boldsymbol{y}_2}$.
Note that the covariance matrix between $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ can be estimated from the set of available observations:

$$\boldsymbol{R}_{\boldsymbol{Y}_1^{\text{init}}\boldsymbol{Y}_2^{\text{init}}} = \frac{1}{n-1} \sum_{i=1}^{n} \left(\boldsymbol{y}_1^{(i)} - \overline{\boldsymbol{Y}_1^{\text{init}}} \right) \left(\boldsymbol{y}_2^{(i)} - \overline{\boldsymbol{Y}_2^{\text{init}}} \right)^T, \tag{5.2.9}$$

where $\boldsymbol{y}_2^{(i)}$ are introduced in Eq. (5.2.3) and $\overline{\boldsymbol{Y}_2^{\text{init}}} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{y}_2^{(i)}$ is the mean trajectory of the observations of $\boldsymbol{y}_2$.

5.2.4 Proposition of a linear model-based dimension reduction of the functional input

In this section, we propose a method for the dimension reduction of the functional input of a code which:

1. is adapted to the surrogate modeling of the output of the code,
2. does not require the knowledge of the derivatives of the output,
3. can take into account additional information about the code, like causality.

This dimension reduction is based on an approximation of the output of the second code which is linear with respect to the functional input.
First, we propose a dimension reduction method which is adapted to a linear model. Then we propose an approximation of the output of a causal system by a linear model which is characterized by a small number of parameters.
The following Proposition provides a projection basis of the functional input of a linear model, which is adapted to output of the model. The case of a linear model is considered and a dimension reduction which is adapted to the linear model is proposed.

Proposition 5.2.1. *If one considers:*

1. *a linear model of the form* $\boldsymbol{A}\boldsymbol{x}_t$ *with* $\boldsymbol{A}$ *a* $(N_t \times N_t)$*-dimensional matrix and* $\boldsymbol{x}_t$ *a zero-mean* N_t*-dimensional vector,*
2. *a set of* N_x *observations of* $\boldsymbol{x}_t$ $\{\boldsymbol{x}_t^{(i)},\ 1 \leq i \leq N_x\}$*, which are gathered in the* $(N_t \times N_x)$*-dimensional matrix* $\boldsymbol{X}_t$*,*

then the m-rank matrix $\boldsymbol{Z}^*$ *which minimizes* $\sum_{i=1}^{N_x} \left\| \boldsymbol{A}\boldsymbol{x}_t^{(i)} - \boldsymbol{A}\boldsymbol{Z}\boldsymbol{x}_t^{(i)} \right\|^2$ *with respect to* $\boldsymbol{Z}$*, is given by:*

$$\boldsymbol{Z}^* = \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-\frac{1}{2}} \boldsymbol{U}_m \boldsymbol{D}_m \boldsymbol{V}_m^T \left(\boldsymbol{X}_t\boldsymbol{X}_t^T\right)^{-\frac{1}{2}}, \tag{5.2.10}$$

where $\left(\boldsymbol{X}_t\boldsymbol{X}_t^T\right)^{-\frac{1}{2}}$ *is the pseudo-inverse of* $\left(\boldsymbol{X}_t\boldsymbol{X}_t^T\right)^{\frac{1}{2}}$*,* $\left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-\frac{1}{2}}$ *is the pseudo-inverse of* $\left(\boldsymbol{A}^T\boldsymbol{A}\right)^{\frac{1}{2}}$*,* $\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$ *is the singular-value decomposition of* $\left(\boldsymbol{A}^T\boldsymbol{A}\right)^{\frac{1}{2}} \left(\boldsymbol{X}_t\boldsymbol{X}_t^T\right)^{\frac{1}{2}}$*,* $\boldsymbol{U}_m$ *and* $\boldsymbol{V}_m$ *gather the m first columns of matrices* $\boldsymbol{U}$ *and* $\boldsymbol{V}$*, and* $\boldsymbol{D}_m$ *is a diagonal matrix, whose diagonal gathers the m highest singular values.*

The proof of this Proposition is given in Section 5.8.

In what follows, we focus on the case where we approximate the output of the second code with a linear model with respect to its functional input in order to reduce the dimension of the functional input according to the method of Proposition 5.2.1. This linear model is of the form $L(\boldsymbol{a})\,\boldsymbol{y}_1$ where:

$$L(\boldsymbol{a}) = \begin{bmatrix} a_1 & & & \\ a_2 & a_1 & \boldsymbol{0} & \\ \vdots & \ddots & \ddots & \\ a_{N_t} & \dots & a_2 & a_1 \end{bmatrix}, \tag{5.2.11}$$

and $\boldsymbol{a} := (a_1, \dots, a_{N_t})$. Such a form corresponds to a linear causal model or filter. By noting that $L(\boldsymbol{a})\,\boldsymbol{y}_1 = L(\boldsymbol{y}_1)\,\boldsymbol{a}$, an estimation of $\boldsymbol{a}$ can be obtained from the observations:

$$\boldsymbol{a}^* = \underset{\boldsymbol{a}\in\mathbb{R}^{N_t}}{\operatorname{argmin}} \sum_{i=1}^{n} \left\| \boldsymbol{y}_2^{(i)} - L\left(\boldsymbol{y}_1^{(i)}\right)\boldsymbol{a} \right\|^2, \tag{5.2.12}$$

where the observations $\boldsymbol{y}_1^{(i)}$ are introduced in Eq. (5.2.2) and the observations $\boldsymbol{y}_2^{(i)}$ are introduced in Eq. (5.2.3). The vector $\boldsymbol{a}^*$ can be estimated from a set of observations of the second code:

$$\boldsymbol{a}^* = \left(\sum_{i=1}^{n} L\left(\boldsymbol{y}_1^{(i)}\right)^T L\left(\boldsymbol{y}_1^{(i)}\right)\right)^+ \sum_{i=1}^{n} L\left(\boldsymbol{y}_1^{(i)}\right)^T \boldsymbol{y}_2^{(i)}, \tag{5.2.13}$$

where $\left(\sum_{i=1}^{n} L\left(\boldsymbol{y}_1^{(i)}\right)^T L\left(\boldsymbol{y}_1^{(i)}\right)\right)^+$ is the pseudo-inverse of $\sum_{i=1}^{n} L\left(\boldsymbol{y}_1^{(i)}\right)^T L\left(\boldsymbol{y}_1^{(i)}\right)$.
This linear causal model is therefore characterized by a N_t-dimensional vector only. The number of parameters to be estimated has been reduced from N_t^2 with a classical linear model to N_t with this causal linear model.

In what follows, we aim at combining the linear model approximation and the projection with a m-rank matrix of Proposition 5.2.1. Such an approach is similar to Partial Least Squares regression, because the dimension reduction of the input is adapted to the output. However, unlike Partial Least Squares Regression, the proposed approach can take into account the sparse structure of the linear model of Eq. (5.2.11). Consequently, we propose a two-step approach: first a linear model with a sparse structure is estimated, then a dimension reduction adapted to this linear model is performed. Thus, if $\boldsymbol{A}^* = L(\boldsymbol{a}^*)$ denotes the matrix characterizing the approximated linear model, then, based on Proposition 5.2.1, we can define the projection matrix as follows:

$$\widehat{\boldsymbol{\Pi}}_m = \boldsymbol{V}_m^T \boldsymbol{R}_{\boldsymbol{\mu}_1}^{-\frac{1}{2}}, \tag{5.2.14}$$

where $\boldsymbol{R}_{\boldsymbol{\mu}_1}$ is defined by Eq. (5.2.7), $\boldsymbol{R}_{\boldsymbol{\mu}_1}^{-\frac{1}{2}}$ is the pseudo-inverse of $\boldsymbol{R}_{\boldsymbol{\mu}_1}^{\frac{1}{2}}$ and $\boldsymbol{V}_m$ gathers the m right-singular vectors corresponding to the m highest singular values of $\left(\left(\boldsymbol{A}^*\right)^T \boldsymbol{A}^*\right)^{\frac{1}{2}} \boldsymbol{R}_{\boldsymbol{\mu}_1}^{\frac{1}{2}}$.

From the previous paragraphs, we can define three functions of dimension reduction r_m of the functional input of the second code:

- for the Principal Component Analysis:

$$r_m(\boldsymbol{y}_1) = \boldsymbol{Q}_m\left(\boldsymbol{R}_{\boldsymbol{\mu}_1}\right)^T \left(\boldsymbol{y}_1 - \overline{\boldsymbol{\mu}_1}\right), \tag{5.2.15}$$

  where $\boldsymbol{Q}_m\left(\boldsymbol{R}_{\boldsymbol{\mu}_1}\right)$ is the $(N_t \times m)$-dimensional matrix gathering the eigenvectors associated with the m highest eigenvalues of $\boldsymbol{R}_{\boldsymbol{\mu}_1}$, defined by Eq. (5.2.7), and $\overline{\boldsymbol{\mu}_1}$ is defined by Eq. (5.2.8),

- for the Partial Least Squares regression:

$$r_m(\boldsymbol{y}_1) = \boldsymbol{U}_m\left(\boldsymbol{R}_{\boldsymbol{Y}_1^{\text{init}}\boldsymbol{Y}_2^{\text{init}}}\right)^T \left(\boldsymbol{y}_1 - \overline{\boldsymbol{Y}_1^{\text{init}}}\right), \tag{5.2.16}$$

  where $\boldsymbol{U}_m\left(\boldsymbol{R}_{\boldsymbol{Y}_1^{\text{init}}\boldsymbol{Y}_2^{\text{init}}}\right)$ is the $(N_t \times m)$-dimensional matrix gathering the left-singular vectors associated with the m highest singular values of $\boldsymbol{R}_{\boldsymbol{Y}_1^{\text{init}}\boldsymbol{Y}_2^{\text{init}}}$, defined by Eq. (5.2.9), and $\overline{\boldsymbol{Y}_1^{\text{init}}}$ is defined by Eq. (5.2.6),

- for the proposed dimension reduction method:

$$r_m(\boldsymbol{y}_1) = \widehat{\boldsymbol{\Pi}}_m\left(\boldsymbol{y}_1 - \overline{\boldsymbol{\mu}_1}\right), \tag{5.2.17}$$

  with $\widehat{\boldsymbol{\Pi}}_m$ defined by Eq. (5.2.14).

The optimal size m of the projection basis can be estimated using a stepwise forward selection criterion (see Section 1.3 for further details) when constructing the predictor of the second code.
Besides, it is worth noting that, when computing the singular values of matrices of rank less than n (the number of observations), only the n first singular values (eigenvalues for the symmetric matrices) are non-zero.

In this chapter, we adopt a Gaussian process regression framework. Thanks to the dimension reduction of the functional input of the second code, the second code can be associated with a Gaussian process which has a low dimensional vectorial input comprising the projection of the functional input and the inputs $\boldsymbol{x}_2$, and has a functional output. In the next section, we will define the properties of the Gaussian processes modeling codes with low dimensional vectorial inputs and a functional output.

5.3 Gaussian process regression with low dimensional inputs and a functional output

As mentioned above, the surrogate modeling will be performed using the Gaussian processes framework proposed in Conti et al. [2009] (see Section 2.2 for further details). The Gaussian processes modeling the codes have therefore a tensorized structure, which means that there is a separation between the time (or the indices of the functional output) and the other inputs, thanks to a Kronecker structure of the covariance function and a specific structure of the mean function.
As mentioned in the introduction of this chapter, we consider the following system:

$$\begin{array}{ccccc} & & \boldsymbol{x}_2 & \searrow & \\ & & & & \boldsymbol{y}_2, \\ \boldsymbol{x}_1 & \rightarrow & \boldsymbol{y}_1 & \nearrow & \end{array} \tag{5.3.1}$$

and we aim at emulating $(\boldsymbol{x}_1, \boldsymbol{x}_2) \mapsto \boldsymbol{y}_2$.

In order to achieve this objective, we propose the following system of Gaussian processes:

$$\begin{array}{ccccc} & & \boldsymbol{x}_2 & \overset{\boldsymbol{Y}_2}{\searrow} & \\ & & & & \boldsymbol{y}_2, \\ \boldsymbol{x}_1 & \overset{\boldsymbol{Y}_3}{\longrightarrow} & \boldsymbol{\rho} := r_m\left(\boldsymbol{y}_1\right) & \nearrow & \end{array} \tag{5.3.2}$$

and

$$\boldsymbol{x}_1 \overset{\boldsymbol{Y}_1}{\longrightarrow} \boldsymbol{y}_1 \tag{5.3.3}$$

where $r_m : \mathbb{R}^{N_t} \mapsto \mathbb{R}^m$ denotes the projection function of the functional input of the second code on its reduced basis of size m, and $\boldsymbol{Y}_1$, $\boldsymbol{Y}_2$ and $\boldsymbol{Y}_3$ are Gaussian processes.

The Gaussian process $\boldsymbol{Y}_1 : \mathbb{X}_1 \rightarrow \mathbb{R}^{N_t}$ will be used for the estimation of the projection basis (precisely for the estimation of the covariance matrix of $\boldsymbol{y}_1$), and for one of the sequential designs. For further details, the reader can refer to the definition of the sequential design criteria in Section 5.4.2.

The Gaussian process $\boldsymbol{Y}_3$ can be used for the emulation of the function $\boldsymbol{x}_1 \mapsto r_m\left(\boldsymbol{y}_1\left(\boldsymbol{x}_1\right)\right)$.
The output of the second code can be emulated thanks to the Gaussian process $\boldsymbol{Y}_2 : \mathbb{R}^m \times \mathbb{X}_2 \rightarrow \mathbb{R}^{N_t}$, which is indexed by $(\boldsymbol{\rho}, \boldsymbol{x}_2) \in \mathbb{R}^m \times \mathbb{X}_2$.

Note that if two outputs of the first code $\boldsymbol{y}_1(\boldsymbol{x}_1)$ and $\boldsymbol{y}_1(\boldsymbol{x}_1')$ have the same image under r_m, then their predicted values under $\boldsymbol{Y}_2$ are the same, but they can be different under $\boldsymbol{y}_2$.
Note that the coupling of the Gaussian predictors based on the Gaussian processes $\boldsymbol{Y}_3$ and $\boldsymbol{Y}_2$ will be useful for the prediction of the function $(\boldsymbol{x}_1, \boldsymbol{x}_2) \mapsto \boldsymbol{y}_2(\boldsymbol{y}_1(\boldsymbol{x}_1), \boldsymbol{x}_2)$, such that $\boldsymbol{Y}_2(\boldsymbol{Y}_3(\boldsymbol{x}_1), \boldsymbol{x}_2)$ approximates $\boldsymbol{y}_2(\boldsymbol{y}_1(\boldsymbol{x}_1), \boldsymbol{x}_2)$.

To harmonize the notations, we define:

$$\bar{\boldsymbol{x}}_i = \begin{cases} \boldsymbol{x}_1, \text{ if } i = 1 \text{ or } i = 3, \\ (\boldsymbol{\rho}, \boldsymbol{x}_2), \text{ if } i = 2, \end{cases} \tag{5.3.4}$$

$$\boldsymbol{y}_3 = r_m(\boldsymbol{y}_1), \tag{5.3.5}$$

and

$$\overline{\mathbb{X}}_i = \begin{cases} \mathbb{X}_1, \text{ if } i = 1 \text{ or } i = 3, \\ \mathbb{R}^m \times \mathbb{X}_2, \text{ if } i = 2. \end{cases} \tag{5.3.6}$$

Following Conti et al. [2009] (see also Section 2.2) and its generalization of Universal Kriging to the multi-output case, a functional output $\boldsymbol{y}_i$, $i \in \{1, 2, 3\}$, can be modeled by a realization of a Gaussian process $\boldsymbol{Y}_i$, whose *a priori* distribution is:

$$\boldsymbol{Y}_i(\cdot) | \boldsymbol{M}_i, \boldsymbol{R}_{t_i}, C_i \sim \mathrm{GP}(\boldsymbol{M}_i \boldsymbol{h}_i(\cdot), \boldsymbol{R}_{t_i} \otimes C_i(\cdot, \cdot)), \tag{5.3.7}$$

where $\boldsymbol{Y}_i(\bar{\boldsymbol{x}}_i) := (\boldsymbol{Y}_i(\bar{\boldsymbol{x}}_i, t_1), \ldots, \boldsymbol{Y}_i(\bar{\boldsymbol{x}}_i, t_{N_t}))$, $\boldsymbol{M}_i$ is a $(N_t \times p_i)$-dimensional matrix to be determined, $\boldsymbol{h}_i$ is a vector of p_i basis functions, $\boldsymbol{R}_{t_i}$ is a $(N_t \times N_t)$-dimensional symmetric non negative definite matrix, to be determined, and C_i is a covariance function on $\overline{\mathbb{X}}_i \times \overline{\mathbb{X}}_i$.

Moreover, a set of observations of the inputs and outputs of the two codes is available. In this chapter, we consider the case where an initial design is drawn according to the distribution of the inputs of the nested code, i.e. $\mu_{\mathbb{X}_1} \times \mu_{\mathbb{X}_2}$. Then this initial design is sequentially enriched. The criteria used for the sequential designs will be defined in the following.
The set of n_1 observations of the inputs and the functional output of the first code are denoted by:

$$\begin{aligned} \boldsymbol{X}_1^{\mathrm{obs}} &:= \left(\boldsymbol{x}_1^{(1)}, \ldots, \boldsymbol{x}_1^{(n_1)}\right), \\ \boldsymbol{Y}_1^{\mathrm{obs}} &:= \left(\boldsymbol{y}_1^{(1)} = \boldsymbol{y}_1\left(\boldsymbol{x}_1^{(1)}\right); \ldots; \boldsymbol{y}_1^{(n_1)} = \boldsymbol{y}_1\left(\boldsymbol{x}_1^{(n_1)}\right)\right), \end{aligned} \tag{5.3.8}$$

where $\overline{\boldsymbol{X}}_1^{\mathrm{obs}}$ is a $(n_1 \times d_1)$-dimensional matrix, and $\boldsymbol{Y}_1^{\mathrm{obs}}$ a $(N_t \times n_1)$-dimensional matrix,
In the same way, the sets of n_2 observations of the inputs and the output of the second code are denoted by:

$$\begin{aligned} \boldsymbol{\Phi}_1^{\mathrm{obs}} &:= \left(\boldsymbol{\varphi}_1^{(1)}; \ldots; \boldsymbol{\varphi}_1^{(n_2)}\right), \\ \boldsymbol{P}^{\mathrm{obs}} &:= \left(\boldsymbol{\rho}^{(1)} = r_m\left(\boldsymbol{\varphi}_1^{(1)}\right), \ldots, \boldsymbol{\rho}^{(n_2)} = r_m\left(\boldsymbol{\varphi}_1^{(n_2)}\right)\right), \\ \boldsymbol{X}_2^{\mathrm{obs}} &:= \left(\boldsymbol{x}_2^{(1)}, \ldots, \boldsymbol{x}_2^{(n_2)}\right), \\ \boldsymbol{Y}_2^{\mathrm{obs}} &:= \left(\boldsymbol{y}_2^{(1)} = \boldsymbol{y}_2\left(\boldsymbol{\varphi}_1^{(1)}, \boldsymbol{x}_2^{(1)}\right); \ldots; \boldsymbol{y}_2^{(n_2)} = \boldsymbol{y}_2\left(\boldsymbol{\varphi}_1^{(n_2)}, \boldsymbol{x}_2^{(n_2)}\right)\right), \end{aligned} \tag{5.3.9}$$

where $\boldsymbol{X}_2^{\mathrm{obs}}$ is a $(n_2 \times d_2)$-dimensional matrix, $\boldsymbol{P}^{\mathrm{obs}}$ is a $(n_2 \times m)$-dimensional matrix and $\boldsymbol{\Phi}_1^{\mathrm{obs}}$ and $\boldsymbol{Y}_1^{\mathrm{obs}}$ are $(N_t \times n_2)$-dimensional matrices. Note that $\boldsymbol{\varphi}_1$ can be the output of the first code or a prediction of the output of the first code.

Note that at the initial stage, we have:

$$\begin{aligned} \boldsymbol{X}_1^{\text{obs}} &= \boldsymbol{X}_1^{\text{init}}, \\ \boldsymbol{Y}_1^{\text{obs}} &= \boldsymbol{Y}_1^{\text{init}}, \\ \boldsymbol{\Phi}_1^{\text{obs}} &= \boldsymbol{Y}_1^{\text{init}}, \\ \boldsymbol{X}_2^{\text{obs}} &= \boldsymbol{X}_2^{\text{init}}, \\ \boldsymbol{Y}_2^{\text{obs}} &= \boldsymbol{Y}_2^{\text{init}}, \end{aligned} \tag{5.3.10}$$

where $\boldsymbol{X}_1^{\text{init}}$, $\boldsymbol{Y}_1^{\text{init}}$ are defined by Eq. (5.2.2) and $\boldsymbol{X}_2^{\text{init}}$, $\boldsymbol{Y}_2^{\text{init}}$ are defined by Eq. (5.2.3). Then, the design of experiments can be enriched thanks to sequential designs and the number of observations of the two codes can be different.
We can therefore introduce the following notations for the observations of the three studied functions:

$$\overline{\boldsymbol{X}}_i^{\text{obs}} := \begin{cases} \boldsymbol{X}_1^{\text{obs}}, \text{ if } i = 1 \text{ or } i = 3, \\ \left(\left(\boldsymbol{\rho}^{(1)}, \boldsymbol{x}_2^{(1)} \right), \ldots, \left(\boldsymbol{\rho}^{(n_2)}, \boldsymbol{x}_2^{(n_2)} \right) \right), \text{ if } i = 2. \end{cases} \tag{5.3.11}$$

Furthermore, the observations of $\boldsymbol{y}_3$ can be defined as:

$$\boldsymbol{Y}_3^{\text{obs}} := \left(r_m \left(\boldsymbol{y}_1 \left(\boldsymbol{x}_1^{(1)} \right) \right), \ldots, r_m \left(\boldsymbol{y}_1 \left(\boldsymbol{x}_1^{(n_1)} \right) \right) \right), \tag{5.3.12}$$

where $\boldsymbol{Y}_3^{\text{obs}}$ is a $(N_t \times n_1)$-dimensional matrix.

If the prior distribution of $\boldsymbol{M}_i$ is an improper uniform distribution on the space of the $(N_t \times p_i)$-dimensional real-valued matrices, the posterior distribution of $\boldsymbol{M}_i$ given the observations is Gaussian, with the following mean:

$$\begin{aligned} \widehat{\boldsymbol{M}}_i &= \mathbb{E}\left[\boldsymbol{M}_i | \boldsymbol{Y}_i^{\text{obs}}, \boldsymbol{R}_{t_i}, C_i \right] \\ &= \mathbb{E}\left[\boldsymbol{M}_i | \boldsymbol{Y}_i^{\text{obs}}, C_i \right] \\ &= \boldsymbol{Y}_i^{\text{obs}} \left(\boldsymbol{R}_i^{\text{obs}} \right)^{-1} \left(\boldsymbol{H}_i^{\text{obs}} \right)^T \left(\boldsymbol{H}_i^{\text{obs}} \left(\boldsymbol{R}_i^{\text{obs}} \right)^{-1} \left(\boldsymbol{H}_i^{\text{obs}} \right)^T \right)^{-1}, \end{aligned} \tag{5.3.13}$$

where $\boldsymbol{R}_i^{\text{obs}}$ is a $(n_i \times n_i)$-dimensional matrix such that:

$$\left(\boldsymbol{R}_i^{\text{obs}} \right)_{kl} = C_i \left(\bar{\boldsymbol{x}}_i^{(k)}, \bar{\boldsymbol{x}}_i^{(l)} \right), \tag{5.3.14}$$

where $\bar{\boldsymbol{x}}_i^{(k)}$ denotes the k-th observation of $\overline{\boldsymbol{X}}_i^{\text{obs}}$ and $\boldsymbol{H}_i^{\text{obs}}$ is a $(n_i \times p_i)$-dimensional matrix whose j-th line is given by $\boldsymbol{h}_i \left(\bar{\boldsymbol{x}}_i^{(j)} \right)$.
Moreover, one has:

$$\boldsymbol{Y}_i^{\text{obs}} | \boldsymbol{M}_i, \boldsymbol{R}_{t_i}, C_i \sim \mathcal{N}\left(\boldsymbol{M}_i \boldsymbol{H}_i^{\text{obs}}, \boldsymbol{R}_{t_i} \otimes \boldsymbol{R}_i^{\text{obs}} \right). \tag{5.3.15}$$

Therefore, the matrix $\boldsymbol{R}_{t_i}$ can be estimated by maximizing the likelihood of the observations, as proposed in Perrin [2018] (see Section 2.2 for further details). The estimator is thus:

$$\widehat{\boldsymbol{R}}_{t_i} = \boldsymbol{R}_{t_i}\left(\boldsymbol{Y}_i^{\text{obs}} \right) = \frac{1}{n_i} \left(\boldsymbol{Y}_i^{\text{obs}} - \widehat{\boldsymbol{M}}_i \boldsymbol{H}_i^{\text{obs}} \right) \left(\boldsymbol{R}_i^{\text{obs}} \right)^{-1} \left(\boldsymbol{Y}_i^{\text{obs}} - \widehat{\boldsymbol{M}}_i \boldsymbol{H}_i^{\text{obs}} \right)^T. \tag{5.3.16}$$

Finally, in the Universal Kriging framework, the conditional distribution of $\boldsymbol{Y}_i$ given the observations is:

$$\boldsymbol{Y}_i^c := \boldsymbol{Y}_i | \boldsymbol{Y}_i^{\text{obs}}, C_i \sim \text{GP}\left(\boldsymbol{\mu}_i^c, \widehat{\boldsymbol{R}}_{t_i} \otimes C_i^c \right), \tag{5.3.17}$$

with:

$$\begin{aligned}
\boldsymbol{\mu}_i^c(\bar{\boldsymbol{x}}_i) = & \ \widehat{\boldsymbol{M}}_i \boldsymbol{h}_i(\bar{\boldsymbol{x}}_i) + \left[\boldsymbol{Y}_i^{\text{obs}} - \widehat{\boldsymbol{M}}_i \boldsymbol{H}_i^{\text{obs}}\right] \left(\boldsymbol{R}_i^{\text{obs}}\right)^{-1} C\left(\overline{\boldsymbol{X}}_i^{\text{obs}}, \bar{\boldsymbol{x}}_i\right), \\
C_i^c\left(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_i'\right) = & \ C_i\left(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_i'\right) - C_i\left(\bar{\boldsymbol{x}}_i, \overline{\boldsymbol{X}}_i^{\text{obs}}\right) \left(\boldsymbol{R}_i^{\text{obs}}\right)^{-1} C_i\left(\overline{\boldsymbol{X}}_i^{\text{obs}}, \bar{\boldsymbol{x}}_i'\right) \\
& + \boldsymbol{u}_i(\bar{\boldsymbol{x}}_i)^T \left(\boldsymbol{H}_i^{\text{obs}} \left(\boldsymbol{R}_i^{\text{obs}}\right)^{-1} \left(\boldsymbol{H}_i^{\text{obs}}\right)^T\right)^{-1} \boldsymbol{u}_i\left(\bar{\boldsymbol{x}}_i'\right), \\
\boldsymbol{u}_i(\bar{\boldsymbol{x}}_i) = & \ \boldsymbol{h}_i(\bar{\boldsymbol{x}}_i) - \boldsymbol{H}_i^{\text{obs}} \left(\boldsymbol{R}_i^{\text{obs}}\right)^{-1} C_i\left(\overline{\boldsymbol{X}}_i^{\text{obs}}, \bar{\boldsymbol{x}}_i\right).
\end{aligned} \tag{5.3.18}$$

The prediction variance of the functional output at input $\bar{\boldsymbol{x}}_i$ is thus defined by:

$$\left(\boldsymbol{\sigma}_i^c\right)^2(\bar{\boldsymbol{x}}_i) := \operatorname{diag}\left(\mathbb{V}\left[\boldsymbol{Y}_i(\bar{\boldsymbol{x}}_i) \,|\boldsymbol{Y}_i^{\text{obs}}\right]\right), \tag{5.3.19}$$

This prediction variance can also be written:

$$\begin{aligned}
\left(\boldsymbol{\sigma}_i^c\right)^2(\bar{\boldsymbol{x}}_i) = & \ \operatorname{diag}\left(\widehat{\boldsymbol{R}}_{t_i}\left(\boldsymbol{Y}_i^{\text{obs}}\right)\right) C_i^c(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_i), \\
= & \ \operatorname{diag}\left(\widehat{\boldsymbol{R}}_{t_i}\left(\boldsymbol{Y}_i^{\text{obs}}\right)\right) v_i\left(\bar{\boldsymbol{x}}_i; \overline{\boldsymbol{X}}_i^{\text{obs}}\right),
\end{aligned} \tag{5.3.20}$$

where $v_i\left(\bar{\boldsymbol{x}}_i; \overline{\boldsymbol{X}}_i^{\text{obs}}\right) = C_i^c(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_i)$.

Finally, the correlation functions C_i are chosen in the parametric family of the Matérn $\frac{5}{2}$ correlation functions. Since our objective is to obtain an accurate prediction mean, the correlation lengths of the correlation function are estimated by minimizing a Cross Validation criterion [Dubrule, 1983]:

$$\begin{aligned}
\widehat{\boldsymbol{\ell}}_i = & \ \underset{\boldsymbol{\ell}_i \in \mathbb{R}^{d_i}}{\operatorname{argmin}} \ \operatorname{Tr}\left(\boldsymbol{Y}_i^{\text{obs}} \boldsymbol{R}_i^-(\boldsymbol{\ell}_i) \operatorname{diag}\left(\boldsymbol{R}_i^-(\boldsymbol{\ell}_i)\right)^{-2} \boldsymbol{R}_i^-(\boldsymbol{\ell}_i) \left(\boldsymbol{Y}_i^{\text{obs}}\right)^T\right), \\
= & \ \underset{\boldsymbol{\ell}_i \in \mathbb{R}^{d_i}}{\operatorname{argmin}} \ \left\| \boldsymbol{Y}_i^{\text{obs}} \boldsymbol{R}_i^-(\boldsymbol{\ell}_i) \operatorname{diag}\left(\boldsymbol{R}_i^-(\boldsymbol{\ell}_i)\right)^{-1} \right\|_F^2,
\end{aligned} \tag{5.3.21}$$

where:

$$\boldsymbol{R}_i^-(\boldsymbol{\ell}_i) = \left(\boldsymbol{R}_i^{\text{obs}}(\boldsymbol{\ell}_i)\right)^{-1} + \left(\boldsymbol{R}_i^{\text{obs}}(\boldsymbol{\ell}_i)\right)^{-1} \left(\boldsymbol{H}_i^{\text{obs}}\right)^T \left(\boldsymbol{H}_i^{\text{obs}} \left(\boldsymbol{R}_i^{\text{obs}}(\boldsymbol{\ell}_i)\right)^{-1} \left(\boldsymbol{H}_i^{\text{obs}}\right)^T\right)^{-1} \boldsymbol{H}_i^{\text{obs}} \left(\boldsymbol{R}_i^{\text{obs}}(\boldsymbol{\ell}_i)\right)^{-1}, \tag{5.3.22}$$

and $\boldsymbol{R}_i^{\text{obs}}(\boldsymbol{\ell}_i)$ is based on Eq. (5.3.14) and takes into account the fact that C_i depends on the correlation length $\boldsymbol{\ell}_i$.
Thanks to this matrix-form criterion, the correlation lengths of the correlation functions can be estimated quickly.

As for the correlation lengths, the size of the projection basis associated with r_m can be chosen according to a Cross Validation criterion (see Eq. (5.3.21)) of the form:

$$m^* = \underset{m \in \{1,\ldots,n\}}{\operatorname{argmin}} \ \left\| \boldsymbol{Y}_2^{\text{obs}} \boldsymbol{R}_2^-(m) \operatorname{diag}\left(\boldsymbol{R}_2^-(m)\right)^{-1} \right\|_F^2, \tag{5.3.23}$$

where:

$$\begin{aligned}
\boldsymbol{R}_2^-(m) = & \ \left(\boldsymbol{R}_2^{\text{obs}}(m)\right)^{-1} + \left(\boldsymbol{R}_2^{\text{obs}}(m)\right)^{-1} \left(\boldsymbol{H}_2^{\text{obs}}(m)\right)^T \\
& \left(\boldsymbol{H}_2^{\text{obs}}(m) \left(\boldsymbol{R}_2^{\text{obs}}(m)\right)^{-1} \left(\boldsymbol{H}_2^{\text{obs}}(m)\right)^T\right)^{-1} \boldsymbol{H}_2^{\text{obs}}(m) \left(\boldsymbol{R}_2^{\text{obs}}(m)\right)^{-1},
\end{aligned} \tag{5.3.24}$$

where $\boldsymbol{R}_2^{\text{obs}}(m)$ is a $(n \times n)$-dimensional matrix and $\boldsymbol{H}_2^{\text{obs}}(m)$ is a $(p_2 \times n)$-dimensional matrix, such that:

$$\left(\boldsymbol{R}_2^{\text{obs}}(m)\right)_{ij} = C_2\left(\left(r_m\left(\varphi_1^{(i)}\right), \boldsymbol{x}_2^{(i)}\right), \left(r_m\left(\varphi_1^{(j)}\right), \boldsymbol{x}_2^{(j)}\right)\right), \tag{5.3.25}$$

and

$$\left(\boldsymbol{H}_2^{\text{obs}}(m)\right)_{\cdot i} = \boldsymbol{h}_2\left(\left(r_m\left(\varphi_1^{(i)}\right), \boldsymbol{x}_2^{(i)}\right)\right). \tag{5.3.26}$$

It is also worth noting that the hyperparameters of the correlation function C_2 have to be re-estimated for each value of m. These computations can be speeded up using the criterion of Eq. (5.3.21).

From Eq. (5.3.17), it follows that the accuracy of the Gaussian posterior predictor depends on the choice of the set of observations $\boldsymbol{Y}_i^{\text{obs}}$. Consequently, the predictor can be improved by choosing an appropriate design.
When one aims at improving the prediction accuracy of a given quantity of interest, a design criterion based on the integrated prediction variance of the quantity of interest is generally used (see Section 1.5). However, the computation of such a design criterion in one step can be cumbersome, because of the high number of possible sets. Indeed, if a discrete search is performed, the number of possible combinations is $\binom{\mathcal{N}_i}{n_i}$, where $\mathcal{N}_i$ is the number of candidates. Moreover, the covariance function of the Gaussian process is generally not known at the initial stage and has to be estimated from an initial set of observations of the codes associated to the Gaussian processes. Then this initial design can be enriched thanks to sequential design criteria or Stepwise Uncertainty Reduction methods [Bect et al., 2012; Picheny et al., 2010]. Besides, the hyperparameters of the correlation function C_i of the Gaussian process can be re-estimated at each step from the new set of observations.
Following Bates et al. [1996] and Picheny et al. [2010], a criterion based on the reduction of the integrated prediction variance can be used. Based on Eq. (5.3.19), this criterion can be written:

$$\bar{\boldsymbol{x}}_i^{new} = \underset{\bar{\boldsymbol{x}}_i^* \in \overline{\mathbb{X}}_i}{\operatorname{argmin}} \int_{\overline{\mathbb{X}}_i} \operatorname{Tr}\left(\mathbb{V}\left[\boldsymbol{Y}_i\left(\bar{\boldsymbol{x}}_i\right) | \boldsymbol{Y}_i^{\text{obs}}, \boldsymbol{y}_i\left(\bar{\boldsymbol{x}}_i^*\right)\right]\right)_j d\mu_{\overline{\mathbb{X}}_i}\left(\bar{\boldsymbol{x}}_i\right), \tag{5.3.27}$$

where

$$\mu_{\overline{\mathbb{X}}_i} := \begin{cases} \mu_{\mathbb{X}_1}, \text{ if } i = 1 \text{ or } i = 3, \\ \mu_{\boldsymbol{\rho}} \times \mu_{\mathbb{X}_2}, \text{ if } i = 2, \end{cases} \tag{5.3.28}$$

with $\mu_{\boldsymbol{\rho}}$ denoting the probability measure over $\mathbb{R}^m$ associated with the distribution of $r_m\left(\boldsymbol{y}_1\left(\boldsymbol{x}_1\right)\right)$ under $\mu_{\mathbb{X}_1}$. This probability measure is unknown, because the distribution of $\boldsymbol{y}_1\left(\boldsymbol{x}_1\right)$ under $\mu_{\mathbb{X}_1}$ is not known. However, draws according to $\mu_{\boldsymbol{\rho}}$ can be obtained by drawing independent points according to $\mu_{\mathbb{X}_1}$ and then considering the image of these points under $\boldsymbol{\mu}_3^c$, which is defined by Eq. (5.3.18).
Note that $\boldsymbol{y}_i\left(\bar{\boldsymbol{x}}_i^*\right)$ is not known. In order to compute the design criterion, a Kriging Believer approach [Ginsbourger et al., 2010] can be used, as mentioned in the following proposition. Moreover, the Proposition is based on the simplifications presented in the Lemma 1.

Lemma 1. *One has:*

$$\mathbb{E}\left[\boldsymbol{M}_i | \{\boldsymbol{Y}_i^{obs}, \boldsymbol{Y}_i\left(\bar{\boldsymbol{x}}_i^*\right) = \boldsymbol{\mu}_i^c\left(\bar{\boldsymbol{x}}_i^*\right)\}, C_i\right] = \mathbb{E}\left[\boldsymbol{M}_i | \boldsymbol{Y}_i^{obs}, C_i\right]. \tag{5.3.29}$$

and

$$\boldsymbol{R}_{t_i}\left(\boldsymbol{Y}_i^{obs}, \boldsymbol{\mu}_i^c\left(\bar{\boldsymbol{x}}_i^*\right)\right) = \frac{n_i}{n_i + 1} \boldsymbol{R}_{t_i}\left(\boldsymbol{Y}_i^{obs}\right), \tag{5.3.30}$$

where $\boldsymbol{R}_{t_i}\left(\boldsymbol{Y}_i^{\text{obs}}\right)$ is defined by Eq. (5.3.16)
The proof of this lemma can be found in Section 5.8.

Proposition 5.3.1. *In the Kriging Believer framework of Ginsbourger et al. [2010], the criterion of Eq.* (5.3.27) *can be written:*

$$\bar{\boldsymbol{x}}_i^{new} = \underset{\bar{\boldsymbol{x}}_i^* \in \overline{\mathbb{X}}_i}{\operatorname{argmin}} \int_{\overline{\mathbb{X}}_i} v_i\left(\bar{\boldsymbol{x}}_i; \overline{\boldsymbol{X}}_i^{obs}, \bar{\boldsymbol{x}}_i^*\right) d\mu_{\overline{\mathbb{X}}_i}\left(\bar{\boldsymbol{x}}_i\right), \tag{5.3.31}$$

where $v_i\left(\bar{\boldsymbol{x}}_i; \overline{\boldsymbol{X}}_i^{obs}, \bar{\boldsymbol{x}}_i^*\right)$ *is defined by Eq.* (5.3.20), *with* $\overline{\boldsymbol{X}}_i^{obs}$ *defined by Eq.* (5.3.11).

The proof of this Proposition is in Section 5.8.

Besides, the integral can be computed using a Monte-Carlo method. Independent points $\bar{\boldsymbol{x}}_i$ are drawn according to $\mu_{\overline{\mathbb{X}}_i}$, the function $v_i\left(\bar{\boldsymbol{x}}_i; \overline{\boldsymbol{X}}_i^{\text{obs}}, \bar{\boldsymbol{x}}_i^*\right)$ is computed at these points and finally an estimate of the integral is obtained by computing the empirical mean of these computed outputs of v_i.
Now that a predictor of all the studied functions can be defined, we will study the surrogate modeling of the nested code.

5.4 Surrogate modeling of the nested code

In this section, we propose a predictor of the nested code which can take into account all the available observations. This predictor is obtained through the coupling of predictors of the two nested codes. We also define sequential design criteria aiming at improving the prediction accuracy of the output of the nested code. Moreover, an adaptation of the coupling of the predictors is proposed, in order to obtain a Gaussian predictor of the output of the nested code.

5.4.1 A Gaussian predictor of the nested code thanks to a linearization of the coupling of two predictors

In the previous section, two Gaussian predictors were obtained:

- $\boldsymbol{Y}_2^c : \mathbb{R}^m \times \mathbb{R}^{d_2} \to \mathbb{R}^{N_t}$,
- $\boldsymbol{Y}_3^c : \mathbb{R}^{d_1} \to \mathbb{R}^m$.

We therefore propose the following predictor of $(\boldsymbol{x}_1, \boldsymbol{x}_2) \mapsto \boldsymbol{y}_2\left(\boldsymbol{y}_1\left(\boldsymbol{x}_1\right), \boldsymbol{x}_2\right)$:

$$\boldsymbol{Y}_{\text{nest}}^c\left(\boldsymbol{x}_1, \boldsymbol{x}_2\right) := \boldsymbol{Y}_2^c\left(\boldsymbol{Y}_3^c\left(\boldsymbol{x}_1\right), \boldsymbol{x}_2\right). \tag{5.4.1}$$

with $\boldsymbol{Y}_2^c$ and $\boldsymbol{Y}_3^c$ defined by Eq. (5.3.17).

Proposition 5.4.1. *For all* $(\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathbb{X}_1 \times \mathbb{X}_2$, *if* $\boldsymbol{\xi} \sim \mathcal{N}\left(0, \widehat{\boldsymbol{R}}_{t_3}\right)$, *then:*

$$\mathbb{E}\left[\boldsymbol{Y}_{nest}^c(\boldsymbol{x}_1, \boldsymbol{x}_2)\right] = \mathbb{E}\left[\boldsymbol{\mu}_2^c(\boldsymbol{\mu}_3^c(\boldsymbol{x}_1) + \sigma_{\boldsymbol{x}_3}^c(\boldsymbol{x}_1)\boldsymbol{\xi}, \boldsymbol{x}_2)\right], \tag{5.4.2}$$

$$\mathbb{E}\left[\boldsymbol{Y}_{nest}^c(\boldsymbol{x}_1, \boldsymbol{x}_2)\left(\boldsymbol{Y}_{nest}^c(\boldsymbol{x}_1, \boldsymbol{x}_2)\right)^T\right] = \mathbb{E}\left[\begin{array}{l} \boldsymbol{\mu}_2^c(\boldsymbol{\mu}_1^c(\boldsymbol{x}_1) + \sigma_{\boldsymbol{x}_1}^c(\boldsymbol{x}_1)\boldsymbol{\xi}, \boldsymbol{x}_2)\ \boldsymbol{\mu}_2^c(\boldsymbol{\mu}_1^c(\boldsymbol{x}_1) + \sigma_{\boldsymbol{x}_1}^c(\boldsymbol{x}_1)\boldsymbol{\xi}, \boldsymbol{x}_2)^T \\ + \{\sigma_{\boldsymbol{x}_2}^c(\boldsymbol{\mu}_1^c(\boldsymbol{x}_1) + \sigma_{\boldsymbol{x}_1}^c(\boldsymbol{x}_1)\boldsymbol{\xi}, \boldsymbol{x}_2)\}^2 \widehat{\boldsymbol{R}}_{t_2} \end{array}\right], \tag{5.4.3}$$

where $\sigma_{\boldsymbol{x}_1}^c(\boldsymbol{x}_1) = \sqrt{C_1^c(\boldsymbol{x}_1, \boldsymbol{x}_1)}$ *and* $\sigma_{\boldsymbol{x}_2}^c(\boldsymbol{x}_2) = \sqrt{C_2^c(\boldsymbol{x}_2, \boldsymbol{x}_2)}$.

The proof of this Proposition is in Section 5.8.

Note that the computation of the two first moments of the coupling of the predictors involves the computation of two m-dimensional integrals. This computation can be performed using Monte-Carlo methods [Baker, 1977] or quadrature rules. However, the computation of these integrals can be computationally expensive. We therefore develop an adaptation of the linearized method proposed in Section 4.3 to the case where the outputs of the codes are two time-varying functions. This enables to obtain a Gaussian predictor of the nested code with conditioned mean and variance which can be computed quickly.

Proposition 5.4.2. *If*

1. $\boldsymbol{Y}_i^c = \boldsymbol{\mu}_i^c + \boldsymbol{\varepsilon}_i^c$ *denotes a predictor of* $\boldsymbol{y}_i$ *for* $i \in \{2,3\}$ *and* $\boldsymbol{\varepsilon}_i^c(\cdot) \sim GP\left(\mathbf{0}, \widehat{\boldsymbol{R}}_{t_i} C_i^c(\cdot,\cdot)\right)$, *where* $\boldsymbol{\mu}_i^c$ *and* C_i^c *are defined in Eq.* (5.3.18) *and* $\widehat{\boldsymbol{R}}_{t_i}$ *in Eq.* (5.3.16).
2. *the magnitude of the prediction error* $\boldsymbol{\varepsilon}_3^c$ *of the predictor associated with the first code enables the linearization,*

then a Gaussian predictor of the nested code can be obtained, and its mean and covariance functions are defined as follows:

$$\boldsymbol{\mu}_{nest}^c(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{\mu}_2^c\left(\boldsymbol{\mu}_3^c(\boldsymbol{x}_1), \boldsymbol{x}_2\right), \tag{5.4.4}$$

$$\begin{aligned} \boldsymbol{C}_{nest}^c\left((\boldsymbol{x}_1, \boldsymbol{x}_2), \left(\boldsymbol{x}_1', \boldsymbol{x}_2'\right)\right) &= \widehat{\boldsymbol{R}}_{t_2} C_2^c\left(\left(\boldsymbol{\mu}_3^c(\boldsymbol{x}_1), \boldsymbol{x}_2\right), \left(\boldsymbol{\mu}_3^c\left(\boldsymbol{x}_1'\right), \boldsymbol{x}_2'\right)\right) \\ &+ \frac{\partial \boldsymbol{\mu}_2^c}{\partial \boldsymbol{\rho}}\left(\boldsymbol{\mu}_3^c(\boldsymbol{x}_1), \boldsymbol{x}_2\right) \widehat{\boldsymbol{R}}_{t_3} \left(\frac{\partial \boldsymbol{\mu}_2^c}{\partial \boldsymbol{\rho}}\left(\boldsymbol{\mu}_3^c\left(\boldsymbol{x}_1'\right), \boldsymbol{x}_2'\right)\right)^T C_3^c\left(\boldsymbol{x}_1, \boldsymbol{x}_1'\right). \end{aligned} \tag{5.4.5}$$

The predictor obtained can also be written in the form $\boldsymbol{Y}_{nest}^c := \boldsymbol{\mu}_{nest}^c + \boldsymbol{\varepsilon}_{nest}^c$, *where:*

$$\boldsymbol{\varepsilon}_{nest}^c(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{\partial \boldsymbol{\mu}_2^c}{\partial \boldsymbol{\rho}}\left(\boldsymbol{\mu}_3^c(\boldsymbol{x}_1), \boldsymbol{x}_2\right) \boldsymbol{\varepsilon}_3^c(\boldsymbol{x}_1) + \boldsymbol{\varepsilon}_2^c\left(\boldsymbol{\mu}_3^c(\boldsymbol{x}_1), \boldsymbol{x}_2\right). \tag{5.4.6}$$

The proof of this Proposition is in Section 5.8.

The predictor obtained is therefore Gaussian and can take into account a different number of observations for each code. It is based on all the possible types of observations.
Note that the smaller C_3^c, the more valid the linearization. Moreover, since the linearization is performed using the conditioned process $\boldsymbol{Y}_3^c$, the norm of C_3^c is more likely to be small. Besides, the norm of C_3^c can be reduced by an appropriate enrichment of the set of observations of the first code.

5.4.2 Sequential designs

In the previous section, we have proposed a method to construct a Gaussian process emulator of the nested code for a given set of observations of the two codes. By construction, the accuracy of this predictor depends on the set of observations of the two codes. A way to improve the prediction accuracy is an appropriate choice of the set of observations.
As for the case of a single code (see Section 5.3), sequential design criteria or Stepwise Uncertainty Reduction [Bect et al., 2012; Picheny et al., 2010] methods are chosen.

Since we want to perform a sensitivity analysis of the nested code output with respect to its inputs, the design should lead to an accurate prediction mean on the most likely regions of the input domain of the nested code. The design criteria are therefore based on the integrated prediction variance [Sacks et al., 1989; Santner et al., 2003].
The two proposed design criteria are a generalization of Eq. (5.3.31) to the case of two nested codes. They are also an adaptation of those proposed in Section 4.2 to the case of two codes with functional outputs. The criteria of Section 4.2 were defined for the case of two nested codes with scalar outputs. They are based on the minimization of the integrated prediction variance. One of the criteria corresponds to the case where the two codes can be launched separately, the other one corresponds to the case where they cannot be launched separately. It is worth noting that the previously proposed predictor of the nested code can take into account a different number of observations for the two codes, which enables to define a sequential design criterion which chooses the best candidate among the two codes and takes into account the computational costs of the codes.

Definition 1. *In the case of two nested codes with functional outputs, two selection criteria based on the minimization of the integrated prediction variance can be defined:*

- *the Chained I-optimal criterion, which selects the candidate in $\mathbb{X}_{nest}$ which minimizes the integrated prediction variance given this candidate:*

$$(\boldsymbol{x}_1^{new}, \boldsymbol{x}_2^{new}) = \underset{(\boldsymbol{x}_1^*, \boldsymbol{x}_2^*) \in \mathbb{X}_{nest}}{\operatorname{argmin}} \int_{\mathbb{X}_1 \times \mathbb{X}_2} \left(Tr\left(\widehat{\boldsymbol{R}}_{t_2}\right) v_2\left(\left(\boldsymbol{\mu}_3^c\left(\boldsymbol{x}_1\right), \boldsymbol{x}_2\right); \overline{\boldsymbol{X}}_2^{obs}, \left(\boldsymbol{\mu}_3^c\left(\boldsymbol{x}_1^*\right), \boldsymbol{x}_2^*\right)\right) + \right.$$
$$\left. Tr\left(\frac{\partial \boldsymbol{\mu}_2^c}{\partial \boldsymbol{\rho}}\left(\boldsymbol{\mu}_3^c\left(\boldsymbol{x}_1\right), \boldsymbol{x}_2\right) \widehat{\boldsymbol{R}}_{t_3} \left(\frac{\partial \boldsymbol{\mu}_2^c}{\partial \boldsymbol{\rho}}\left(\boldsymbol{\mu}_3^c\left(\boldsymbol{x}_1\right), \boldsymbol{x}_2\right)\right)^T\right) v_3\left(\boldsymbol{x}_1; \overline{\boldsymbol{X}}_1^{obs}, \boldsymbol{x}_1^*\right)\right) d\mu_{\mathbb{X}_1}\left(\boldsymbol{x}_1\right) d\mu_{\mathbb{X}_2}\left(\boldsymbol{x}_2\right), \tag{5.4.7}$$

  where $\widehat{\boldsymbol{R}}_{t_i}$ $i \in \{2,3\}$ is defined by Eq. (5.3.16), and v_i is defined by Eq. (5.8.4),

- *the Best I-optimal criterion, which selects the candidate among the two codes which maximizes the reduction of the integrated prediction variance per unit of computational cost given this candidate:*

$$(i^{new}, \boldsymbol{x}_{i^{new}}^{new}) = \underset{(i, \tilde{\boldsymbol{x}}_i) \in \{1,2\} \times \tilde{\mathbb{X}}_i}{\operatorname{argmin}} \frac{1}{\tau_i} \mathcal{V}_i\left(\tilde{\boldsymbol{x}}_i\right), \tag{5.4.8}$$

  with τ_i denoting the computational cost of the code i, and:

$$\mathcal{V}_1\left(\tilde{\boldsymbol{x}}_1\right) := \int_{\mathbb{X}_1 \times \mathbb{X}_2} Tr\left(\frac{\partial \boldsymbol{\mu}_2^c}{\partial \boldsymbol{\rho}}\left(\boldsymbol{\mu}_3^c\left(\boldsymbol{x}_1\right), \boldsymbol{x}_2\right) \widehat{\boldsymbol{R}}_{t_3} \left(\frac{\partial \boldsymbol{\mu}_2^c}{\partial \boldsymbol{\rho}}\left(\boldsymbol{\mu}_3^c\left(\boldsymbol{x}_1\right), \boldsymbol{x}_2\right)\right)^T\right)$$
$$\left(v_3\left(\boldsymbol{x}_1; \overline{\boldsymbol{X}}_1^{obs}\right) - v_3\left(\boldsymbol{x}_1; \overline{\boldsymbol{X}}_1^{obs}, \tilde{\boldsymbol{x}}_1\right)\right) d\mu_{\mathbb{X}_1}\left(\boldsymbol{x}_1\right) d\mu_{\mathbb{X}_2}\left(\boldsymbol{x}_2\right), \tag{5.4.9}$$

$$\mathcal{V}_2\left(\tilde{\boldsymbol{x}}_2\right) := \int_{\mathbb{X}_1 \times \mathbb{X}_2} \left(v_2\left(\left(\boldsymbol{\mu}_3^c\left(\boldsymbol{x}_1\right), \boldsymbol{x}_2\right); \overline{\boldsymbol{X}}_2^{obs}\right) - v_2\left(\left(\boldsymbol{\mu}_3^c\left(\boldsymbol{x}_1\right), \boldsymbol{x}_2\right); \overline{\boldsymbol{X}}_2^{obs}, \left(\boldsymbol{\mu}_3^c\left(\boldsymbol{x}_1^*\right), \boldsymbol{x}_2^*\right)\right)\right)$$
$$Tr\left(\widehat{\boldsymbol{R}}_{t_2}\right) d\mu_{\mathbb{X}_1}\left(\boldsymbol{x}_1\right) d\mu_{\mathbb{X}_2}\left(\boldsymbol{x}_2\right), \tag{5.4.10}$$

  and:

$$\{\tilde{\boldsymbol{x}}_i, \tilde{\mathbb{X}}_i\} = \begin{cases} \{\boldsymbol{x}_1^*, \mathbb{X}_1\}, \text{ if } i = 1, \\ \{(\boldsymbol{x}_1^*, \boldsymbol{x}_2^*) \ , \ \mathbb{X}_1 \times \mathbb{X}_2\}, \text{ if } i = 2. \end{cases} \tag{5.4.11}$$

  Moreover, if a candidate $\tilde{\boldsymbol{x}}_2 = (\boldsymbol{x}_1^*, \boldsymbol{x}_2^*)$ is chosen, the second code will be evaluated at $(\boldsymbol{\mu}_1^c\left(\boldsymbol{x}_1^*\right), \boldsymbol{x}_2^*)$.

Note that the Best I-optimal criterion is based on the assumption that the integrated prediction variance decreases linearly at each step.

Both criteria imply a multidimensional integration and optimization on $\mathbb{X}_1$ or $\mathbb{X}_1 \times \mathbb{X}_2$. The integration is computed using the empirical average of a Monte-Carlo draw. The optimization is performed on a finite set of candidates drawn according to the probability measure of $\mu_{\mathbb{X}_1}$ or $\mu_{\mathbb{X}_1 \times \mathbb{X}_2}$. The fact that the prediction variance has a closed-form expression, and is thus fast to evaluate, is an advantage for the computation of the criteria, because they both require a high number of evaluations of the prediction variance.
The estimation of the projection basis, of the hyperparameters of the covariance functions $C_i,\ i \in \{1,2,3\}$, of the functional covariance matrices $\widehat{\boldsymbol{R}}_{t_i},\ i \in \{1,2,3\}$, and of the conditioned mean functions $\boldsymbol{\mu}_i^c,\ i \in \{1,2,3\}$, require an initial design of experiments. Once again, since we aim at predicting the nested code output on the most likely regions of the input domain $\mathbb{X}_{\text{nest}}$ and we have no *a priori* information, an initial set of observations is drawn according to the probability measure $\mu_{\mathbb{X}_1} \times \mu_{\mathbb{X}_2}$.

The projection basis associated with r_m is estimated from the initial set of observations. The dimension m of the projection of the functional input of the second code is chosen according to the criterion of Eq. (5.3.23).

The hyperparameters of the correlation function C_3 are estimated from the initial design and when a new observation of the first code is added. The hyperparameters of the correlation functions C_1 and C_2 are estimated from the initial design and when a new observation of the second code is added. The hyperparameters are estimated thanks to the Cross-Validation criterion of Eq. (5.3.21).

5.5 First numerical example

In this section, we apply the proposed methods to a numerical example.

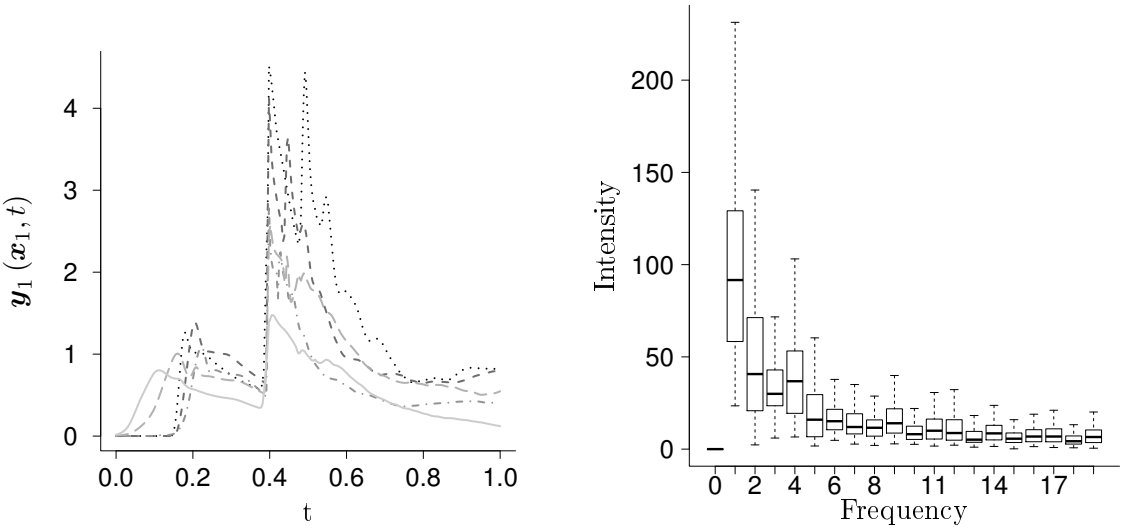
5.5.1 Description of the numerical example

The first numerical example is the coupling of two codes: a detonation code and a damped oscillator. The second code has a closed-form expression and is thus very fast to evaluate. The first code is a quick and simplified version of a detonation code.
The two codes of this application case are therefore fast to evaluate, which enables to repeat the draws and the validation of the proposed methods, thus quantifying their performance. However, their features are representative of the features of the application case which motivates this thesis:

- low dimensional vectorial input and functional output for the first code,
- low dimensional vectorial and functional inputs and functional output for the second code,
- the functional input of the second code is the output of the first code,
- the functional output of the first code can be non-smooth.

The inputs of the first code have a uniform distribution on $\mathbb{X}_1$, with:

$$\mathbb{X}_1 = [0.01, 0.02] \times [-1, 1]^3 . \tag{5.5.1}$$



(a) Time domain

(b) Frequency domain

Figure 5.1: First numerical example: Output of the first code for five different values of $\boldsymbol{x}_1$ in the time domain and for the twenty first frequencies in the frequency domain, estimated from a set of 200 observations of $\boldsymbol{y}_1$.

The variable $(\boldsymbol{x}_1)_1$ corresponds to the radius of the explosive charge, $(\boldsymbol{x}_1)_2$ to a temporal dilatation parameter, $(\boldsymbol{x}_1)_3$ to a shock magnitude parameter and $(\boldsymbol{x}_1)_4$ to an attenuation parameter.

Figure 5.1(a) shows several trajectories of the functional output of the first code. The trajectories have a similar shape, but their magnitudes can vary considerably depending on the inputs of the code, $\boldsymbol{x}_1$.
Figure 5.1(b) shows the output of the first code in the frequency domain. The highest magnitudes correspond to the lowest frequencies.

The second code is a damped oscillator, which is defined by the following second order linear differential equation:

$$\ddot{\boldsymbol{y}}_2 + 2\,(\boldsymbol{x}_2)_1\,((\boldsymbol{x}_2)_2 + \omega_0)\,\dot{\boldsymbol{y}}_2 + ((\boldsymbol{x}_2)_2 + \omega_0)^2\,\boldsymbol{y}_2 = \boldsymbol{y}_1.$$

This damped oscillator is characterized by an angular frequency ω_0. Considering the spectral density of Figure 5.1(b), three frequencies are studied: the first one with a high density (1), a second one with an intermediate density (4) and a third one with a low density (10). Consequently, we have $\omega_0 \in \{2\pi, 8\pi, 20\pi\}$. These three values of ω_0 correspond to qualitatively different relationships between the two codes.
The inputs of the second code have a uniform distribution on $\mathbb{X}_2$, with:

$$\mathbb{X}_2 = [0, 0.2] \times [-0.1, 0.1]\,. \tag{5.5.2}$$

Consequently, the definition domain of the nested code is:

$$\mathbb{X}_{\text{nest}} = [0.01, 0.02] \times [-1, 1]^3 \times [0, 0.2] \times [-0.1, 0.1]\,. \tag{5.5.3}$$

Moreover, the basis functions of the mean function of the Gaussian processes are defined as follows:

$$\boldsymbol{h}_i\left(\bar{\boldsymbol{x}}_i\right) = \left(1, \bar{\boldsymbol{x}}_i\right). \tag{5.5.4}$$

Note that an increase of the order of the polynomials does not yield more accurate prediction. Figure 5.2 shows several trajectories of the output of the second code for the three studied frequencies. For a given frequency, the trajectories have a similar shape, but their magnitudes vary depending on the inputs of the code.

5.5.2 Dimension reduction of the functional input of the second code

In this section we study the previously mentioned dimension reduction methods of the functional input of the second code.
Figure 5.3 compares the error of the estimation of the covariance matrix of the output of the first code with an estimation from the observations only (see Eq. (5.2.5)) or from the mean of a Kriging predictor constructed using the observations (see Eq. (5.2.7)). The estimation error is given by $\left\|\widehat{\boldsymbol{R}} - \boldsymbol{R}_{\boldsymbol{y}_1}\right\|_F^2$ where $\boldsymbol{R}_{\boldsymbol{y}_1}$ is the reference covariance matrix and $\widehat{\boldsymbol{R}}$ the estimation of the covariance matrix. The reference covariance matrix is estimated from 10^3 observations of the output of the first code. The figure shows that the estimation with the predictor of the first code is more accurate than the estimation from the small set of observations.

Given that $\sum_{i=1}^{n} L\left(\boldsymbol{y}_1^{(i)}\right)^T L\left(\boldsymbol{y}_1^{(i)}\right)$ is not always invertible, the estimation of the vector $\boldsymbol{a}$ given by Eq. (5.2.13) is not always numerically stable. In what follows, we study two methods of regularization of this matrix in order to invert it. In the case of a regularization, the estimate of $\boldsymbol{a}^*$ given by Eq. (5.2.13) can be rewritten:

$$\boldsymbol{a}^* = \boldsymbol{a}\left(\boldsymbol{D}, \boldsymbol{Y}_1^{\text{obs}}, \boldsymbol{Y}_2^{\text{obs}}\right) = \left(\sum_{i=1}^{n} L\left(\boldsymbol{y}_1^{(i)}\right)^T L\left(\boldsymbol{y}_1^{(i)}\right) + \boldsymbol{D}\right)^{-1} \sum_{i=1}^{n} L\left(\boldsymbol{y}_1^{(i)}\right)^T \boldsymbol{y}_2^{(i)}, \tag{5.5.5}$$

with $\boldsymbol{D}$ a positive-definite $(N_t \times N_t)$-dimensional matrix. Consequently, the following optimization problem is solved:

$$\boldsymbol{a}^* = \underset{\boldsymbol{a}\in\mathbb{R}^{N_t}}{\operatorname{argmin}} \sum_{i=1}^{n} \left\|\boldsymbol{y}_2^{(i)} - L\left(\boldsymbol{y}_1^{(i)}\right)\boldsymbol{a}\right\|^2 + \left\|\boldsymbol{D}^{\frac{1}{2}}\boldsymbol{a}\right\|^2. \tag{5.5.6}$$

Two parametric forms are studied for the definition of the matrix $\boldsymbol{D}$:

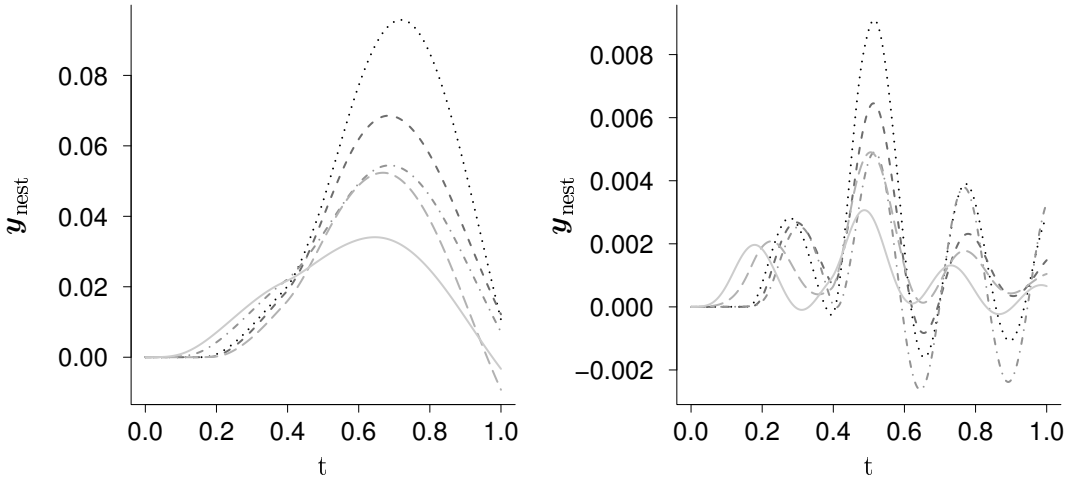
$$\boldsymbol{D} = \delta \boldsymbol{I}_{N_t},\ \delta \in \mathbb{R}_+, \tag{5.5.7}$$

and:

$$\boldsymbol{D} = \operatorname{diag}\left(\beta_0 + \beta_1\left(\frac{1}{N_t}, \frac{2}{N_t}, \ldots, \frac{N_t - 1}{N_t}, 1\right)\right),\ \beta_0 \in \mathbb{R}_+,\ \beta_1 \in \mathbb{R}_+. \tag{5.5.8}$$

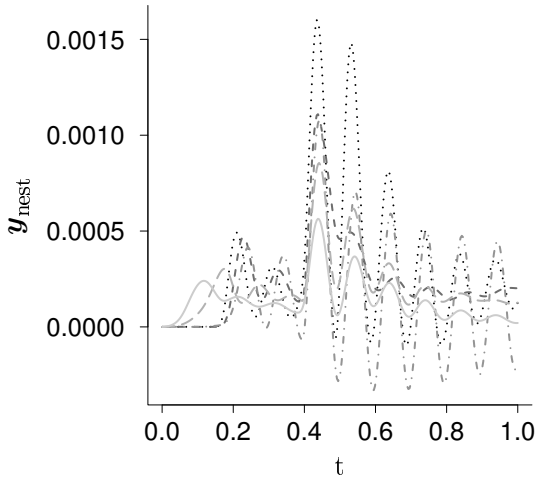
This second increasing regularization is proposed because the considered system is a damped system, and thus $i \mapsto a_i$ is a decreasing function.
By noting that $L\left(\boldsymbol{a}\right)\boldsymbol{y}_1 = L\left(\boldsymbol{y}_1\right)\boldsymbol{a}$, the parameters δ, β_0 and β_1 can be estimated with Cross Validation criteria of the form:

$$\underset{\delta\in\mathbb{R}_+}{\operatorname{argmin}} \sum_{i=1}^{n} \left\|\boldsymbol{y}_2^{(i)} - L\left(\boldsymbol{y}_1^{(i)}\right)\boldsymbol{a}\left(\delta \boldsymbol{I}_{N_t}, \left(\boldsymbol{Y}_1^{\text{obs}}\right)_{-i}, \left(\boldsymbol{Y}_2^{\text{obs}}\right)_{-i}\right)\right\|^2, \tag{5.5.9}$$



(a) $\omega_0 = 2\pi$

(b) $\omega_0 = 8\pi$



(c) $\omega_0 = 20\pi$

Figure 5.2: First numerical example: Output of the nested code for five values of $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ and three values of ω_0.

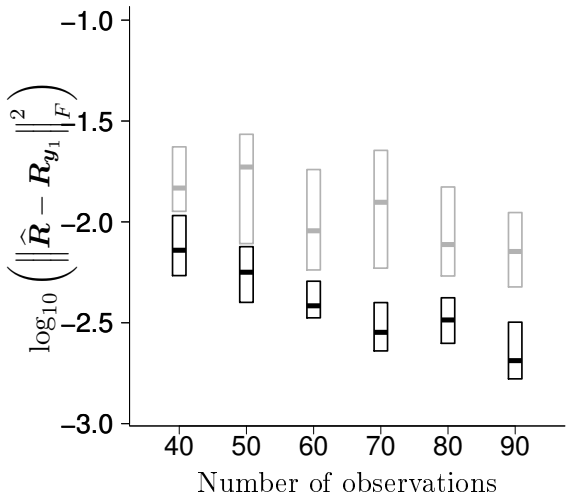


Figure 5.3: First numerical example: Boxplots of the estimation error of the covariance matrix of the output of the first code. The estimation error is given by $\left\|\widehat{\boldsymbol{R}}-\boldsymbol{R}_{\boldsymbol{y}_1}\right\|_F^2$ where $\boldsymbol{R}_{\boldsymbol{y}_1}$ is the reference covariance matrix and $\widehat{\boldsymbol{R}}$ the estimation of the covariance matrix. The reference covariance matrix is estimated from 10^3 observations of the first code output. The covariance matrices have been estimated either from the observations (see Eq. (5.2.5))(grey) or from the mean of a Gaussian predictor constructed from the observations (see Eq. (5.2.7)) (black). The designs of increasing size contain observations which are independently drawn according to $\mu_{\mathbb{X}_1}$.

and:

$$\underset{\beta_0,\beta_1\in\mathbb{R}_+^2}{\operatorname{argmin}} \sum_{i=1}^{n} \left\| \boldsymbol{y}_2^{(i)} - L\left(\boldsymbol{y}_1^{(i)}\right) \boldsymbol{a}\left(\operatorname{diag}\left(\beta_0 + \beta_1\left(\frac{1}{N_t},\ldots,1\right)\right), \left(\boldsymbol{Y}_1^{\text{obs}}\right)_{-i}, \left(\boldsymbol{Y}_2^{\text{obs}}\right)_{-i}\right)\right\|^2, \tag{5.5.10}$$

with $\left(\boldsymbol{Y}_1^{\text{obs}}\right)_{-i}$ and $\left(\boldsymbol{Y}_2^{\text{obs}}\right)_{-i}$ denoting the observations of the functional input and output of the second code, except the i-th.

Figure 5.4 shows the error $\dfrac{\|\boldsymbol{y}_2 - L(\boldsymbol{a}^*)\,\boldsymbol{y}_1\|^2}{\|\boldsymbol{y}_2 - \bar{\boldsymbol{y}}_2\|^2}$ of the linear sparse model compared to the output of the code $\boldsymbol{y}_2$. The two regularization methods of Eqs. (5.5.7) and (5.5.8) are studied. The linear regularization of Eq. (5.5.8) leads to the most accurate prediction of the linear model.
Given that the regularization matrix $\boldsymbol{D}$ of Eq. (5.5.8) leads to the most accurate linear model of $\boldsymbol{y}_2$, this regularization matrix will be chosen for the estimation of $\boldsymbol{a}$ in the remainder of the numerical examples. Indeed, the more accurate the linear model is, the more accurate the projection basis based on this linear model will be (see Proposition 5.2.1).

We assume that a set of N validation points of the codes is available. The prediction errors of the second code and the nested code can thus be defined as:

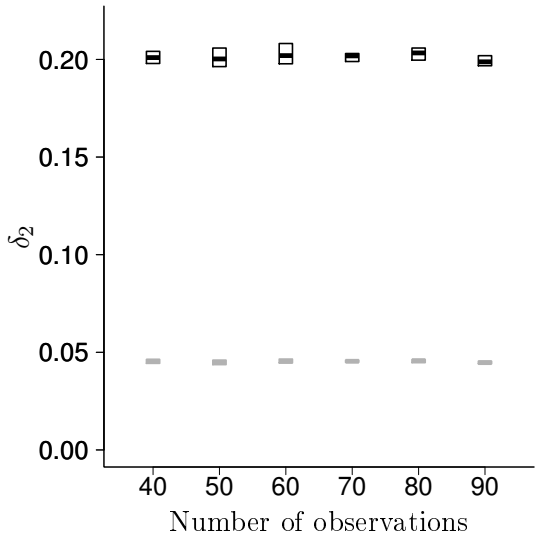
$$\Delta_2 = \frac{\sum_{i=1}^{N} \left\| \boldsymbol{y}_2\left(\boldsymbol{\varphi}_1^{(val,i)}, \boldsymbol{x}_2^{(val,i)}\right) - \boldsymbol{\mu}_2^c\left(r_m\left(\boldsymbol{\varphi}_1^{(val,i)}\right), \boldsymbol{x}_2^{(val,i)}\right)\right\|^2}{\sum_{i=1}^{N} \left\| \boldsymbol{y}_2\left(\boldsymbol{\varphi}_1^{(val,i)}, \boldsymbol{x}_2^{(val,i)}\right) - \frac{1}{N_2}\sum_{j=1}^{N} \boldsymbol{y}_2\left(\boldsymbol{\varphi}_1^{(val,j)}, \boldsymbol{x}_2^{(val,j)}\right)\right\|^2}, \tag{5.5.11}$$

and

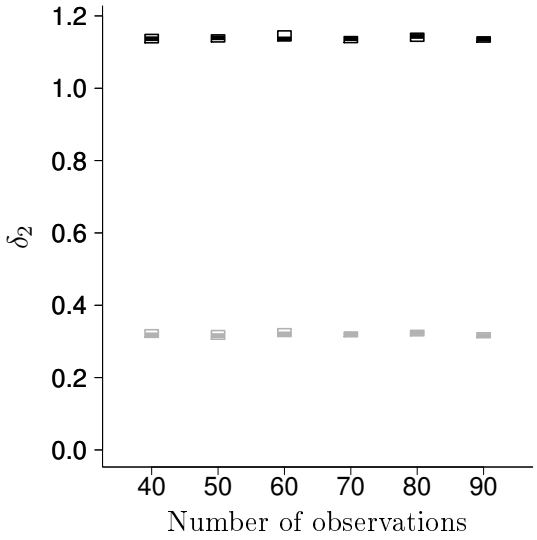
$$\Delta_{\text{nest}} = \frac{\sum_{i=1}^{N} \left\| \boldsymbol{y}_{\text{nest}}\left(\boldsymbol{x}_{\text{nest}}^{(val,i)}\right) - \boldsymbol{\mu}_{\text{nest}}^c\left(\boldsymbol{x}_{\text{nest}}^{(val,i)}\right)\right\|^2}{\sum_{i=1}^{N} \left\| \boldsymbol{y}_{\text{nest}}\left(\boldsymbol{x}_{\text{nest}}^{(val,i)}\right) - \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{y}_{\text{nest}}\left(\boldsymbol{x}_{\text{nest}}^{(val,j)}\right)\right\|^2}, \tag{5.5.12}$$

where $\boldsymbol{x}_{\text{nest}}^{(val,j)} = \left(\boldsymbol{x}_1^{(val,j)}, \boldsymbol{x}_2^{(val,j)}\right)$ and $\boldsymbol{\varphi}_1^{(val,j)} = \boldsymbol{y}_1\left(\boldsymbol{x}_1^{(val,j)}\right)$. The observations of the inputs $\{\boldsymbol{x}_{\text{nest}}^{(val,j)}, 1 \le j \le N\}$ are independently drawn according to $\mu_{\mathbb{X}_1} \times \mu_{\mathbb{X}_2}$.

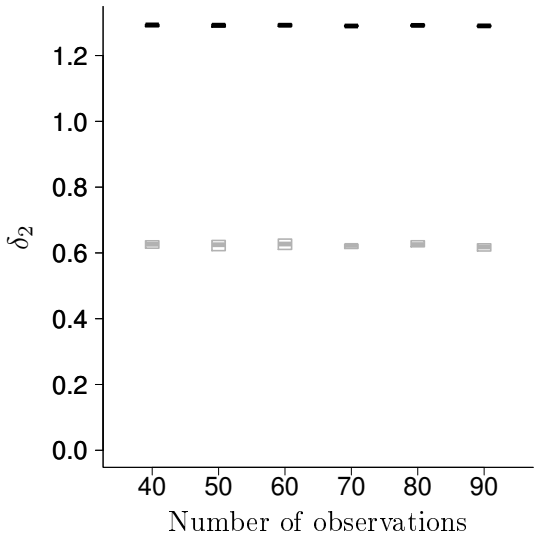
Figure 5.5 shows the prediction error Δ_2 (defined by Eq. (5.5.11)) of the Gaussian predictor $\boldsymbol{Y}_2^c$ of the second code. Three projection functions r_m are studied (see Eqs. (5.2.15), (5.2.16) and (5.2.17)). The first one is based on a Principal Components Analysis performed using the covariance matrix defined by Eq. (5.2.5). The second one is based on Partial Least Squares Regression. The third one is based on the matrix $\boldsymbol{Z}^*$ of Proposition 5.2.1 with a linear model of the form $L(\boldsymbol{a})\,\boldsymbol{y}_1$ and $\boldsymbol{a}$ estimated from Eq. (5.5.5) with the regularization matrix of Eq. (5.5.8). Two cases are considered: in the first case, the projection bases are estimated from a large set of 10^3 observations of the functional input and output of the second code; in the second case, the projection bases are estimated from the observations used for the construction of the predictor.
In both cases, the dimension reduction of Proposition 5.2.1 based on the linear model of the form $L(\boldsymbol{a})\,\boldsymbol{y}_1$ leads to the most accurate prediction.
Besides, the accuracy of the prediction with the dimension reductions based on PCA and on Proposition 5.2.1 with the linear model of the form $L(\boldsymbol{a})\,\boldsymbol{y}_1$ is quite the same if the projection basis is estimated from a small number of observations or from a high number of observations.



(a) $\omega_0 = 2\pi$



(b) $\omega_0 = 8\pi$



(c) $\omega_0 = 20\pi$

Figure 5.4: First numerical example: Boxplots of the error $\delta_2 = \dfrac{\|\boldsymbol{y}_2 - L(\boldsymbol{a}^*)\,\boldsymbol{y}_1\|^2}{\|\boldsymbol{y}_2 - \bar{\boldsymbol{y}}_2\|^2}$ depending on the regularization matrix used for the estimation of $\boldsymbol{a}$: the one defined in Eq. (5.5.7) (black) or the one defined in Eq. (5.5.8) (grey). The estimation of $\boldsymbol{a}^*$ from draws according to $\mu_{\mathbb{X}_1} \times \mu_{\mathbb{X}_2}$ of increasing size is repeated 20 times.

Thanks to the increase of the rank of the covariance matrix of $\boldsymbol{y}_1$ with a predictor of the first code and the sparse structure of the linear model, these two projection bases can be estimated accurately with a relatively small number of observations.
The projection basis based on Partial Least Squares is much more sensitive to the decrease in the number of observations from which it is estimated.
Finally, the size of the projection bases is estimated with the Leave One Out criterion of Eq. (5.3.23).

Figure 5.6 shows the size of the bases estimated with the Cross-Validation criterion of Eq. (5.3.23) for the three studied dimension reduction methods presented in Figure 5.5. The size of the projection basis varies less and is generally smaller with the dimension reduction of Proposition 5.2.1 with a linear model of the form $L(\boldsymbol{a}^*)\boldsymbol{y}_1$ and $\boldsymbol{a}^*$ given by Eqs. (5.5.5) and (5.5.8).

In the remainder of the section, the dimension reduction of the functional intermediary variable will be performed using the method defined in Proposition 5.2.1. A linear model of the form $L(\boldsymbol{a})\boldsymbol{y}_1$ will be considered, and $\boldsymbol{a}$ will be estimated using Eqs. (5.5.5) and (5.5.8). Indeed, this method leads to the most accurate linear approximation of $\boldsymbol{y}_2$. The more accurate the linear model is, the more appropriate the dimension reduction based on this linear model will be.
Following the results of Figure 5.6, the following values of m are considered:

$$m = \begin{cases} 4, \text{ if } \omega_0 = 2\pi, \\ 7, \text{ if } \omega_0 = 8\pi, \\ 6-7, \text{ if } \omega_0 = 20\pi. \end{cases} \tag{5.5.13}$$

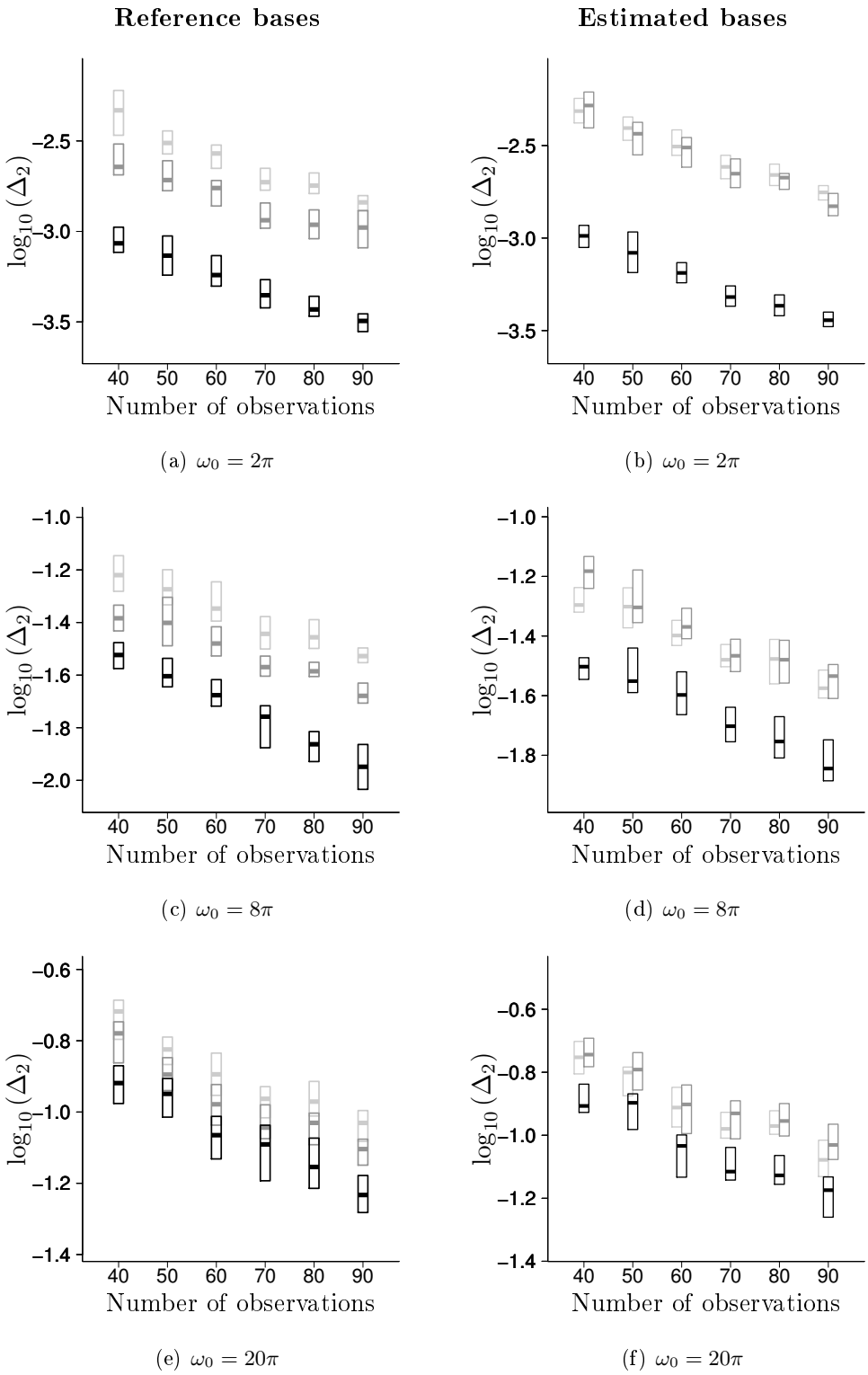
The covariance matrix of the functional input of the second code will be computed using a predictor of the output of the first code, as defined in Eq. (5.2.7). Finally, the size m of the projection basis is estimated with the Cross-Validation criterion of Eq. (5.3.23).

5.5.3 Prediction of the nested code

Once the dimension of the functional intermediary variable has been efficiently reduced, we can study the surrogate modeling of the nested code. We first define the reference method which corresponds to the case where the nested code is considered to be a single code.
This reference method is called "blind-box". With this method, a surrogate model of $\boldsymbol{x}_{\text{nest}} \mapsto \boldsymbol{y}_{\text{nest}}$ is constructed using the formalism of Section 5.3. This blind-box method cannot take into account the intermediary observations of the nested code.
The "linearized" method refers to the construction of the predictor of the nested code proposed in Proposition 5.4.2.

Figure 5.7 compares the prediction error Δ_{nest} (see Eq. (5.5.12)) for predictors of the nested code constructed with the linearized and the blind-box methods. The prediction accuracy is better with the linearized method.

Figure 5.8 shows the prediction error Δ_{nest} (see Eq. (5.5.12)) along the sequential designs of Definition 1. A reference corresponding to the case of the blind-box predictor with an I-optimal sequential design (see Eq. (5.3.31) and Proposition 5.3.1) is also shown. All the sequential designs have the same initial designs drawn according to $\mu_{\mathbb{X}_1} \times \mu_{\mathbb{X}_2}$.
It can be seen that both proposed sequential designs with the linearized predictor enable to obtain a better prediction accuracy than the blind-box predictor with an I-optimal design.

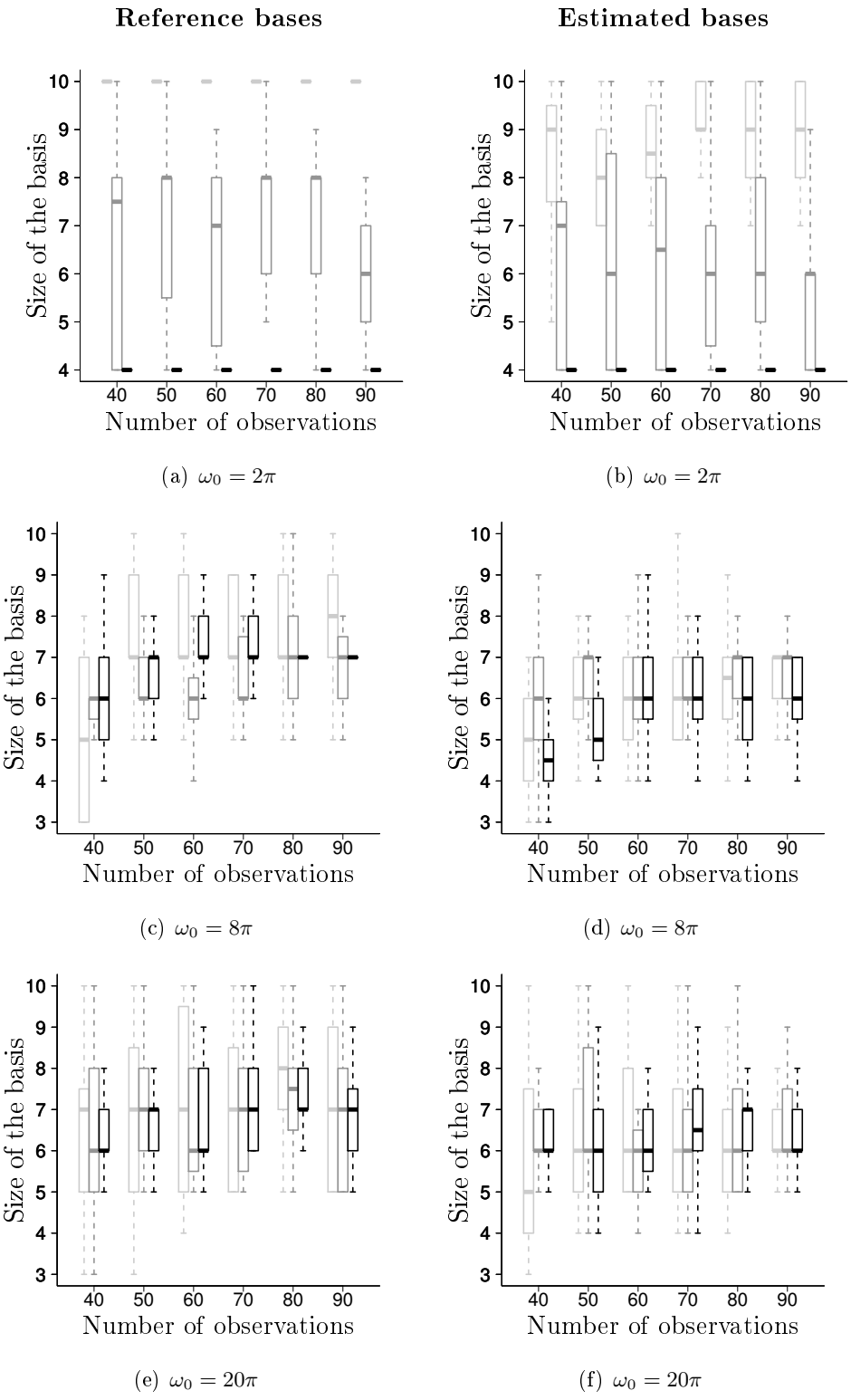


(a) $\omega_0 = 2\pi$ (b) $\omega_0 = 2\pi$

(c) $\omega_0 = 8\pi$ (d) $\omega_0 = 8\pi$

(e) $\omega_0 = 20\pi$ (f) $\omega_0 = 20\pi$

Figure 5.5: First numerical example: Prediction error of the output of the second code Δ_2 (see Eq. (5.5.11)) in log scale. The observations are independently drawn according to $\mu_{\mathbb{X}_1} \times \mu_{\mathbb{X}_2}$ and the designs are of increasing size. Three types of projection functions r_m are considered. The first one is based on Principal Components Analysis (light grey), the second one is based on PLS regression (mid-grey) and the third one is based on the projection of Proposition 5.2.1 with a linear model of the form $L(\boldsymbol{a}^*)\boldsymbol{y}_1$ where $\boldsymbol{a}^*$ is given by Eqs. (5.5.5) and (5.5.8) (black). The size of the projection basis is estimated with the Leave One Out criterion of Eq. (5.3.23). The reference bases are computed from a set of 1000 observations of the functional input and output of the second code. The estimated bases are estimated from the observations of the designs.

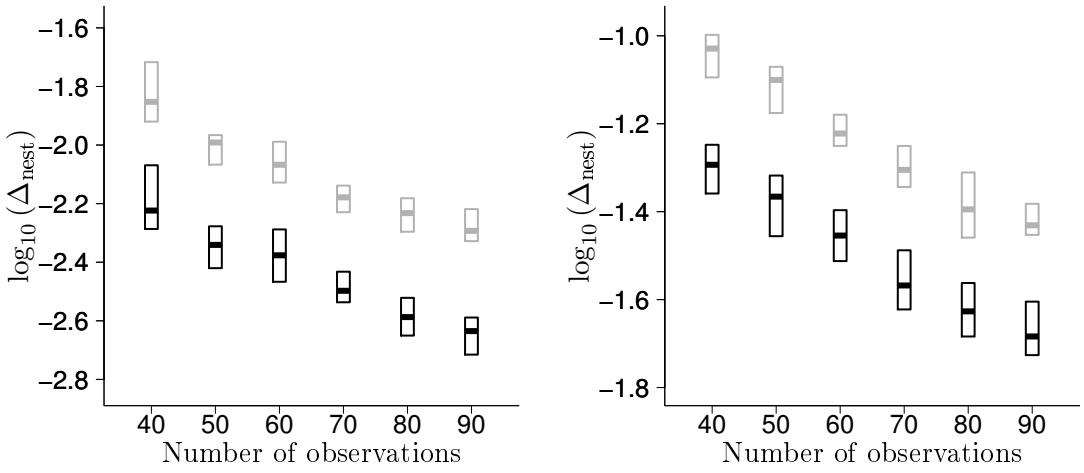


(a) $\omega_0 = 2\pi$ (b) $\omega_0 = 2\pi$

(c) $\omega_0 = 8\pi$ (d) $\omega_0 = 8\pi$

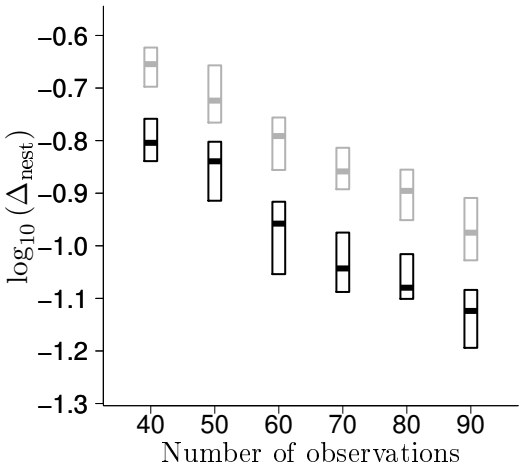
(e) $\omega_0 = 20\pi$ (f) $\omega_0 = 20\pi$

Figure 5.6: First numerical example: Sizes of the projection bases estimated with the Leave One Out criterion of Eq. (5.3.23). The observations are independently drawn according to $\mu_{\mathbb{X}_1} \times \mu_{\mathbb{X}_2}$ and the designs are of increasing size. Three types of projection functions r_m are considered. The first one is based on Principal Components Analysis (light grey), the second one is based on PLS regression (mid-grey) and the third one is based on the projection of Proposition 5.2.1 with a linear model of the form $L(\boldsymbol{a}^*)\boldsymbol{y}_1$ where $\boldsymbol{a}^*$ is given by Eqs. (5.5.5) and (5.5.8) (black). The study has been restricted to $1 \leq m \leq 10$. The reference bases are computed from a set of 1000 observations of the functional input and output of the second code. The estimated bases are estimated from the observations of the designs.



(a) $\omega_0 = 2\pi$

(b) $\omega_0 = 8\pi$



(c) $\omega_0 = 20\pi$

Figure 5.7: First numerical example: Boxplots of the prediction error Δ_{nest} (see Eq. (5.5.12)) for the linearized (black) and the blind-box (grey) predictors. The observations are drawn according to $\mu_{\mathbb{X}_1} \times \mu_{\mathbb{X}_2}$ and the designs are of increasing size. The draws are repeated 20 times. The dimension reduction of the intermediary variable is performed using the method of Proposition 5.2.1 and a linear approximation of the form $L(\boldsymbol{a}^*)\boldsymbol{y}_1$ where $\boldsymbol{a}^*$ is given by Eqs. (5.5.5) and (5.5.8).

Furthermore, the best I-optimal criterion leads to an improved prediction accuracy compared to the Chained I-optimal design. Note that with the Best I-optimal criterion, the number of code evaluations of the two codes can be different.

Figure 5.9 shows the distribution of the number of code evaluations of the two codes along the Best I-optimal sequential design. The first new observation points are observations of the first code. However, the distribution of the number of observations of the two codes can vary a lot depending on the angular frequency of the oscillator of the second code.

For the angular frequency equal to 2π, there are more evaluations of the first code. For the angular frequency equal to 8π, the number of evaluations is almost the same for both codes. For the angular frequency equal to 20π, the number of evaluations is almost the same for both codes at the beginning, then the number of observations of the second code increases faster than those of the first code.

Finally, Figure 5.10 shows the prediction accuracy along the proposed sequential designs for different computational costs of the two codes. The total cost of the two codes is the same, but three cases are considered for the distribution of the cost between the codes. Only the angular frequency $\omega_0 = 2\pi$ of the oscillator of the second code is considered, because a very different number of observations of the codes 1 and 2 has been observed in Figure 5.9. On the three figures are presented the prediction errors Δ_{nest} (see Eq. (5.5.12)) for the Chained I-optimal design and the Best I-optimal design. The prediction error of the Chained I-optimal design is the same on all the figures, because the total cost of the two codes is the same. On the contrary, the prediction error with the Best I-optimal design depends on the distribution of the costs between the two codes. The lower the cost of the first code is, the lower the prediction error is. These results are in accordance with those of Figure 5.9, which show that the new observations are mostly added to the first code.

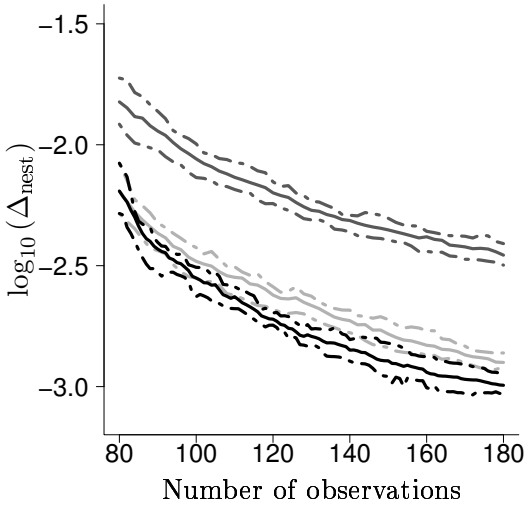
Once an accurate predictor of the nested code has been obtained at the end of the sequential design (Best I-optimal), the predictor can be used in order to perform a sensitivity analysis (see Section 1.6). A Kriging Believer [Picheny et al., 2010] approach will be used, which means that $\boldsymbol{y}_{\text{nest}}$ will be replaced by $\boldsymbol{\mu}^c_{\text{nest}}$ for the computation of the Sobol indices for the first Principal Component of $\boldsymbol{y}_{\text{nest}}$.

5.5.4 Sensitivity analysis

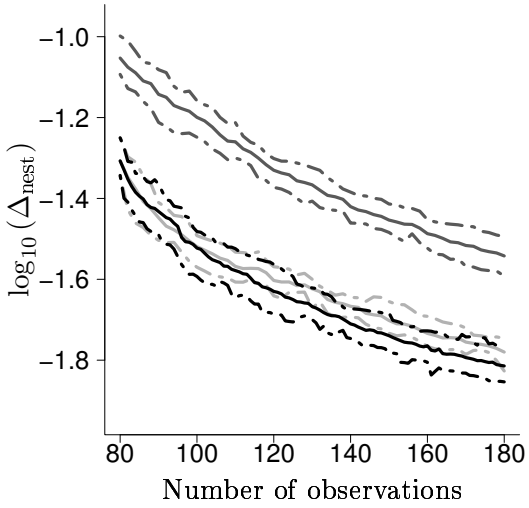
Figure 5.11 presents the first-order and total Sobol indices (see Section 1.6 for further details) for the first Principal Component of the output of the nested code. The figures show that the first-order index is very close to the total index. The first-order effects are therefore more important than the interaction effects. This means that an approximation of the form:

$$f_0 + \sum_{i=1}^{d_1+d_2} f_i\left((\boldsymbol{x}_{\text{nest}})_i\right), \tag{5.5.14}$$

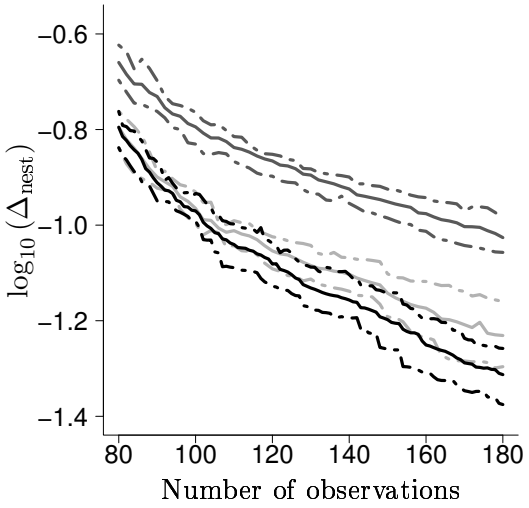
with f_i defined by Eq. (1.6.3), can provide a good approximation of the first Principal Component of $\boldsymbol{y}_{\text{nest}}$. Besides, the indices of the first input of the first code are very high compared to those of the other inputs. The radius of the explosive charge is thus a very sensitive input for the first Principal Component of the output of the code.



(a) $\omega_0 = 2\pi$

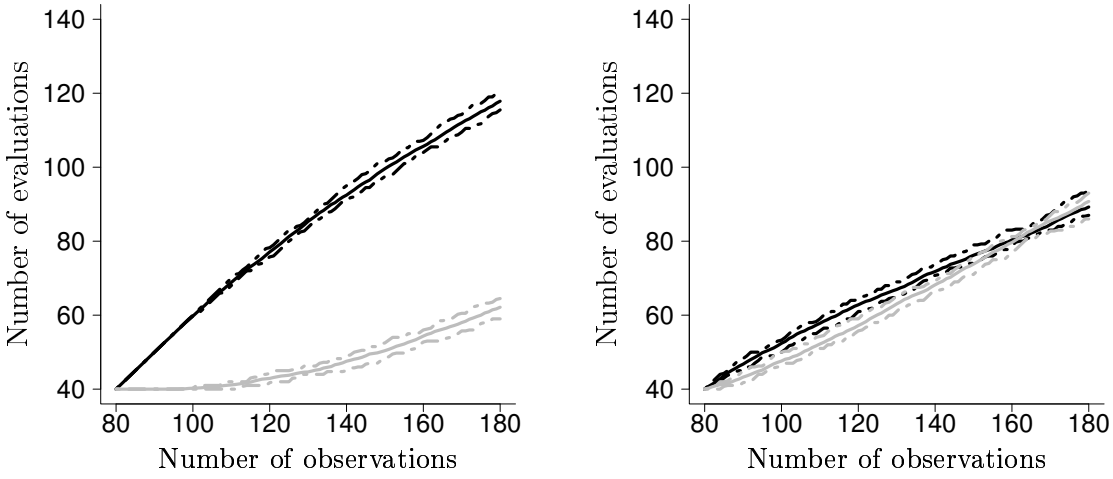


(b) $\omega_0 = 8\pi$



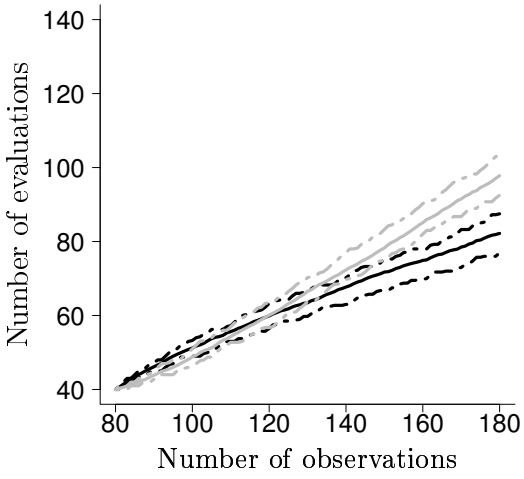
(c) $\omega_0 = 20\pi$

Figure 5.8: First numerical example: Error of the nested code prediction along the sequential designs, in log scale. The initial designs are 40-points designs, drawn according to $\mu_{\mathbb{X}_1} \times \mu_{\mathbb{X}_2}$ and are the same for the three series. The draw of the initial design is repeated 20 times. The solid lines represent the median and the two-dashed lines represent the first and the third quartiles. The three series correspond to the blind-box predictor with an I-optimal design (mid-grey), the linearized predictor with a Chained I-optimal design (light grey) and with a Best I-optimal design (black). The computational cost of both codes is the same: $\tau_1 = \tau_2$.



(a) $\omega_0 = 2\pi$

(b) $\omega_0 = 8\pi$



(c) $\omega_0 = 20\pi$

Figure 5.9: First numerical example: Boxplots of the number of evaluations of the first code (black) and the second code (grey) along the Best I-optimal sequential design.

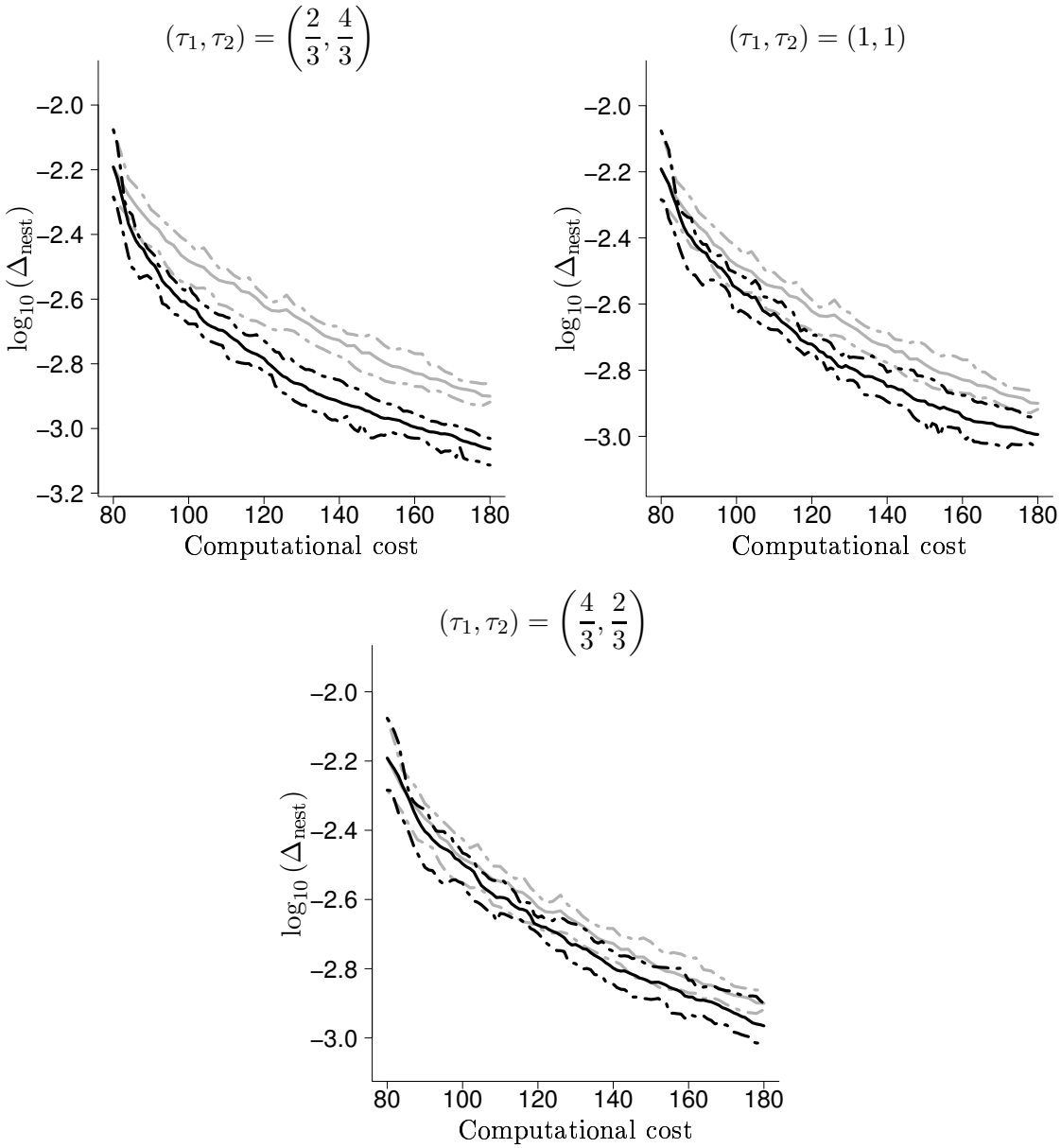
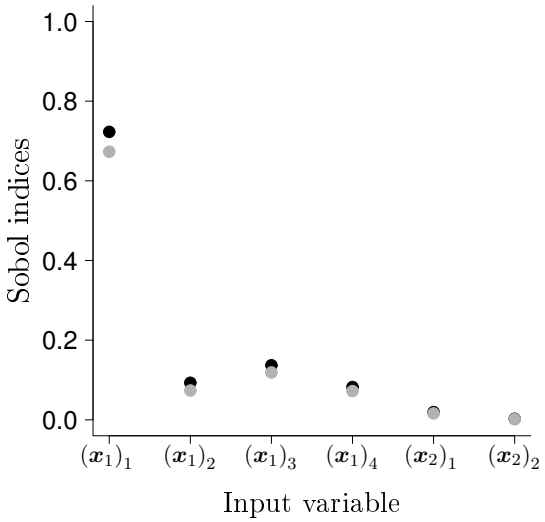
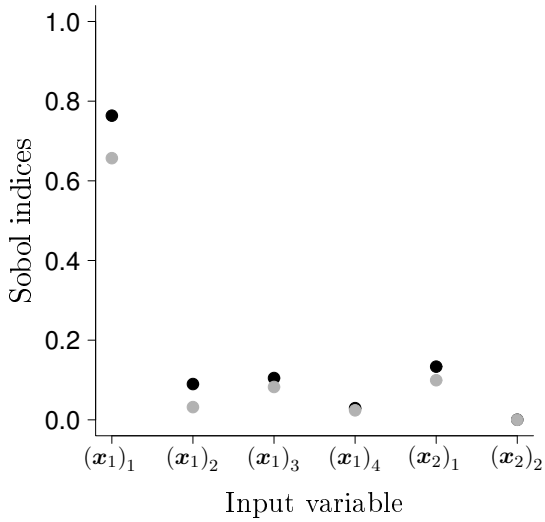


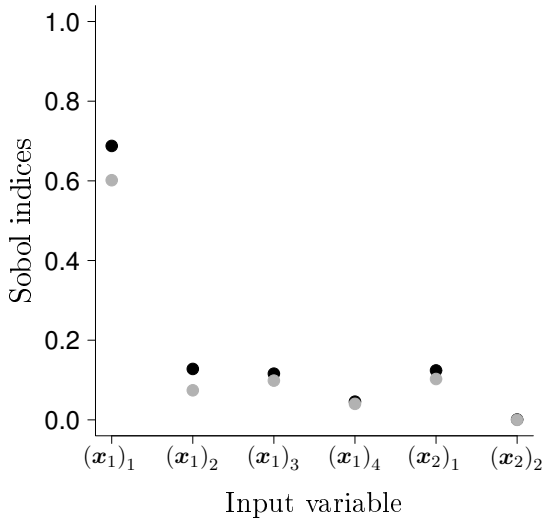
Figure 5.10: First numerical example: Error of the nested code prediction along the sequential designs, in log scale. The initial designs are 40-points designs, drawn according to $\mu_{\mathbb{X}_1} \times \mu_{\mathbb{X}_2}$ and are the same for the three series. The draw of the initial design is repeated 20 times. The solid lines represent the median and the two-dashed lines represent the first and the third quartiles. On each plot, the two series correspond to the linearized predictor with a Chained I-optimal design (black) and with a Best I-optimal design (light grey). Three cases are studied for the computational costs of the codes: $\frac{\tau_1}{\tau_2} = \frac{1}{2}$, $\frac{\tau_1}{\tau_2} = 1$, and $\frac{\tau_1}{\tau_2} = 2$. The total cost of the two codes is the same in the three cases. The angular frequency of the second code is $1 \times 2\pi$.



(a) $\omega_0 = 2\pi$

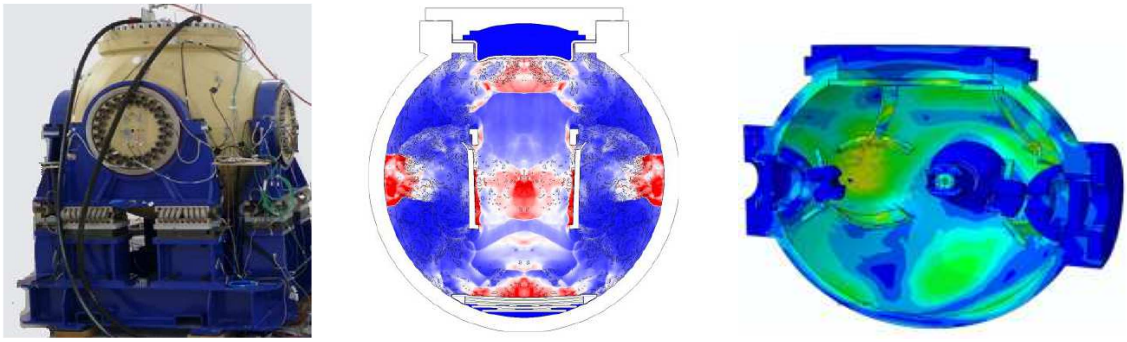


(b) $\omega_0 = 8\pi$



(c) $\omega_0 = 20\pi$

Figure 5.11: First numerical example: Sobol indices of the first Principal Component of the nested code output: first-order index in grey and total index in black. The points are drawn according to a Best I-optimal design of total cost 180 and an initial design drawn according to $\mu_{\mathbb{X}_1} \times \mu_{\mathbb{X}_2}$ of cost 80.



(a) Tank (b) First code (c) Second code

Figure 5.12: Second numerical example: Images from Defaux and Evrard [2014]. The left-hand plot shows the tank. The middle plot shows the output of the first code, with the detonation products in white and red. The right-hand plot shows the output of the second code, the Von Mises stress at a given time at the inner surface of the tank. The areas in green are under stress and those in blue are not under stress.

5.6 Second numerical example

5.6.1 Description of the numerical example

This second study corresponds to the motivating industrial problem. We aim at performing a sensitivity analysis of the output of a chain of two computer codes with respect to its inputs. The two codes model the explosion of an explosive charge into the spherical tank presented in Figure 5.12.
This first code is an Eulerian 2-dimensional code, which simulates the burn of the explosive charge, the internal gas dynamics and the shock wave propagation. Its inputs are the same than those of the first code of the first test case. The second code provides the mechanical response of the tank through the von Mises stress test at the inner surface of the tank.

The spherical tank is presented in on the left-hand plot of Figure 5.12. The output of the first code is shown on the middle plot of Figure 5.12. The right-hand plot of Figure 5.12 shows the second code output.

In this work, in order to reduce the computational cost of a code evaluation, we use a 1-dimensional approximation of the first code, by considering that the explosion and the tank are spherically symmetric. The output of the first code corresponds here to the spherically symmetric time-varying shock-wave.
In the same way, a 2-dimensional approximation of the second code is considered. The approximation is based on an axisymmetry of the mechanical response of the tank. Then the maximum value of the Von Mises stress over the surface (in fact a curve) for each time step is considered. The time-varying output of the second code is therefore this time-varying maximal value.
Moreover, the elastic limits of the tank and the tap have been set to an artificially high value in order to obtain unsaturated output trajectories.

The components of $\boldsymbol{x}_2$ are independent and of dimension 4. Table 5.1 presents the input variables of the second code and their distributions.

Input name	Description and unit	Distribution of the input	95% confidence interval
$(\boldsymbol{x}_2)_1$	Internal radius of the tank (m)	$\mathcal{N}\left(0.72, 0.005^2\right)$	$[0.71, 0.73]$
$(\boldsymbol{x}_2)_2$	Thickness (m)	$\mathcal{N}\left(0.073, 0.0015^2\right)$	$[0.07, 0.076]$
$(\boldsymbol{x}_2)_3$	Young modulus of the tank (Pa)	$\mathcal{N}\left(2.1\ 10^{11}, \left(2.1\ 10^{10}\right)^2\right)$	$[1.69, 2.51] \times 10^{11}$
$(\boldsymbol{x}_2)_4$	Young modulus of the tap (Pa)	$\mathcal{N}\left(2.1\ 10^{11}, \left(2.1\ 10^{10}\right)^2\right)$	$[1.69, 2.51] \times 10^{11}$

Table 5.1: Input parameters of the second code in the second numerical example. All variables are independent.

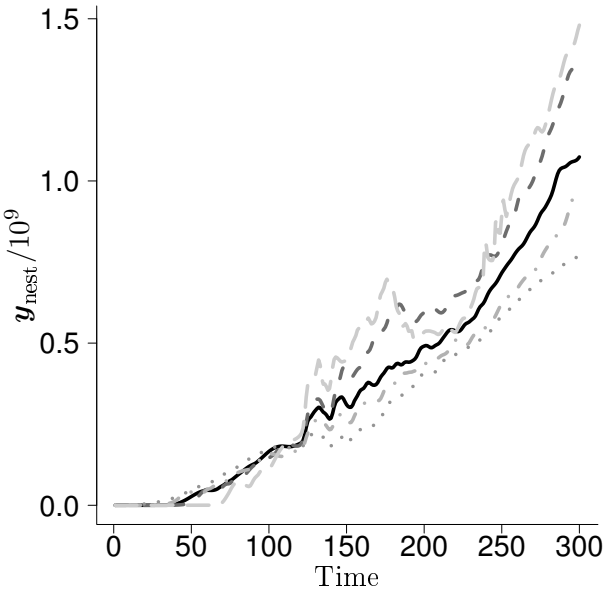


Figure 5.13: Second numerical example: Example of trajectories of the output of the nested code for five values of $(\boldsymbol{x}_1, \boldsymbol{x}_2)$.

Figure 5.13 shows some trajectories of the output of the nested code for this second numerical example. The trajectories have very similar shapes and are overall growing with time.

5.6.2 Results of the numerical example

Figure 5.14 shows the prediction accuracy for the output of the nested code along sequential designs. Three cases are presented. The first one is based on the blind-box predictor with an I-optimal enrichment (see Proposition 5.3.1), the other two are based on the linearized predictor of Proposition 5.4.2 with the two sequential design criteria of Definition 1. The results show that the linearized predictor is more accurate than the blind box predictor. These results illustrate the interest of taking into account the intermediary observations of the nested code.

Once an accurate predictor of the nested code has been obtained at the end of the sequential design (Best I-optimal with linearized predictor), the predictor can be used in order to perform a sensitivity analysis (see Section 1.6). A Kriging Believer approach [Picheny et al., 2010] will be used, which means that $\boldsymbol{y}_{\text{nest}}$ will be replaced by $\boldsymbol{\mu}^c_{\text{nest}}$ for the computation of the Sobol

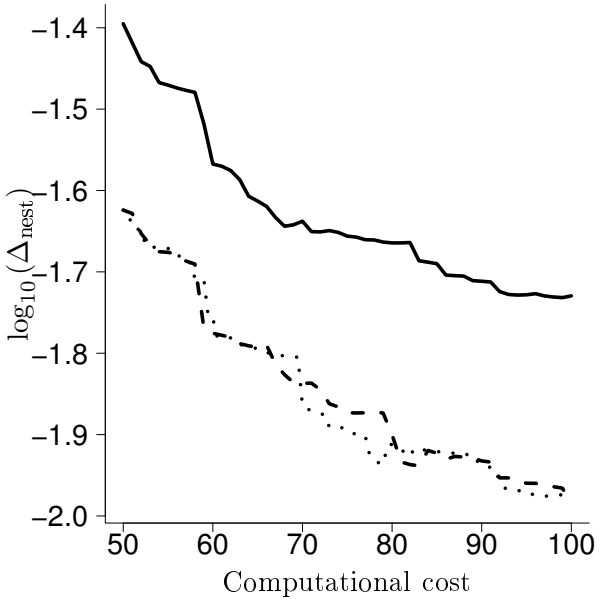


Figure 5.14: Second numerical example: Prediction error Δ_{nest} (see Eq. (5.5.12)), in log scale, for the nested code along the sequential designs: I-optimal sequential design with the blind-box predictor (solid line), Chained I-optimal sequential design with the linearized predictor (dashed line), Best I-optimal sequential design with the linearized (dotted line) predictor. The initial design of 50 points on $\mathbb{X}_{\text{nest}}$ is drawn according to $\mu_{\mathbb{X}_1} \times \mu_{\mathbb{X}_2}$.

indices of the maximum value of the high dimensional vectorial output $\boldsymbol{y}_{\text{nest}}$.

Figure 5.15 shows the Sobol indices of the maximum value of the temporal output of the nested code with respect to the inputs of the nested code. The first-order and total indices are very close. This means that the first-order effects are important and the interactions almost negligible (see Section 1.6 for further details). Besides, the indices associated with the inputs $\boldsymbol{x}_2$ of the second code are very low compared to those associated with the first code. The index associated with $(\boldsymbol{x}_1)_1$ (radius of the explosive charge) is especially high compared to those of all the other inputs. The second most influent input is the shock magnitude parameter $(\boldsymbol{x}_1)_3$.

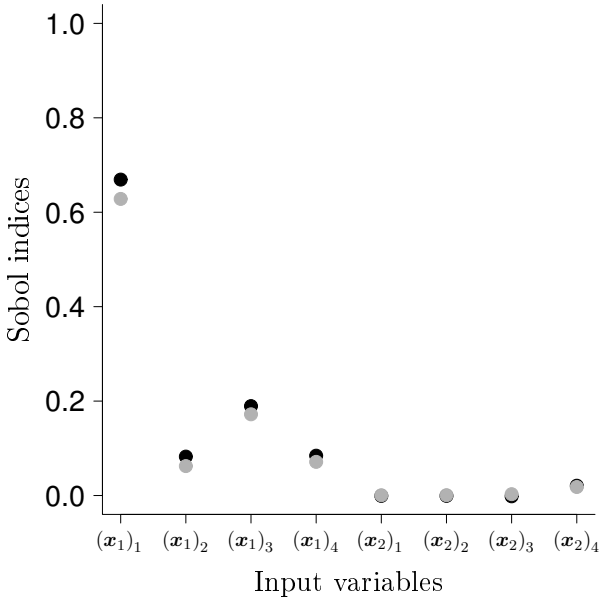


Figure 5.15: Second numerical example: Sobol indices (first-order in grey and total in black) of the maximum value of the functional output of the nested code. The indices are estimated with a predictor constructed from the set of observations at the end of the Best I-optimal sequential design.

5.7 Conclusions

In this chapter, we have focused on a system of two nested codes with functional outputs. The functional output of the first code is one of the inputs of the second code. The objective was to construct a surrogate model of the output of the nested code in order to perform analyses for the design and the certification of the studied system (like sensitivity analysis, risk analysis or optimization).
The surrogate models are constructed in the framework of Universal Kriging. However, the existing Gaussian process predictors for codes with a functional output generally consider low dimensional vectorial inputs.
Our first objective was therefore to reduce the dimension of the functional input of the second code in order to use a surrogate model of the second code which has a low dimensional vectorial input.
We have therefore developed a method to reduce the dimension of the functional input of a code which is adapted to the output of this code. This method of dimension reduction is a two-step approach. First the output of the code is approximated by a linear causal filter with respect to the functional input of the second code. This linear model has a very sparse structure and can be accurately estimated from a small number of observations. Then a dimension reduction based on this linear model is performed.
This method of dimension reduction is particularly useful when the number of observations is small compared to the number of discretization steps of the functional variables.
This appropriate dimension reduction of the functional input of the second code enables to construct a more relevant Gaussian predictor of the second code, compared to other dimension reduction techniques such as Principal Components Analysis or Partial Least Squares.
We have also seen that a well-chosen regularization can lead to a better estimation of the sparse linear model used for the proposed dimension reduction.
Furthermore, we have extended the framework proposed in Chapter 4 for the prediction of the

output of a chain of two codes with scalar outputs to the prediction of the functional output of a chain of two codes with functional outputs.
Finally, we have adapted two sequential design criteria which enable to improve the prediction accuracy of the nested code and to take into account the difference in the computational costs of the two codes.
The results obtained demonstrate the interest of taking into account the intermediary observations. Moreover, the selection criterion which enables to choose which code to launch leads to a more accurate prediction of the nested code.

5.8 Proofs

5.8.1 Proof of Proposition 5.2.1

We aim at finding a m-rank matrix $\boldsymbol{Z}^*$ such that:

$$\boldsymbol{Z}^* = \underset{\boldsymbol{Z}\in\mathcal{M}_{N_t\times N_t}}{\operatorname{argmin}} \sum_{i=1}^{N_x} \left\| \boldsymbol{A}\boldsymbol{x}_t^{(i)} - \boldsymbol{A}\boldsymbol{Z}\boldsymbol{x}_t^{(i)} \right\|^2$$

with $\boldsymbol{Z}$ a m-rank matrix and $\boldsymbol{x}_t^{(i)}$ the i-th observation of the functional input.
The previous equation can be rewritten:

$$\begin{aligned}
\boldsymbol{Z}^* = & \underset{\boldsymbol{Z}\in\mathcal{M}_{N_t\times N_t}}{\operatorname{argmin}} \sum_{i=1}^{N_x} \left\| \boldsymbol{A}\boldsymbol{x}_t^{(i)} - \boldsymbol{A}\boldsymbol{Z}\boldsymbol{x}_t^{(i)} \right\|^2, \\
= & \underset{\boldsymbol{Z}\in\mathcal{M}_{N_t\times N_t}}{\operatorname{argmin}} \operatorname{Tr}\left((\boldsymbol{A}\boldsymbol{X}_t - \boldsymbol{A}\boldsymbol{Z}\boldsymbol{X}_t)^T (\boldsymbol{A}\boldsymbol{X}_t - \boldsymbol{A}\boldsymbol{Z}\boldsymbol{X}_t) \right), \\
= & \underset{\boldsymbol{Z}\in\mathcal{M}_{N_t\times N_t}}{\operatorname{argmin}} \operatorname{Tr}\left((\boldsymbol{A}\boldsymbol{Z}^c\boldsymbol{X}_t)^T (\boldsymbol{A}\boldsymbol{Z}^c\boldsymbol{X}_t) \right), \\
= & \underset{\boldsymbol{Z}\in\mathcal{M}_{N_t\times N_t}}{\operatorname{argmin}} \operatorname{Tr}\left(\boldsymbol{X}_t^T (\boldsymbol{Z}^c)^T \boldsymbol{A}^T\boldsymbol{A}\boldsymbol{Z}^c\boldsymbol{X}_t \right), \\
= & \underset{\boldsymbol{Z}\in\mathcal{M}_{N_t\times N_t}}{\operatorname{argmin}} \operatorname{Tr}\left((\boldsymbol{Z}^c)^T \boldsymbol{A}^T\boldsymbol{A}\boldsymbol{Z}^c\boldsymbol{X}_t\boldsymbol{X}_t^T \right), \\
= & \underset{\boldsymbol{Z}\in\mathcal{M}_{N_t\times N_t}}{\operatorname{argmin}} \operatorname{Tr}\left(\left(\boldsymbol{X}_t\boldsymbol{X}_t^T\right)^{\frac{1}{2}} (\boldsymbol{Z}^c)^T \boldsymbol{A}^T\boldsymbol{A}\boldsymbol{Z}^c \left(\boldsymbol{X}_t\boldsymbol{X}_t^T\right)^{\frac{1}{2}} \right), \\
= & \underset{\boldsymbol{Z}\in\mathcal{M}_{N_t\times N_t}}{\operatorname{argmin}} \operatorname{Tr}\left(\left(\boldsymbol{X}_t\boldsymbol{X}_t^T\right)^{\frac{1}{2}} (\boldsymbol{Z}^c)^T \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{\frac{1}{2}} \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{\frac{1}{2}} \boldsymbol{Z}^c \left(\boldsymbol{X}_t\boldsymbol{X}_t^T\right)^{\frac{1}{2}} \right), \\
= & \underset{\boldsymbol{Z}\in\mathcal{M}_{N_t\times N_t}}{\operatorname{argmin}} \operatorname{Tr}\left(\left(\left(\boldsymbol{A}^T\boldsymbol{A}\right)^{\frac{1}{2}} \boldsymbol{Z}^c \left(\boldsymbol{X}_t\boldsymbol{X}_t^T\right)^{\frac{1}{2}} \right)^T \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{\frac{1}{2}} \boldsymbol{Z}^c \left(\boldsymbol{X}_t\boldsymbol{X}_t^T\right)^{\frac{1}{2}} \right), \\
= & \underset{\boldsymbol{Z}\in\mathcal{M}_{N_t\times N_t}}{\operatorname{argmin}} \left\| \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{\frac{1}{2}} \boldsymbol{Z}^c \left(\boldsymbol{X}_t\boldsymbol{X}_t^T\right)^{\frac{1}{2}} \right\|_F^2, \\
= & \underset{\boldsymbol{Z}\in\mathcal{M}_{N_t\times N_t}}{\operatorname{argmin}} \left\| \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{\frac{1}{2}} \left(\boldsymbol{X}_t\boldsymbol{X}_t^T\right)^{\frac{1}{2}} - \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{\frac{1}{2}} \boldsymbol{Z} \left(\boldsymbol{X}_t\boldsymbol{X}_t^T\right)^{\frac{1}{2}} \right\|_F^2,
\end{aligned}$$

where $\|\cdot\|_F^2$ denotes the Frobenius norm.
In Eckart and Young [1936], it is shown that the matrix $\boldsymbol{Q}_m$ of rank m which minimizes $\left\| \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{\frac{1}{2}} \left(\boldsymbol{X}_t\boldsymbol{X}_t^T\right)^{\frac{1}{2}} - \boldsymbol{Q}_m \right\|_F^2$ is given by the m first singular-values of the singular-value decomposition of $\left(\boldsymbol{A}^T\boldsymbol{A}\right)^{\frac{1}{2}} \left(\boldsymbol{X}_t\boldsymbol{X}_t^T\right)^{\frac{1}{2}}$. If we denote by $\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$ the singular-value decomposition of $\left(\boldsymbol{A}^T\boldsymbol{A}\right)^{\frac{1}{2}} \left(\boldsymbol{X}_t\boldsymbol{X}_t^T\right)^{\frac{1}{2}}$ with $\boldsymbol{D}$ gathering the singular values in decreasing order, then we have $\boldsymbol{Q}_m = \boldsymbol{U}_m\boldsymbol{D}_m\boldsymbol{V}_m^T$ where $\boldsymbol{U}_m$ and $\boldsymbol{V}_m$ gather the m first columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ and $\boldsymbol{D}_m$ contain the m first lines and columns of $\boldsymbol{D}$. It can therefore be inferred that:

$$\left(\boldsymbol{A}^T\boldsymbol{A}\right)^{\frac{1}{2}} \boldsymbol{Z}^* \left(\boldsymbol{X}_t\boldsymbol{X}_t^T\right)^{\frac{1}{2}} = \boldsymbol{U}_m\boldsymbol{D}_m\boldsymbol{V}_m^T.$$

Hence, we have:

$$\boldsymbol{Z}^* = \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-\frac{1}{2}} \boldsymbol{U}_m\boldsymbol{D}_m\boldsymbol{V}_m^T \left(\boldsymbol{X}_t\boldsymbol{X}_t^T\right)^{-\frac{1}{2}}.$$

Finally, the m-dimensional projection of $\boldsymbol{x}_t$ which is optimal for the linear model is obtained by multiplying $\boldsymbol{x}_t$ by a $m \times N_t$ matrix Π defined by:

$$\Pi = \boldsymbol{V}_m^T \left(\boldsymbol{X}_t\boldsymbol{X}_t^T\right)^{-\frac{1}{2}}.$$

5.8.2 Proof of Lemma 1

5.8.2.1 First equation

From Eq. (5.3.13), it follows that:

$$\mathbb{E}\left[\boldsymbol{M}_i|\{\boldsymbol{Y}_i^{\text{obs}}, \boldsymbol{Y}_i\left(\bar{\boldsymbol{x}}_i^*\right) = \boldsymbol{\mu}_i^c\left(\bar{\boldsymbol{x}}_i^*\right)\}, C_i\right] = \begin{array}{l} \boldsymbol{Y}_i^{\text{obs,new}} \left(\boldsymbol{R}_i^{\text{obs,new}}\right)^{-1} \left(\boldsymbol{H}_i^{\text{obs,new}}\right)^T \\ \left(\boldsymbol{H}_i^{\text{obs,new}} \left(\boldsymbol{R}_i^{\text{obs,new}}\right)^{-1} \left(\boldsymbol{H}_i^{\text{obs,new}}\right)^T\right)^{-1}, \end{array}$$

where

$$\begin{aligned} \boldsymbol{Y}_i^{\text{obs,new}} &= \left(\boldsymbol{Y}_i^{\text{obs}} \ , \ \boldsymbol{\mu}_i^c\left(\bar{\boldsymbol{x}}_i^*\right)\right), \\ \boldsymbol{H}_i^{\text{obs,new}} &= \left(\boldsymbol{H}_i^{\text{obs}} \ , \ \boldsymbol{h}_i\left(\bar{\boldsymbol{x}}_i^*\right)\right), \end{aligned} \tag{5.8.1}$$

and

$$\boldsymbol{R}_i^{\text{obs,new}} = \begin{pmatrix} \boldsymbol{R}_i^{\text{obs,new}} & C\left(\overline{\boldsymbol{X}}_i^{\text{obs}}, \bar{\boldsymbol{x}}_i^*\right) \\ C\left(\bar{\boldsymbol{x}}_i^*, \overline{\boldsymbol{X}}_i^{\text{obs}}\right) & 1 \end{pmatrix}.$$

Using the Schur complement formulae, one gets:

$$\left(\boldsymbol{R}_i^{\text{obs,new}}\right)^{-1} = \begin{pmatrix} \boldsymbol{Q}_{11} & \boldsymbol{Q}_{12} \\ \boldsymbol{Q}_{21} & \boldsymbol{Q}_{22} \end{pmatrix}, \tag{5.8.2}$$

where

$$\begin{aligned} \boldsymbol{Q}_{11} &= \left(\boldsymbol{R}_i^{\text{obs}}\right)^{-1} + \boldsymbol{Q}_{22}\left(\boldsymbol{R}_i^{\text{obs}}\right)^{-1} C\left(\overline{\boldsymbol{X}}_i^{\text{obs}}, \bar{\boldsymbol{x}}_i^*\right) C\left(\bar{\boldsymbol{x}}_i^*, \overline{\boldsymbol{X}}_i^{\text{obs}}\right)\left(\boldsymbol{R}_i^{\text{obs}}\right)^{-1} \\ &= \left(\boldsymbol{R}_i^{\text{obs}}\right)^{-1} + \boldsymbol{Q}_{22}\boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)\boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)^T, \\ \boldsymbol{Q}_{12} &= -\boldsymbol{Q}_{22}\left(\boldsymbol{R}_i^{\text{obs}}\right)^{-1} C\left(\overline{\boldsymbol{X}}_i^{\text{obs}}, \bar{\boldsymbol{x}}_i^*\right) = -\boldsymbol{Q}_{22}\boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right), \\ \boldsymbol{Q}_{21} &= -\boldsymbol{Q}_{22} C\left(\bar{\boldsymbol{x}}_i^*, \overline{\boldsymbol{X}}_i^{\text{obs}}\right)\left(\boldsymbol{R}_i^{\text{obs}}\right)^{-1} = -\boldsymbol{Q}_{22}\boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)^T, \end{aligned} \tag{5.8.3}$$

$\boldsymbol{Q}_{22} \in \mathbb{R}$, and:

$$\boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right) = \left(\boldsymbol{R}_i^{\text{obs}}\right)^{-1} C\left(\overline{\boldsymbol{X}}_i^{\text{obs}}, \bar{\boldsymbol{x}}_i^*\right) \tag{5.8.4}$$

From the previous equations, one has:

$$\begin{aligned} &\boldsymbol{H}_i^{\text{obs,new}}\left(\boldsymbol{R}_i^{\text{obs,new}}\right)^{-1}\left(\boldsymbol{H}_i^{\text{obs,new}}\right)^T \\ &= \boldsymbol{H}_i^{\text{obs}}\boldsymbol{Q}_{11}\left(\boldsymbol{H}_i^{\text{obs}}\right)^T + \boldsymbol{h}_i\left(\bar{\boldsymbol{x}}_i^*\right)\boldsymbol{Q}_{21}\left(\boldsymbol{H}_i^{\text{obs}}\right)^T + \boldsymbol{H}_i^{\text{obs}}\boldsymbol{Q}_{12}\boldsymbol{h}_i\left(\bar{\boldsymbol{x}}_i^*\right)^T + \boldsymbol{h}_i\left(\bar{\boldsymbol{x}}_i^*\right)\boldsymbol{Q}_{22}\boldsymbol{h}_i\left(\bar{\boldsymbol{x}}_i^*\right)^T \\ &= \boldsymbol{H}_i^{\text{obs}}\left(\boldsymbol{R}_i^{\text{obs}}\right)^{-1}\left(\boldsymbol{H}_i^{\text{obs}}\right)^T + \boldsymbol{Q}_{22}\Big(\boldsymbol{H}_i^{\text{obs}}\boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)\boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)^T\left(\boldsymbol{H}_i^{\text{obs}}\right)^T - \boldsymbol{h}_i\left(\bar{\boldsymbol{x}}_i^*\right)\boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)^T\left(\boldsymbol{H}_i^{\text{obs}}\right)^T \\ &\quad -\boldsymbol{H}_i^{\text{obs}}\boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)\boldsymbol{h}_i\left(\bar{\boldsymbol{x}}_i^*\right)^T + \boldsymbol{h}_i\left(\bar{\boldsymbol{x}}_i^*\right)\boldsymbol{h}_i\left(\bar{\boldsymbol{x}}_i^*\right)^T\Big) \\ &= \boldsymbol{H}_i^{\text{obs}}\left(\boldsymbol{R}_i^{\text{obs}}\right)^{-1}\left(\boldsymbol{H}_i^{\text{obs}}\right)^T + \boldsymbol{Q}_{22}\left(\boldsymbol{h}_i\left(\bar{\boldsymbol{x}}_i^*\right) - \boldsymbol{H}_i^{\text{obs}}\boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)\right)\left(\boldsymbol{h}_i\left(\bar{\boldsymbol{x}}_i^*\right) - \boldsymbol{H}_i^{\text{obs}}\boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)\right)^T \\ &= \boldsymbol{H}_i^{\text{obs}}\left(\boldsymbol{R}_i^{\text{obs}}\right)^{-1}\left(\boldsymbol{H}_i^{\text{obs}}\right)^T + \boldsymbol{Q}_{22}\boldsymbol{u}_i\left(\bar{\boldsymbol{x}}_i^*\right)\boldsymbol{u}_i\left(\bar{\boldsymbol{x}}_i^*\right)^T. \end{aligned} \tag{5.8.5}$$

In the same way, one has:

$$\begin{aligned}
&\boldsymbol{Y}_i^{\text{obs,new}}\left(\boldsymbol{R}_i^{\text{obs,new}}\right)^{-1}\left(\boldsymbol{H}_i^{\text{obs,new}}\right)^T \\
&= \boldsymbol{Y}_i^{\text{obs}}\boldsymbol{Q}_{11}\left(\boldsymbol{H}_i^{\text{obs}}\right)^T + \boldsymbol{\mu}_i^c\left(\bar{\boldsymbol{x}}_i^*\right)\boldsymbol{Q}_{21}\left(\boldsymbol{H}_i^{\text{obs}}\right)^T + \boldsymbol{Y}_i^{\text{obs}}\boldsymbol{Q}_{12}\boldsymbol{h}_i\left(\bar{\boldsymbol{x}}_i^*\right)^T + \boldsymbol{\mu}_i^c\left(\bar{\boldsymbol{x}}_i^*\right)\boldsymbol{Q}_{22}\boldsymbol{h}_i\left(\bar{\boldsymbol{x}}_i^*\right)^T \\
&= \boldsymbol{Y}_i^{\text{obs}}\left(\boldsymbol{R}_i^{\text{obs}}\right)^{-1}\left(\boldsymbol{H}_i^{\text{obs}}\right)^T + \boldsymbol{Q}_{22}\Big(\boldsymbol{Y}_i^{\text{obs}}\boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)\boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)^T\left(\boldsymbol{H}_i^{\text{obs}}\right)^T - \boldsymbol{\mu}_i^c\left(\bar{\boldsymbol{x}}_i^*\right)\boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)^T\left(\boldsymbol{H}_i^{\text{obs}}\right)^T \\
&\quad -\boldsymbol{Y}_i^{\text{obs}}\boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)\boldsymbol{h}_i\left(\bar{\boldsymbol{x}}_i^*\right)^T + \boldsymbol{\mu}_i^c\left(\bar{\boldsymbol{x}}_i^*\right)\boldsymbol{h}_i\left(\bar{\boldsymbol{x}}_i^*\right)^T\Big) \\
&= \boldsymbol{Y}_i^{\text{obs}}\left(\boldsymbol{R}_i^{\text{obs}}\right)^{-1}\left(\boldsymbol{H}_i^{\text{obs}}\right)^T + \boldsymbol{Q}_{22}\left(\boldsymbol{\mu}_i^c\left(\bar{\boldsymbol{x}}_i^*\right) - \boldsymbol{Y}_i^{\text{obs}}\boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)\right)\left(\boldsymbol{h}_i\left(\bar{\boldsymbol{x}}_i^*\right) - \left(\boldsymbol{H}_i^{\text{obs}}\right)\boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)\right)^T \\
&= \boldsymbol{Y}_i^{\text{obs}}\left(\boldsymbol{R}_i^{\text{obs}}\right)^{-1}\left(\boldsymbol{H}_i^{\text{obs}}\right)^T + \boldsymbol{Q}_{22}\left(\boldsymbol{\mu}_i^c\left(\bar{\boldsymbol{x}}_i^*\right) - \boldsymbol{Y}_i^{\text{obs}}\boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)\right)\boldsymbol{u}_i\left(\bar{\boldsymbol{x}}_i^*\right)^T.
\end{aligned}$$

From Eq. (5.3.18), it follows that:

$$\boldsymbol{\mu}_i^c\left(\bar{\boldsymbol{x}}_i^*\right) - \boldsymbol{Y}_i^{\text{obs}}\boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right) = \widehat{\boldsymbol{M}}_i\boldsymbol{u}_i\left(\bar{\boldsymbol{x}}_i^*\right).$$

where $\widehat{\boldsymbol{M}}_i = \mathbb{E}\left[\boldsymbol{M}_i|\boldsymbol{Y}_i^{\text{obs}}, C_i\right]$ is defined by Eq. (5.3.13).
Therefore, one has:

$$\boldsymbol{Y}_i^{\text{obs,new}}\left(\boldsymbol{R}_i^{\text{obs,new}}\right)^{-1}\left(\boldsymbol{H}_i^{\text{obs,new}}\right)^T = \boldsymbol{Y}_i^{\text{obs}}\left(\boldsymbol{R}_i^{\text{obs}}\right)^{-1}\left(\boldsymbol{H}_i^{\text{obs}}\right)^T + \boldsymbol{Q}_{22}\widehat{\boldsymbol{M}}_i\boldsymbol{u}_i\left(\bar{\boldsymbol{x}}_i^*\right)\boldsymbol{u}_i\left(\bar{\boldsymbol{x}}_i^*\right)^T.$$

From Eq. (5.8.5) it follows that:

$$\begin{aligned}
\boldsymbol{Y}_i^{\text{obs,new}}\left(\boldsymbol{R}_i^{\text{obs,new}}\right)^{-1}\left(\boldsymbol{H}_i^{\text{obs,new}}\right)^T &= \widehat{\boldsymbol{M}}_i\boldsymbol{H}_i^{\text{obs}}\left(\boldsymbol{R}_i^{\text{obs}}\right)^{-1}\left(\boldsymbol{H}_i^{\text{obs}}\right)^T + \boldsymbol{Q}_{22}\widehat{\boldsymbol{M}}_i\boldsymbol{u}_i\left(\bar{\boldsymbol{x}}_i^*\right)\boldsymbol{u}_i\left(\bar{\boldsymbol{x}}_i^*\right)^T. \\
&= \widehat{\boldsymbol{M}}_i\Big(\boldsymbol{H}_i^{\text{obs}}\left(\boldsymbol{R}_i^{\text{obs}}\right)^{-1}\left(\boldsymbol{H}_i^{\text{obs}}\right)^T + \boldsymbol{Q}_{22}\boldsymbol{u}_i\left(\bar{\boldsymbol{x}}_i^*\right)\boldsymbol{u}_i\left(\bar{\boldsymbol{x}}_i^*\right)^T.\Big) \\
&= \widehat{\boldsymbol{M}}_i\boldsymbol{H}_i^{\text{obs,new}}\left(\boldsymbol{R}_i^{\text{obs,new}}\right)^{-1}\left(\boldsymbol{H}_i^{\text{obs,new}}\right)^T.
\end{aligned}$$

Therefore, one has:

$$\boldsymbol{Y}_i^{\text{obs,new}}\left(\boldsymbol{R}_i^{\text{obs,new}}\right)^{-1}\left(\boldsymbol{H}_i^{\text{obs,new}}\right)^T\left(\boldsymbol{H}_i^{\text{obs,new}}\left(\boldsymbol{R}_i^{\text{obs,new}}\right)^{-1}\left(\boldsymbol{H}_i^{\text{obs,new}}\right)^T\right)^{-1} = \widehat{\boldsymbol{M}}_i.$$

Thus, one can conclude from the previous equation and Eq. (5.3.13):

$$\mathbb{E}\left[\boldsymbol{M}_i|\{\boldsymbol{Y}_i^{\text{obs}}, \boldsymbol{Y}_i\left(\bar{\boldsymbol{x}}_i^*\right) = \boldsymbol{\mu}_i^c\left(\bar{\boldsymbol{x}}_i^*\right)\}, C_i\right] = \mathbb{E}\left[\boldsymbol{M}_i|\boldsymbol{Y}_i^{\text{obs}}, C_i\right].$$

Second equation

From Eq. (5.3.16) it follows that:

$$\boldsymbol{R}_{t_i}\left(\boldsymbol{Y}_i^{\text{obs}}, \boldsymbol{\mu}_i^c\left(\bar{\boldsymbol{x}}_i^*\right)\right) = \frac{1}{n_i+1}\left(\boldsymbol{Y}_i^{\text{obs,new}} - \widehat{\boldsymbol{M}}_i\boldsymbol{H}_i^{\text{obs,new}}\right)\left(\boldsymbol{R}_i^{\text{obs,new}}\right)^{-1}\left(\boldsymbol{Y}_i^{\text{obs,new}} - \widehat{\boldsymbol{M}}_i\boldsymbol{H}_i^{\text{obs,new}}\right)^T. \tag{5.8.6}$$

From Eq. (5.8.1), one has:

$$\boldsymbol{Y}_i^{\text{obs,new}} - \widehat{\boldsymbol{M}}_i\boldsymbol{H}_i^{\text{obs,new}} = \left(\boldsymbol{Y}_i^{\text{obs}} - \widehat{\boldsymbol{M}}_i\boldsymbol{H}_i^{\text{obs}}\ ,\ \boldsymbol{\mu}_i^c\left(\bar{\boldsymbol{x}}_i^*\right) - \widehat{\boldsymbol{M}}_i\boldsymbol{h}_i\left(\bar{\boldsymbol{x}}_i^*\right)\right). \tag{5.8.7}$$

From Eq. (5.3.18), one gets:

$$\boldsymbol{\mu}_i^c\left(\bar{\boldsymbol{x}}_i^*\right) - \widehat{\boldsymbol{M}}_i\boldsymbol{h}_i\left(\bar{\boldsymbol{x}}_i^*\right) = \left(\boldsymbol{Y}_i^{\text{obs}} - \widehat{\boldsymbol{M}}_i\boldsymbol{H}_i^{\text{obs}}\right)\boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right),$$

where $\boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)$ is defined by Eq. (5.8.4).
Therefore, Eq. (5.8.7) becomes:

$$\boldsymbol{Y}_i^{\text{obs,new}} - \widehat{\boldsymbol{M}}_i \boldsymbol{H}_i^{\text{obs,new}} = \left(\boldsymbol{Y}_i^{\text{obs}} - \widehat{\boldsymbol{M}}_i \boldsymbol{H}_i^{\text{obs}}\right)\left(\boldsymbol{I}_{n_i}\ ,\ \boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)\right).$$

Eq. (5.8.6) can thus be rewritten:

$$\begin{aligned}\boldsymbol{R}_{t_i}\left(\boldsymbol{Y}_i^{\text{obs}}, \boldsymbol{\mu}_i^c\left(\bar{\boldsymbol{x}}_i^*\right)\right) = \frac{1}{n_i+1} \quad & \left(\boldsymbol{Y}_i^{\text{obs}} - \widehat{\boldsymbol{M}}_i \boldsymbol{H}_i^{\text{obs}}\right)\left(\boldsymbol{I}_{n_i}\ ,\ \boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)\right)\left(\boldsymbol{R}_i^{\text{obs,new}}\right)^{-1} \\ & \left(\boldsymbol{I}_{n_i}\ ,\ \boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)\right)^T \left(\boldsymbol{Y}_i^{\text{obs}} - \widehat{\boldsymbol{M}}_i \boldsymbol{H}_i^{\text{obs}}\right)^T.\end{aligned} \tag{5.8.8}$$

Besides, one has:

$$\begin{aligned}\left(\boldsymbol{I}_{n_i}\ ,\ \boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)\right)\left(\boldsymbol{R}_i^{\text{obs,new}}\right)^{-1} &= \left(\boldsymbol{I}_{n_i}\ ,\ \boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)\right)\begin{pmatrix}\boldsymbol{Q}_{11} & \boldsymbol{Q}_{12}\ , \\ \boldsymbol{Q}_{21} & \boldsymbol{Q}_{22}\ ,\end{pmatrix} \\ &= \left(\boldsymbol{Q}_{11} + \boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)\boldsymbol{Q}_{21}\ ,\ \boldsymbol{Q}_{12} + \boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)\boldsymbol{Q}_{22}\right) \\ &= \left(\left(\boldsymbol{R}_i^{\text{obs}}\right)^{-1}\ ,\ \boldsymbol{0}_{n_i}\right)\end{aligned}$$

One can infer that:

$$\left(\boldsymbol{I}_{n_i}\ ,\ \boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)\right)\begin{pmatrix}\boldsymbol{Q}_{11} & \boldsymbol{Q}_{12}\ , \\ \boldsymbol{Q}_{21} & \boldsymbol{Q}_{22}\ ,\end{pmatrix}\begin{pmatrix}\boldsymbol{I}_{n_i} \\ \left(\boldsymbol{v}_i\left(\bar{\boldsymbol{x}}_i^*\right)\right)^T\end{pmatrix} = \left(\boldsymbol{R}_i^{\text{obs}}\right)^{-1}$$

One can conclude from Eq. (5.8.8) that:

$$\boldsymbol{R}_{t_i}\left(\boldsymbol{Y}_i^{\text{obs}}, \boldsymbol{\mu}_i^c\left(\bar{\boldsymbol{x}}_i^*\right)\right) = \frac{n_i}{n_i+1}\boldsymbol{R}_{t_i}\left(\boldsymbol{Y}_i^{\text{obs}}\right).$$

5.8.3 Proof of Proposition 5.3.1

From Eq. (5.3.20) it follows that the criterion can be rewritten:

$$\bar{\boldsymbol{x}}_i^{new} = \underset{\bar{\boldsymbol{x}}_i^* \in \overline{\mathbb{X}}_i}{\operatorname{argmin}} \int_{\overline{\mathbb{X}}_i} \operatorname{Tr}\left(\boldsymbol{R}_{t_i}\left(\boldsymbol{Y}_i^{\text{obs}}, \boldsymbol{y}_i\left(\bar{\boldsymbol{x}}_i^*\right)\right)\right) v_i\left(\bar{\boldsymbol{x}}_i | \overline{\boldsymbol{X}}_i^{\text{obs}}, \bar{\boldsymbol{x}}_i^*\right) d\mu_{\overline{\mathbb{X}}_i}\left(\bar{\boldsymbol{x}}_i\right). \tag{5.8.9}$$

Moreover, based on Lemma 1:

$$\bar{\boldsymbol{x}}_i^{new} = \underset{\bar{\boldsymbol{x}}_i^* \in \overline{\mathbb{X}}_i}{\operatorname{argmin}} \int_{\overline{\mathbb{X}}_i} \frac{n_i}{n_i+1}\operatorname{Tr}\left(\boldsymbol{R}_{t_i}\left(\boldsymbol{Y}_i^{\text{obs}}\right)\right) v_i\left(\bar{\boldsymbol{x}}_i | \overline{\boldsymbol{X}}_i^{\text{obs}}, \bar{\boldsymbol{x}}_i^*\right) d\mu_{\overline{\mathbb{X}}_i}\left(\bar{\boldsymbol{x}}_i\right). \tag{5.8.10}$$

By noting that $\operatorname{Tr}\left(\boldsymbol{R}_{t_i}\left(\boldsymbol{Y}_i^{\text{obs}}\right)\right)$ does not depends on $\bar{\boldsymbol{x}}_i$ and $\bar{\boldsymbol{x}}_i^*$, the criterion can finally be written:

$$\bar{\boldsymbol{x}}_i^{new} = \underset{\bar{\boldsymbol{x}}_i^* \in \overline{\mathbb{X}}_i}{\operatorname{argmin}} \int_{\overline{\mathbb{X}}_i} v_i\left(\bar{\boldsymbol{x}}_i | \overline{\boldsymbol{X}}_i^{\text{obs}}, \bar{\boldsymbol{x}}_i^*\right) d\mu_{\overline{\mathbb{X}}_i}\left(\bar{\boldsymbol{x}}_i\right). \tag{5.8.11}$$

5.8.4 Proof of Proposition 5.4.1

By definition (see Eq. (5.4.1)) and given the independence of $\boldsymbol{Y}_2$ and $\boldsymbol{Y}_3$, one has:

$$\boldsymbol{Y}^c_{\text{nest}}(\boldsymbol{x}_1,\boldsymbol{x}_2) \overset{d}{=} \boldsymbol{\mu}^c_2\left(\boldsymbol{\mu}^c_3(\boldsymbol{x}_1)+\boldsymbol{\xi}_3\boldsymbol{\sigma}^c_{\boldsymbol{x}3}(\boldsymbol{x}_1),\boldsymbol{x}_2\right)+\boldsymbol{\xi}_2\boldsymbol{\sigma}^c_{\boldsymbol{x}2}\left(\boldsymbol{\mu}^c_3(\boldsymbol{x}_1)+\boldsymbol{\xi}_3\boldsymbol{\sigma}^c_{\boldsymbol{x}3}(\boldsymbol{x}_1),\boldsymbol{x}_2\right),$$

where $\boldsymbol{\xi}_2$ and $\boldsymbol{\xi}_3$ are independent Gaussian variables, such that $\boldsymbol{\xi}_2 \sim \mathcal{N}\left(\boldsymbol{0},\widehat{\boldsymbol{R}}_{t_2}\right)$ and $\boldsymbol{\xi}_3 \sim \mathcal{N}\left(\boldsymbol{0},\widehat{\boldsymbol{R}}_{t_3}\right)$.

This implies:

$$\mathbb{E}\left[\boldsymbol{Y}^c_{\text{nest}}(\boldsymbol{x}_1,\boldsymbol{x}_2)\right]=\mathbb{E}\left[\boldsymbol{\mu}^c_2\left(\boldsymbol{\mu}^c_3(\boldsymbol{x}_1)+\boldsymbol{\xi}_3\boldsymbol{\sigma}^c_{\boldsymbol{x}3}(\boldsymbol{x}_1),\boldsymbol{x}_2\right)\right],$$

because $\boldsymbol{\xi}_2$ and $\boldsymbol{\xi}_3$ are independent and $\mathbb{E}[\boldsymbol{\xi}_2]=\boldsymbol{0}$.

Besides, one has:

$$\begin{aligned}\boldsymbol{Y}^c_{\text{nest}}(\boldsymbol{x}_1,\boldsymbol{x}_2)\,\boldsymbol{Y}^c_{\text{nest}}(\boldsymbol{x}_1,\boldsymbol{x}_2)^T = \;& \boldsymbol{\mu}^c_2\left(\boldsymbol{\mu}^c_3(\boldsymbol{x}_1)+\boldsymbol{\xi}_3\boldsymbol{\sigma}^c_{\boldsymbol{x}3}(\boldsymbol{x}_1),\boldsymbol{x}_2\right)\boldsymbol{\mu}^c_2\left(\boldsymbol{\mu}^c_3(\boldsymbol{x}_1)+\boldsymbol{\xi}_3\boldsymbol{\sigma}^c_{\boldsymbol{x}3}(\boldsymbol{x}_1),\boldsymbol{x}_2\right)^T\\ &+\left(\boldsymbol{\sigma}^c_2\left(\boldsymbol{\mu}^c_3(\boldsymbol{x}_1)+\boldsymbol{\xi}_3\boldsymbol{\sigma}^c_{\boldsymbol{x}3}(\boldsymbol{x}_1),\boldsymbol{x}_2\right)\right)^2\boldsymbol{\xi}_2\boldsymbol{\xi}_2^T\\ &+\boldsymbol{\sigma}^c_2\left(\boldsymbol{\mu}^c_3(\boldsymbol{x}_1)+\boldsymbol{\xi}_3\boldsymbol{\sigma}^c_{\boldsymbol{x}3}(\boldsymbol{x}_1),\boldsymbol{x}_2\right)\boldsymbol{\xi}_2\boldsymbol{\mu}^c_2\left(\boldsymbol{\mu}^c_3(\boldsymbol{x}_1)+\boldsymbol{\xi}_3\boldsymbol{\sigma}^c_{\boldsymbol{x}3}(\boldsymbol{x}_1),\boldsymbol{x}_2\right)^T\\ &+\boldsymbol{\sigma}^c_2\left(\boldsymbol{\mu}^c_3(\boldsymbol{x}_1)+\boldsymbol{\xi}_3\boldsymbol{\sigma}^c_{\boldsymbol{x}3}(\boldsymbol{x}_1),\boldsymbol{x}_2\right)\boldsymbol{\mu}^c_2\left(\boldsymbol{\mu}^c_3(\boldsymbol{x}_1)+\boldsymbol{\xi}_3\boldsymbol{\sigma}^c_{\boldsymbol{x}3}(\boldsymbol{x}_1),\boldsymbol{x}_2\right)\boldsymbol{\xi}_2^T.\end{aligned}$$

Given that $\boldsymbol{\xi}_2$ and $\boldsymbol{\xi}_3$ are independent, $\mathbb{E}[\boldsymbol{\xi}_2]=\boldsymbol{0}$ and $\mathbb{E}\left[\boldsymbol{\xi}_2\boldsymbol{\xi}_2^T\right]=\widehat{\boldsymbol{R}}_{t_2}$, one has:

$$\begin{aligned}\mathbb{E}\left[\boldsymbol{Y}^c_{\text{nest}}(\boldsymbol{x}_1,\boldsymbol{x}_2)\,\boldsymbol{Y}^c_{\text{nest}}(\boldsymbol{x}_1,\boldsymbol{x}_2)^T\right] = \;& \mathbb{E}\left[\left(\boldsymbol{\sigma}^c_2\left(\boldsymbol{\mu}^c_3(\boldsymbol{x}_1)+\boldsymbol{\xi}_3\boldsymbol{\sigma}^c_{\boldsymbol{x}3}(\boldsymbol{x}_1),\boldsymbol{x}_2\right)\right)^2\right]\widehat{\boldsymbol{R}}_{t_2}\\ &+\mathbb{E}\left[\boldsymbol{\mu}^c_2\left(\boldsymbol{\mu}^c_3(\boldsymbol{x}_1)+\boldsymbol{\xi}_3\boldsymbol{\sigma}^c_{\boldsymbol{x}3}(\boldsymbol{x}_1),\boldsymbol{x}_2\right)\boldsymbol{\mu}^c_2\left(\boldsymbol{\mu}^c_3(\boldsymbol{x}_1)+\boldsymbol{\xi}_3\boldsymbol{\sigma}^c_{\boldsymbol{x}3}(\boldsymbol{x}_1),\boldsymbol{x}_2\right)^T\right].\end{aligned}$$

5.8.5 Proof of Proposition 5.4.2

By definition (see Eq. (5.4.1)), one has:

$$\boldsymbol{Y}^c_{\text{nest}}(\boldsymbol{x}_1,\boldsymbol{x}_2) \overset{d}{=} \boldsymbol{\mu}^c_2\left(\boldsymbol{\mu}^c_3(\boldsymbol{x}_1)+\boldsymbol{\xi}_3\boldsymbol{\sigma}^c_{\boldsymbol{x}3}(\boldsymbol{x}_1),\boldsymbol{x}_2\right)+\boldsymbol{\xi}_2\boldsymbol{\sigma}^c_{\boldsymbol{x}2}\left(\boldsymbol{\mu}^c_3(\boldsymbol{x}_1)+\boldsymbol{\xi}_3\boldsymbol{\sigma}^c_{\boldsymbol{x}3}(\boldsymbol{x}_1),\boldsymbol{x}_2\right),$$

where $\boldsymbol{\xi}_2 \sim \mathcal{N}\left(\boldsymbol{0},\widehat{\boldsymbol{R}}_{t_2}\right)$ and $\boldsymbol{\xi}_3 \sim \mathcal{N}\left(\boldsymbol{0},\widehat{\boldsymbol{R}}_{t_3}\right)$.
If $\sigma^c_{\boldsymbol{x}3}(\boldsymbol{x}_1)$ is small enough, then the previous equation can be linearized with respect to $\sigma^c_{\boldsymbol{x}3}(\boldsymbol{x}_1)$. Thus, one has:

$$\boldsymbol{Y}^c_{\text{nest}}(\boldsymbol{x}_1,\boldsymbol{x}_2) \overset{d}{\approx} \boldsymbol{\mu}^c_2\left(\boldsymbol{\mu}^c_3(\boldsymbol{x}_1),\boldsymbol{x}_2\right)+\frac{\partial\boldsymbol{\mu}^c_2}{\partial\boldsymbol{\rho}}\left(\boldsymbol{\mu}^c_3(\boldsymbol{x}_1),\boldsymbol{x}_2\right)\boldsymbol{\sigma}^c_{\boldsymbol{x}3}(\boldsymbol{x}_1)\,\boldsymbol{\xi}_3+\boldsymbol{\xi}_2\boldsymbol{\sigma}^c_{\boldsymbol{x}2}\left(\boldsymbol{\mu}^c_3(\boldsymbol{x}_1),\boldsymbol{x}_2\right).$$

Thanks to the fact that $\boldsymbol{\xi}_2$ and $\boldsymbol{\xi}_3$ are independent Gaussian processes, a Gaussian predictor of the nested code can be obtained from the previous equation. Furthermore, this Gaussian process has the following mean function:

$$\boldsymbol{\mu}^c_{\text{nest}}(\boldsymbol{x}_1,\boldsymbol{x}_2)=\boldsymbol{\mu}^c_2\left(\boldsymbol{\mu}^c_3(\boldsymbol{x}_1),\boldsymbol{x}_2\right), \tag{5.8.12}$$

and its covariance function is:

$$\begin{aligned}\boldsymbol{C}^c_{\text{nest}}\left((\boldsymbol{x}_1,\boldsymbol{x}_2),\left(\boldsymbol{x}_1',\boldsymbol{x}_2'\right)\right) = \;& \widehat{\boldsymbol{R}}_{t_2}C^c_2\left(\left(\boldsymbol{\mu}^c_3(\boldsymbol{x}_1),\boldsymbol{x}_2\right),\left(\boldsymbol{\mu}^c_3\left(\boldsymbol{x}_1'\right),\boldsymbol{x}_2'\right)\right)\\ &+\frac{\partial\boldsymbol{\mu}^c_2}{\partial\boldsymbol{\rho}}\left(\boldsymbol{\mu}^c_3(\boldsymbol{x}_1),\boldsymbol{x}_2\right)\widehat{\boldsymbol{R}}_{t_3}\left(\frac{\partial\boldsymbol{\mu}^c_2}{\partial\boldsymbol{\rho}}\left(\boldsymbol{\mu}^c_3\left(\boldsymbol{x}_1'\right),\boldsymbol{x}_2'\right)\right)^T C^c_3\left(\boldsymbol{x}_1,\boldsymbol{x}_1'\right).\end{aligned}$$

Conclusions

This work was motivated by an application case. This application case is the coupling of two computationally expensive codes. The first code is a detonation code and the second code is a structural dynamics code. The two codes have functional (i.e. high dimensional vectorial) outputs. One of the inputs of the second code is the functional output of the first code. The objective was to perform design and certification studies on this system.
The methods used for the design and the certification, like sensitivity analysis, risk analysis or optimization, require a large number of evaluations of the output of the considered system. Considering the high computational cost of the codes, it is in practice impossible to apply these methods directly to the real codes. In this work, we were particularly interested in performing a sensitivity analysis of the output of the nested code with respect to its inputs. The objective of this work was therefore to construct a predictor of the output of the system from a small set of observations, which is accurate on the most likely regions of the input domain of the nested code.
Several difficulties were raised by the surrogate modeling of the considered system:

- There are two codes.
- The two codes are costly, and therefore there will be a few observations available.
- The second code has functional input and output.

This thesis made contributions to help achieve the initially set objective.
The framework of Gaussian process regression, more precisely Universal Kriging in a Bayesian framework, was used.
In a first step, the case of two nested codes with scalar outputs and no intermediary observations was considered. An original parametrization of the mean function of the Gaussian process modeling the nested code was proposed. This parametrization consists of the coupling of two polynomials. It yields a better prediction accuracy than a classical Universal Kriging predictor with a polynomial mean function.
In a second step, the case of two nested codes with scalar outputs and observations of the intermediary variable was considered. A stochastic predictor of the nested code output based on the coupling of Gaussian predictors of the two codes was proposed. The predictor can be constructed from all the types of observations available: those of the nested code, those of the first code and those of the second code. The predictor is non-Gaussian and its mean and variance have to be evaluated with Monte Carlo methods. Furthermore, we proposed two sequential design criteria which aim at improving the accuracy of the predictor on the whole input domain. One of the criteria can take account of the difference between the computational costs of the two codes.
The two sequential design criteria requiring a large number of evaluations of the prediction variance, two adaptations of the predictor were proposed for accelerating the computation of the prediction variance. The first adaptation can lead to closed forms of the mean and the variance of the predictor, if the output is assumed to be infinitely differentiable. The second

one was obtained by proposing a linearization of the coupling of the predictors of the two codes. The predictor of the nested code is then Gaussian with mean and variance functions in closed forms.
In a third and final step, the case of two nested codes with functional outputs and observations of the intermediary variable was considered. An original dimension reduction of the functional input of the second code was proposed. It is based on the approximation of the output of the second code by a linear causal filter and on the projection of the functional input on a basis which is adapted to the linear approximation.
Thanks to this dimension reduction an efficient predictor of the second code is obtained.
Then, similarly to the case of scalar outputs, we proposed a Gaussian predictor of the nested code based on the linearization of the coupling the Gaussian predictors associated with the two codes. Finally, the previously defined sequential design criteria were adapted to the case of codes with functional outputs.

In this thesis, we focused on the surrogate modeling of two nested codes. The study of the surrogate modeling for the coupling of more than two codes or of more complex networks of computer codes is a promising topic. In a non-ringed network, several other relationships between the codes can be found. There can be chains of more than two codes. The output of two different codes can be the inputs of a third code. Besides the case of ringed network could also be studied. Finally, the case of two nested codes with a functional output for the first code and a scalar output for the second code could be studied.
From a practical point of view, the use of parallel computing for the computation of the sequential design criteria with Monte Carlo methods could be useful, especially when the dimension of the input domain is high and the number of Monte Carlo draws too. This could be applied to the case of a one-by-one sequential enrichment of the design. For the case of a batch enrichment, with the addition of $k > 1$ new observations at each step, the number of possible combinations can be very high, which can lead to a high computational burden. The number of possible combinations increases significantly when the number of candidates and k increase. Moreover, the number of candidates is generally higher when the dimension of the inputs is high.
Besides, if we note that the inversion of the covariance matrix of the observations can be expensive when the number of observations is high, it could be interesting to study the possible combination between the proposed linearized predictor and the nested Kriging approach of Rullière et al. [2018] in order to extend the results obtained to the case of a high number of observations.
The study of optimization strategies for nested codes could also be of great interest. Note that the Expected Improvement criterion presented in Section 1.5 is adapted to the case of a computer code with a scalar output (the quantity to optimize) and its adaptation to the case of a functional output is not direct. If the scalar criterion to be optimized is obtained by a linear transformation of the output, then, thanks to the Gaussianity of the proposed predictor, an enrichment based on the Expected Improvement could be performed.

Bibliography

P. Abrahamsen. A review of Gaussian random fields and correlation functions. Technical report, Norwegian computing center,, 1997.

M. Abramowitz and I. Stegun. *Handbook of mathematical functions*. Dover, New York, 1965.

M. Arnst, R. Ghanem, and C. Soize. Identification of Bayesian posteriors for coefficients of chaos expansions. *Journal of Computational Physics*, 229 (9):3134–3154, 2010.

F. Bachoc. Cross validation and maximum likelihood estimation of hyper-parameters of Gaussian processes with model misspecification. *Computational Statistics and Data Analysis*, 66:55––69, 2013a.

F. Bachoc. *Parametric estimation of covariance function in Gaussian-process based Kriging models. Application to uncertainty quantification for computer experiments*. PhD thesis, Université Paris-Diderot - Paris VII, 2013b.

C. T. H. Baker. *The numerical treatment of integral equations*. Clarendon Press, Oxford, 1977.

R. A. Bates, R. J. Buck, E. Riccomagno, and H. P. Wynn. Experimental design and observation for large systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):77–94, 1996.

M. J. Bayarri, J. O. Berger, J. Cafeo, G. Garcia-Donato, F. Liu, J. Palomo, R. J. Parthasarathy, R. Paulo, J. Sacks, and D. Walsh. Computer model validation with functional output. *The Annals of Statistics*, 35(5):1874–1906, 2007.

J. Bect, D. Ginsbourger, L. Li, V. Picheny, and E. Vasquez. Sequential design of computer experiments for the estimation of a probability of failure. *Statistics and Computing*, 22: 773–797, 2012.

J. O. Berger, V. De Oliveira, and B. Sansó. Objective Bayesian analysis of spatially correlated data. *Journal of the American Statistical Association*, 96(456):1361–1374, 2001.

B.J. Bichon, M.S. Eldred, L.P. Swiler, S. Mahadevan, and J.M. McFarland. Efficient global reliability analysis for non linear implicit performance functions. *AIAA Journal*, 46(10), 2008.

I. Bilionis, N. Zabaras, B.A. Konomi, and G. Lin. Multi-output separable Gaussian process: towards an efficient, fully Bayesian paradigm for uncertainty quantification. *Journal of Computational Physics*, 241, 2013.

G. Blatman and B. Sudret. An adaptive algorithm to build up sparse polynomial chaos expansions for stochastic finite element analysis. *Probabilistic Engineering Mechanics*, 25 (2):183–197, 2010.

G. Blatman and B. Sudret. Adaptive sparse polynomial chaos expansion based on least angle regression. *Journal of Computational Physics*, 230(6):2345–2367, 2011.

R. Brent. *Algorithms for Minimization without Derivatives*. Englewood Cliffs N.J.: Prentice-Hall, 1973.

E. Candes and T. Tao. The dantzig selector: Statistical estimation when p is much larger than n. *The Annals of Statistics*, 35(6):2313–2351, 12 2007.

C. Chevalier, J. Bect, D. Ginsbourger, and E. Vazquez. Fast parallel Kriging-based stepwise uncertainty reduction with application to the identification of an excursion set. *Technometrics*, 56(4):455–465, 2014. doi: 10.1080/00401706.2013.860918>.

P.G. Constantine, E. Dow, and Q-Q. Wang. Active Subspace methods in theory and practice: Applications to Kriging surfaces. *SIAM Journal on Scientific Computing*, 36(4):1500–1524, 2014.

S. Conti and A. O'Hagan. Bayesian emulation of complex multi-output and dynamic computer models. *Journal of Statistical Planning and Inference*, 140:640–651, 2010.

S. Conti, J.P. Gosling, J.E. Oakley, and A. O'Hagan. Gaussian process emulation of dynamic computer codes. *Biometrika*, 96(3):663–676, 2009.

R.D. Cook and C.J. Nachtsheim. A comparison of algorithms for constructing exact D-optimal designs. *Technornetrics*, 22:315–324, 1980.

A. Damianou and N. D. Lawrence. Deep Gaussian processes. In C. Carvalho and P. Ravikumar, editors, *Proceedings of the Sixteenth International Workshop on Artificial Intelligence and Statistics (AISTATS)*, AISTATS '13, pages 207–215. JMLR W&CP 31, 2013.

S. Das, R. Ghanem, and S. Finette. Polynomial chaos representation of spatio-temporal random field from experimental measurements. *J. Comput. Phys.*, 228:8726–8751, 2009.

G. Defaux and P. Evrard. Probabilistic analysis of a containment vessel subjected to dynamic pressure loading using surrogate models. In *Safety, Reliability, Risk and Life-Cycle Performance of Structures and Infrastructures*, pages 3203–3210. CRC Press, January 2014. doi: 10.1201/b16387-463.

O. Dubrule. Cross validation of Kriging in a unique neighborhood. *Mathematical Geology*, 15 (6):687–699, 1983.

B. Echard, N. Gayton, and M. Lemaire. AK-MCS: An active learning reliability method combining Kriging and Monte Carlo simulation. *Structural Safety*, 33:145–154, 2011.

G. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1:211–218, 1936.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Ann. Stat.*, 32: 407–499, 2004.

M. Efroymson. *Mathematical models for digital computers*, volume 1, chapter Multiple regression analysis, pages 191—-203. Wiley, 1960.

G. Elfving. Optimum allocation in linear regression theory. *The Annals of Mathematical Statistics*, 23(2):255–262, 1952.

K.T. Fang and D.K. Lin. Uniform experimental designs and their applications in industry. *Handbook of Statistics*, 22:131–178, 2003.

K.T. Fang, R. Li, and A. Sudjianto. *Design and modeling for computer experiments*. Chapman & Hall, Computer Science and Data Analysis Series, London, 2006.

V.V. Fedorov. *Theory Of Optimal Experiments*. Academic Press, New York, 1972.

V.V. Fedorov and P. Hackl. *Model-Oriented Design of Experiments*. Springer-Verlag, New York, 1997.

R. Fisher. *Statistical Methods for Research Workers*. Oliver & Boyd, 1925.

T.E. Fricker, J.E. Oakley, and N.M. Urban. Multivariate Gaussian process emulators with nonseparable covariance structures. *Technometrics*, 55(1):47–56, 2013.

G.M. Furnival and R.W. Wilson. Regressions by leaps and bounds. *Technometrics*, 16(4): 499–511, 1974.

D. Geman and B. Jedynak. An active testing model for tracking roads in satellite images. 18: 1 – 14, 02 1996.

R. Ghanem and P. D. Spanos. Polynomial chaos in stochastic finite elements. *Journal of Applied Mechanics*, 57(1):197–202, 1990.

R. Ghanem and P. D. Spanos. *Stochastic Finite Elements: A Spectral Approach, rev. ed.* Dover Publications, New York, 2003.

D.M. Ghiocel and R.G. Ghanem. Stochastic finite-element analysis of seismic soil-structure interaction. *Journal of Engineering Mechanics*, 128(1):66–77, 2002.

D. Ginsbourger, R. Le Riche, and L. Carraro. *Computational intelligence in expensive optimization problems*, volume 2 of *Adaptation Learning and Optimization*, chapter Kriging is well-suited to parallelize optimization, pages 131–162. Springer Berlin Heidelberg, 2010.

R. Gramacy and H. Lian. Gaussian process single-index models as emulators for computer experiments. *Technometrics*, 54:1:30–41, 2012.

R. B. Gramacy and H. K. H. Lee. Cases for the nugget in modeling computer experiments. *Statistics and Computing*, 22:713–722, 2012.

J. H. Halton. Algorithm 247: Radical-inverse quasi-random point sequence. *Commun. ACM*, 7(12):701–702, 1964.

J. Hammersley. *Monte Carlo Methods*. Springer Netherlands, 1964.

M.S. Handcock and M.L. Stein. A Bayesian analysis of Kriging. *Technometrics*, 35:403–4010, 1993.

T. Hastie and R. Tibshirani. *Generalized additive models*. Chapman & Hall, 1990.

T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: Data mining, inference and prediction*. Springer, New York, 2001.

T. Hastie, R. Tibshirani, and Friedman. *Elements of Statistical Learning*. Springer, New York, 2002.

T. Hastie, J. Taylor, R. Tibshirani, and G. Walther. Forward stagewise regression and the monotone lasso. *Electronic Journal of Statistics*, 1:1–29, 2007.

C. Helbert, D. Dupuy, and L. Carraro. Assessment of uncertainty in computer experiments, from Universal to Bayesian Kriging. *Applied Stochastic Models in Business and Industry*, 25:99–113, 2009.

T. Hesterberg, N. H. Choi, L. Meier, and C. Fraley. Least angle and ℓ_1 penalized regression: A review. *Statistics Surveys*, 2:61–93, 2008.

D. Higdon, J. Gattiker, B. Williams, and M. Rightley. Computer model calibration using high-dimensional output. *Journal of the American Statistical Association*, 103(482):570–583, 2008.

W. Hoeffding. A class of statistics with asymptotically normal distributions. *The Annals of Mathematical Statistics*, 19:293–325, 1948.

A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

T. Homma and A. Saltelli. Importance measures in global sensitivity analysis of nonlinear models. *Reliability Engineering & System Safety*, 52(1):1–17, 1996.

A. Höskuldsson. PLS regression methods. *Journal of chemometrics*, 2(3):211–228, 1988.

R. Hu and M. Ludkovski. Sequential design for ranking response surfaces. *SIAM/ASA Journal on Uncertainty Quantification*, 5:212–239, 2017.

Y. Hung, V.R. Joseph, and S.N. Melkote. Analysis of computer experiments with functional response. *Technometrics*, 5(1):35–44, 2015.

J.E. Jackson. *A User's Guide to Principal Components*. Wiley, Hoboken, New Jersey, 2003.

J. D. Jakeman, M. S. Eldred, and K. Sargsyan. Enhancing ℓ_1-minimization estimates of polynomial chaos expansions using basis selection. *Journal of Computational Physics*, 289: 18 – 34, 2015.

G. James, P. Radchenko, and J. Lv. DASSO: Connections between the dantzig selector and lasso. *J. Royal Stat. Soc., Series B*, 71(1):127–142, 2008.

M. E. Johnson, L. M. Moore, and D. Ylvisaker. Minimax and maximin distance designs. *Journal of Statistical Planning and Inference*, 26(2):131–148, 1990.

D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Biometrika*, 13:455–492, 1998.

V. R. Joseph, Y. Hung, and A. Sudjianto. Blind Kriging: A new method for developing metamodels. *Journal of mechanical design*, 130(3):031102, 2008.

M. C. Kennedy and A. O'Hagan. Predicting the output from a complex computer code when fast approximations are avalaible. *Biometrika*, 87:1–13, 2000.

M. C. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 63(3):425–464, 2001.

P. Kersaudy, B. Sudret, N. Varsier, and O. Picon. A new surrogate modeling technique combining Kriging and polynomial chaos expansions - application to uncertainty analysis in computational dosimetry. *Journal of Computational Physics*, 286:103–117, 2015.

J. Kiefer. Optimum designs in regression problems, ii. *The Annals of Mathematical Statistics*, 32(1):298–325, 1961.

J. Kiefer and J. Wolfowitz. Optimum designs in regression problems. *The Annals of Mathematical Statistics*, 30(2):271–294, 1959.

J.P.C. Kleijnen. Regression and Kriging metamodels with their experimental designs in simulation: A review. *European Journal of Operational Research*, 256:1–16, 2017.

K. Konakli and B. Sudret. Polynomial meta-models with canonical low-rank approximations: Numerical insights and comparison to sparse polynomial chaos expansions. *Journal of Computational Physics*, 321:1144 – 1169, 2016.

H.J. Kushner. A new method of locating the maximal point of an arbitrary multipeak curve in the presence of noise. *J. Basic Eng.*, 86:97–106, 1964.

L Le Gratiet. *Multi-fidelity Gaussian process regression for computer experiments.* PhD thesis, Université Paris-Diderot - Paris VII, 2013.

L. Le Gratiet and J. Garnier. Recursive co-Kriging model for design of computer experiments with multiple levels of fidelity. *International Journal for Uncertainty Quantification*, 4(5): 365–386, 2014.

O.P. Le Maître and O.M. Knio. *Spectral Methods for Uncertainty Quantification.* Springer, 2010.

R. Lebrun and A. Dutfoy. Do Rosenblatt and Nataf isoprobabilistic transformations really differ? *Probabilistic Engineering Mechanics*, 24(4):577–584, 2009.

M. Loève. *Probability theory.* Springer, 1955.

S. Marque-Pucheu, G. Perrin, and J. Garnier. Efficient sequential experimental design for surrogate modeling of nested codes. *ESAIM Probability and Statistics*, 2018.

G. Matheron and F. Blondel. *Traité de géostatistique appliquée.* Paris, 1962.

M.D. McKay, R.J. Beckman, and W.J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

N. Meinshausen, G. Rocha, and B. Yu. A tale of three cousins: Lasso, ℓ_2-boosting, and dantzig. *Annals Statistics*, 35:2373–2384, 2007.

Y. Meyer and D. H. Salinger. *Wavelets and operators*, volume 1. Cambridge university press, Cambridge, 1995.

R. G. Miller. The jackknife - a review. *Biometrika*, 61:1–15, 1974.

I. Molchanov and S. Zuyev. Steepest descent algorithms in a space of measures. *Statistics and Computing*, 12(2):115–123, 2002.

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

A. Nataf. Détermination des distributions de probabilité dont les marges sont données. *Comptes Rendus de l'Académie des Sciences*, 225:42–43, 1962.

H. Niederreiter. Quasi-Monte Carlo methods and pseudo-random numbers. *Bulletin of the American Mathematical Society*, 84(6):957–1041, November 1978.

A. Nouy. Proper generalized decomposition and separated representations for the numerical solution of high dimensional stochastic problems. *Archives of computational methods in engineering*, 17:403–434, 2010.

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

J. E. Oakley and A. O'Hagan. Probabilistic sensitivity analysis of complex models: A Bayesian approach. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 66 (3):751–769, 2004.

A. O'Hagan. Curvefitting and optimal design for prediction. *J. R. Stat. Soc., Ser. B, Methodol.*, 40(1):1–42, 1978.

A. Papoulis and S. U. Pillai. *Probability, Random Variables and Stochastic Processes.* McGraw-Hill, Boston, 2002.

E. Parzen. An approach to time series analysis. *Ann. MAth. Stat.*, 32:951–989, 1962.

R. Paulo. Default priors for Gaussian processes. *Annals of Statistics*, 33(2):556–582, 2005.

K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(11):559–572, 1901.

P. Perdikaris, M. Raissi, A. Damianou, N. D. Lawrence, and G. E. Karniadakis. Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 473(2198), 2017.

G. Perrin. Active learning surrogate models for the conception of systems with multiple failure modes. *Reliability Engineering and System Safety*, 149:130–136, 2016.

G. Perrin. Adaptive calibration of a computer code with time-series output. *Journal of Statistical Planning and Inference*, 2018.

G. Perrin and C. Cannamela. A repulsion-based method for the definition and the enrichment of optimized space filling designs in constrained input spaces. *Journal de la Société Française de Statistique*, 158(1):37–67, 2017.

G. Perrin, C. Soize, D. Duhamel, and C. Funfschilling. Identification of polynomial chaos representations in high dimension from a set of realizations. *SIAM Journal on Scientific Computing*, 34(6):2917–2945, 2012.

G. Perrin, C. Soize, D. Duhamel, and C. Funfschilling. A posteriori error and optimal reduced basis for stochastic processes defined by a finite set of realizations. *SIAM/ASA J. Uncertainty Quantification*, 2:745–762, 2014.

G. Perrin, C. Soize, S. Marque-Pucheu, and J. Garnier. Nested polynomial trends for the improvement of Gaussian process-based predictors. *Journal of Computational Physics*, 346: 389–402, 2017.

V Picheny and D Ginsbourger. A nonstationary space-time Gaussian process model for partially converged simulations. *SIAM/ASA Journal on Uncertainty Quantification*, 1(1):37–67, 2013.

V. Picheny, D. Ginsbourger, O. Roustant, R.T Haftka, and N-H. Kim. Adaptive designs of experiments for accurate approximation of a target region. *Journal of Mechanical Design*, 132(7):071008–071008–9, 2010.

A. Pinkus. *Ridge functions*. Cambridge University Press, Cambridge, 2015.

F.J. Pinski, G. Simpson, A.M. Stuart, and H. Weber. Kullback–Leibler approximation for probability measures on infinite dimensional spaces. *SIAM Journal on Mathematical Analysis*, 47(6):4091–4122, 2015.

C. E. Rasmussen and C. K.I. Williams. *Gaussian processes for machine learning*. The MIT Press, Cambridge, 2006.

C. Robert. *The Bayesian Choice*. Springer-Verlag New York, New York, 2007.

M. Rosenblatt. Remarks on a multivariate transformation. *The Annals of Mathematical Statistics*, 23(3):470–472, 1952.

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

R. T. Rubinstein and D.P. Kroese. *Simulation and the Monte Carlo method*. John Wiley and Sons, Inc., Hoboken, New Jersey, 2008.

D. Rullière, N. Durrande, F. Bachoc, and C. Chevalier. Nested Kriging predictions for datasets with a large number of observations. *Statistics and Computing*, 28(4), 2018.

T.M. Russi. *Uncertainty Quantification with Experimental Data and Complex System Models*. PhD thesis, UC Berkeley, 2010.

J. Sacks, W.J. Welch, T.J. Mitchell, and H.P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–423, 1989.

A. Saltelli and I. Sobol. About the use of rank transformation in sensitivity of model output. *Reliability Engineering & System Safety*, 50(3):225 – 239, 1995.

A. Saltelli, K. Chan, and E. M. Scott. *Sensitivity analysis*. Wiley, New Jersey, 2000.

T.J. Santner, B.J. Williams, and W.I. Notz. *The Design and Analysis of Computer Experiments*. Springer-Verlag New York, 2003.

I.M. Sobol. Distribution of points in a cube and approximate evaluation of integrals. *U.S.S.R Comput. Maths. Math. Phys.*, 7:86–112, 1967.

I.M. Sobol. Sensitivity estimates for nonlinear mathematical models. *Mathematical Modeling & Computational Experiment*, 1:407–414, 1993.

M.I. Sobol. Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and Computers in Simulation*, 55:271–280, 2001.

C. Soize and R. Ghanem. Physical systems with random uncertainties: Chaos representations with arbitrary probability measure. *SIAM Journal on Scientific Computing*, 26:395–410, 2004.

M.L. Stein. *Interpolation of spatial data: some theory for Kriging*. Springer, New York, 1999.

R. Stroh, S. Demeyer, N. Fischer, J. Bect, and E. Vazquez. Sequential design of experiments to estimate a probability of exceeding a threshold in a multi-fidelity stochastic simulator. In *61th World Statistics Congress of the International Statistical Institute (ISI 2017)*, Marrakech, Morocco, July 2017.

R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B*, 58(1):267–288, 1989.

A. N. Tikhonov and V. Y. Arsenin. *Solution of Ill-posed Problems*. Winston & Sons, Washington, 1977.

R. Tuo, C.F Jeff Wu, and D. Yu. Surrogate modeling of computer experiments with different mesh densities. *Technometrics*, 56(3):372–380, 2014.

J.G. Van der Corput. Verteilungsfunktionen. I. Mitt. *Proc. Akad. Wet. Amsterdam*, 38: 813–821, 1935.

E. Vazquez and J. Bect. A sequential Bayesian algorithm to estimate a probability of failure. In *15th IFAC Symposium on System Identification, SYSID 2009*, Saint-Malo, France, July 2009.

B. Williams, D. Higdony, J. Gattiker, L. Moore, M. McKay, and S. Keller-McNulty. Combining experimental data and computer simulations, with an application to flyer plate experiments. *Bayesian Analysis*, 1(4):765–792, 2006.

H. Wold. *Estimation of Principal Components and Related Models by Iterative Least squares*. Academic Press, 1966.

C.-F. Wu and H. P. Wynn. The convergence of general step-length algorithms for regular optimum design criteria. *The Annals of Statistics*, 6(6):1273–1285, 11 1978.

D. Xiu. Fast numerical methods for stochastic computations: a review. *Communications in computational physics*, 5(2-4):242–272, 2009.

O. Zahm, P. Constantine, C. Prieur, and Y. Marzouk. Gradient-based dimension reduction of multivariate vector-valued functions. *HAL preprint : hal-1801.07922*.