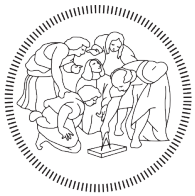

POLITECNICO
MILANO 1863

POLITECNICO DI MILANO
SCHOOL OF INDUSTRIAL AND INFORMATION ENGINEERING
Department of Mathematics "Francesco Brioschi"
M. Sc. in Mathematical Engineering

Multifidelity surrogate modeling in reservoir simulation: a functional cokriging approach to uncertainty quantification and prediction

Advisor: Dr. Alessandra MENAFOGLIO
Co-advisors: Dr. Ernesto DELLA ROSSA
Dr. Riccardo PELI

Student: Marco BEZZEGATO
Matr. 883091

Academic Year 2018-2019

Politecnico di Milano:
www.polimi.it

School of Industrial and Information Engineering:
www.ingindinf.polimi.it

Table of Contents

List of Figures

List of Tables

Abstract

Uncertainty quantification in reservoir simulation often entails the exploration of high-dimensional parameter spaces, so that a precise characterization of the uncertainty on outputs could be a computationally intensive problem. A valid approach in reducing such effort resides in incorporating information retrieved by approximated realizations of the model under study. The coupling of simulations with different levels of fidelity requires the knowledge of how they correlate with respect to their input configurations, and how to model such correlation for the scope of prediction. This thesis aims at testing the predictive capability of a surrogate model for uncertainty quantification of functional outputs of reservoir simulation in presence of two levels of fidelity. The proposed methodology considers a functional cokriging predictor for the high-level simulation outputs which takes advantage of the available data at different fidelity levels, through the dependence structure between/within levels. We introduce an alternative approach to covariance characterization for such data, improving its estimation process in presence of several input parameters. We also propose an extensive comparison with the corresponding univariate version (functional kriging) built with high-fidelity outputs only, and estimate their performances for different amounts of available data. The comparison shows that, in conditions of sparsity of high-fidelity observations, the prediction with functional cokriging is comparable or better than its alternative. In comparison with univariate approaches (functional kriging), the use of multi-fidelity cokriging allows obtaining high-quality predictions with fewer high-fidelity outputs, resulting in a sensible reduction of computational time. This results in an effective reduction of the required computational effort for the construction of the surrogate model, while a reasonable level of prediction quality is maintained.

Sommario

La quantificazione dell'incertezza nell'ambito delle simulazioni di giacimento spesso implica l'esplorazione di spazi ad alta dimensionalità, rendendo computazionalmente intensiva una precisa caratterizzazione dell'incertezza sugli output. Un approccio valido nel ridurre tale sforzo consiste nell'incorporare le informazioni ottenute da realizzazioni approssimate del modello in esame. L'accoppiamento di simulazioni con diversi livelli di fedeltà richiede la conoscenza di come queste correlino rispetto alle loro configurazioni di input e di come modellare tale correlazione nell'ambito della predizione. L'obbiettivo della tesi è quello di testare la capacità predittiva di un modello surrogato per la quantificazione dell'incertezza degli output funzionali di simulazioni di giacimento in presenza di due livelli di fedeltà. La metodologia proposta considera un predittore costruito tramite cokriging funzionale per gli output di alto livello, che utilizzi i dati disponibili a diversi livelli di fedeltà e la struttura di dipendenza presente tra i due livelli e tra le singole osservazioni in ciascun livello. Introduciamo un approccio alternativo alla caratterizzazione della covarianza per tali dati, migliorandone il processo di stima in presenza di numerosi parametri di input. Proponiamo anche un esteso confronto con la corrispondente versione univariata (kriging funzionale) costruita solo con output ad alta fedeltà, e stimiamo le loro prestazioni al variare della quantità di dati disponibili. Il confronto mostra che in condizioni di scarsità di osservazioni di alto livello la previsione con il metodo proposto è comparabile o migliore dell'alternativa. Inoltre, rispetto all'approccio univariato, la scelta di utilizzare il cokriging con diversi livelli di fedeltà permette di ottenere predizioni di buona qualità con un minore utilizzo di output ad alta fedeltà, comportando una sensibile riduzione del tempo di computazione. Ciò implica un efficace abbattimento dei tempi di calcolo per la costruzione del modello surrogato, mantenendo una ragionevole capacità predittiva.

Introduction

The continuous development of modern technologies in computer sciences have provided us with the ability to simulate various physical processes whose experimentation would be otherwise infeasible. However, the effective implementation of numerical approximations relies strongly on the choice of parameters whose exact value is mostly uncertain. Reservoir simulation of oil and gas recovery processes, for one, requires costly on-field inquiries to produce rock samples for parameters estimation. An important part of such numerical experiments is, therefore, the assessment of how uncertainty among these inputs propagates in the simulation output. The statistical theory of *uncertainty quantification* studies such problems.

Even when realistic reservoir models are available, their complexity often traduces in the inability to explore adequately the whole span of parameters' admissible values. A possible strategy to speed up the exploring procedure, for example, could be to consider simplifications of the original model whose demand in terms of computational resources is reduced. This approach (known as *multi-fidelity*) requires the evaluation of how the two resulting outputs are correlated, and how results from the latter model could replace the lack of knowledge from the former's ones.

The purpose of this thesis is to study the problem of uncertainty quantification of functional responses when multiple levels of a computer code are available. We approach the problem within the surrogate modeling framework focusing in particular on statistical models, for which outputs are considered as realizations of a stochastic process whose features are to be estimated. As the statistical theory on complex data has gained paramount importance in nowadays researches, we opt for an approach that studies simulation results as functions embedded in suitable abstract spaces. In this we follow the theory of *Functional Data Analysis* (FDA, [32]), which provides extensions of common statistical models (e.g. linear regression, clustering, etc.) to complex objects, such as curves or surfaces. Following the recent developments in the surrogate modeling research field, we concentrate our interest on techniques that consider a dependence structure between data with respect to the spatial displacement of the inputs. These methods are derived from the framework

of *geostatistics*, a branch of statistics originally meant to model phenomena arising in geography and geosciences. An important step in the construction of such models is the estimate of the covariance structure of the stochastic process under study arising from available data, which is often a delicate process when dealing with ample uncertainty spaces. In this sense we develop an alternative approach which exploits a cross-validation technique. The latter approach allows avoiding the need of sample variogram estimation by focusing on available data to optimize model parameters through minimization of the prediction error.

We consider a functional extension of the multivariate statistical model known as *cokriging*, implementing the aforementioned improvement to the original method and comparing it to its univariate declination (functional kriging). This is aimed at testing whether a secondary variable incorporation could improve the results in terms of predictive capability, or reducing the computational effort in building the surrogate model lessening the amount of expensive code runs needed. In particular we concentrate on reservoir simulation, for which an extensive literature dealing with model approximations is available.

The thesis structure is organized as follows:

- in Chapter 1 we present a review of the literature regarding functional extensions to kriging and cokriging and a brief introduction to surrogate modeling and flow in porous media;
- in Chapter 2 we illustrate the theory of the selected statistical model (functional cokriging) and describe the cross-validation procedure proposed for modeling the spatial dependence;
- in Chapter 3 we describe the extensive set of synthetic case studies developed for the comparison tests and the numerical procedures to produce their approximated counterparts;
- in Chapter 4 we present the original results concerning the comparison tests between our method and its univariate counterpart.

The last sections is devoted to conclusions, while Appendix A presents the original computer codes we developed for the analyses.

Chapter 1

State of the art

The aim of this Chapter is to introduce the reader to key concepts in functional geostatistics and meta-modeling we have employed in the development of this work. We propose a review of functional geostatistics in literature, explaining how previous works have inspired us in the development of our approach and what aspects of novelty our analysis brings to the overall subject. We propose also a brief introduction to modeling of flow in porous media focusing on field properties upscaling techniques, as this is the kind of computer experiments presented in order to validate our methods. A detailed discussion of the statistical model we adopt will follow in Chapter 2.

1.1 Geostatistics for data in Hilbert spaces

In recent years the interest of the scientific community has focused on studying phenomena in which continuous observations can be modeled as curves, surfaces or more general types of functions (see Ramsay and Silverman [32] for a complete discussion). Alternatively to the usual approach dealing with independent identically distributed (i.i.d.) random variables, several examples can be provided of situations in which observations are endowed with a structure of dependence with respect to time, space, or both (see Figure 1.1 for an example). Thus the need for developing a theory for spatially dependent functional data which can somehow expand well-known concepts of scalar and multivariate spatial variability and spatial prediction.

1.1.1 Functional random fields

Let $(\Omega, \mathfrak{F}, \mathbb{P})$ be a probability space and H a separable Hilbert space endowed with the inner product $\langle \cdot, \cdot \rangle$ and the induced norm $\|\cdot\|$ (elements of H are $x : T \to \mathbb{R}$, with $T \subseteq \mathbb{R}$ compact). A measurable function $\mathcal{X} : \Omega \to H$

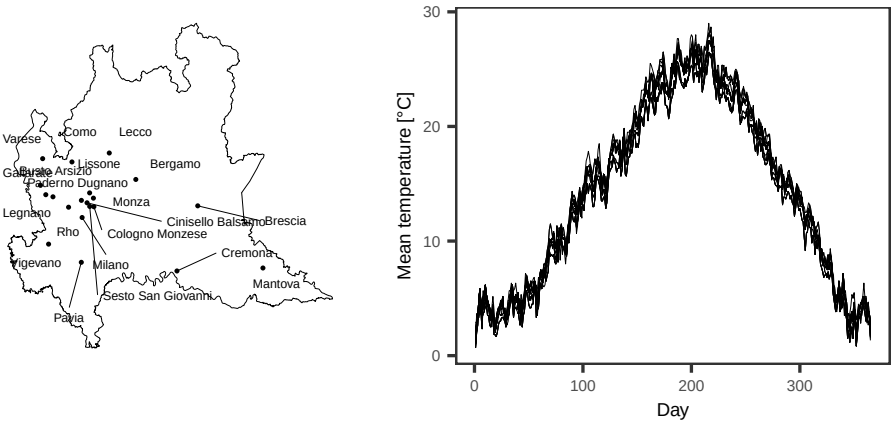

Figure 1.1: Example of functional dataset, averages (over 5 years) of mean daily temperature curves (right) observed at 20 cities in Lombardia, Italy (left).

whose realization $x : T \to \mathbb{R}$ is an element of H is called *functional random variable*, while the realization is the *functional datum*. Consider a collection of functional random variables indexed over a subset D of $\mathbb{R}^d$

$$\{\mathcal{X}_{\mathbf{s}},\ \mathbf{s} \in D \subseteq \mathbb{R}^d\}, \tag{1.1}$$

such a collection will be called *functional random field*.

In this framework, a functional dataset $\mathcal{X}_{\mathbf{s}_1}, \ldots, \mathcal{X}_{\mathbf{s}_n}$ is the collection of n observations of the random field (1.1) relative to n locations $\mathbf{s}_1, \ldots, \mathbf{s}_n \in D$. In non-trivial situations, a vector of observations $\boldsymbol{\mathcal{X}} = (\mathcal{X}_{\mathbf{s}_1}, \ldots, \mathcal{X}_{\mathbf{s}_n})^T$ is characterized by a structure of spatial dependence reflecting the covariance structure of the generating random process (1.1). For $1 \leq p \leq \infty$ denote with $L^p(\Omega; H)$ the vector space of measurable functions $\boldsymbol{\mathcal{X}} : \Omega \to H$ such that $\int_\Omega \|\boldsymbol{\mathcal{X}}\|^p \, \mathbb{P}(d\omega) = \mathbb{E}\left[\|\boldsymbol{\mathcal{X}}\|^p\right] < \infty$ – i.e. $\|\boldsymbol{\mathcal{X}}\| \in L^p(\Omega)$ – which is a Banach space with respect to the norm $\|\boldsymbol{\mathcal{X}}\|_{L^p(\Omega,H)} = \left(\mathbb{E}\left[\|\boldsymbol{\mathcal{X}}\|^p\right]\right)^{1/p}$. Following Menafoglio et al. [29] we recall the main operative assumption, that is

Assumption 1 (Square integrability)**.** *Each element* $\mathcal{X}_{\mathbf{s}},\ \mathbf{s} \in D$ *of the random field* (1.1) *belongs to* $L^2(\Omega; H)$*.*

Given Assumption 1, the expected value $m_{\mathbf{s}}$ of the random field can be defined by Bochner integral as

$$m_{\mathbf{s}} = \int_\Omega \boldsymbol{\mathcal{X}}_{\mathbf{s}}(\omega)\mathbb{P}(d\omega), \quad \mathbf{s} \in D, \tag{1.2}$$

while a measure of (global) spatial dependence can be defined by the covariance function

$$C(\mathbf{s}_i, \mathbf{s}_j) := \mathrm{Cov}(\mathcal{X}_{\mathbf{s}_i}, \mathcal{X}_{\mathbf{s}_j}) = \mathbb{E}\left[\langle \mathcal{X}_{\mathbf{s}_i} - m_{\mathbf{s}_i}, \mathcal{X}_{\mathbf{s}_j} - m_{\mathbf{s}_j} \rangle\right], \quad \mathbf{s}_i, \mathbf{s}_j \in D \tag{1.3}$$

which induces naturally the concepts of (global) variance and semivariogram, respectively

$$\sigma^2(\mathbf{s}) = \mathbb{E}\left[\|\mathcal{X}_{\mathbf{s}} - m_{\mathbf{s}}\|^2\right], \quad \mathbf{s} \in D, \tag{1.4}$$

$$\gamma(\mathbf{s}_i, \mathbf{s}_j) = \frac{1}{2}\mathrm{Var}(\mathcal{X}_{\mathbf{s}_i} - \mathcal{X}_{\mathbf{s}_j}), \quad \mathbf{s}_i, \mathbf{s}_j \in D. \tag{1.5}$$

These functions preserve the same properties as their finite-dimensional analogue (see Chilès and Delfiner [9]). Moreover there is a strict relation between global covariance and semivariogram with their operatorial counterparts, in that they represent the trace of the corresponding operator (see Menafoglio et al. [29] for a theoretical justification), therefore they are also known respectively as *trace-covariance* and *trace-semivariogram*.

Concerning the notion of stationarity and isotropy, global definitions can be stated as follows:

Definition 1. *A process* $\{\mathcal{X}_{\mathbf{s}},\ \mathbf{s} \in D \subseteq \mathbb{R}^d\}$ *is said to be* (globally) second order stationary *if the following conditions hold:*

(i) $\mathbb{E}[\mathcal{X}_{\mathbf{s}}] = m,\ \forall \mathbf{s} \in D \subseteq \mathbb{R}^d$;

(ii) $\mathrm{Cov}(\mathcal{X}_{\mathbf{s}_i}, \mathcal{X}_{\mathbf{s}_j}) = \mathbb{E}\left[\langle \mathcal{X}_{\mathbf{s}_i} - m_{\mathbf{s}_i}, \mathcal{X}_{\mathbf{s}_j} - m_{\mathbf{s}_j} \rangle\right] = C(\mathbf{h}), \quad \forall \mathbf{s}_i, \mathbf{s}_j \in D \subseteq \mathbb{R}^d$ *where* $\mathbf{h} = \mathbf{s}_i - \mathbf{s}_j$.

A process is $\{\mathcal{X}_{\mathbf{s}},\ \mathbf{s} \in D \subseteq \mathbb{R}^d\}$ *is said to be* (globally) intrinsically stationary *if the following hold:*

(i) $\mathbb{E}[\mathcal{X}_{\mathbf{s}}] = m,\ \forall \mathbf{s} \in D \subseteq \mathbb{R}^d$;

(ii') $\mathrm{Var}(\mathcal{X}_{\mathbf{s}_i} - \mathcal{X}_{\mathbf{s}_j}) = \mathbb{E}\left[\left\|\mathcal{X}_{\mathbf{s}_i} - \mathcal{X}_{\mathbf{s}_j}\right\|^2\right] = 2\gamma(\mathbf{h}), \quad \forall \mathbf{s}_i, \mathbf{s}_j \in D \subseteq \mathbb{R}^d$ *where* $\mathbf{h} = \mathbf{s}_i - \mathbf{s}_j$.

Definition 2. *A second order stationary process* $\{\mathcal{X}_{\mathbf{s}},\ \mathbf{s} \in D \subseteq \mathbb{R}^d\}$ *is said to be* isotropic *if*

$$\mathrm{Cov}(\mathcal{X}_{\mathbf{s}_i}, \mathcal{X}_{\mathbf{s}_j}) = C(\|\mathbf{h}\|), \quad \forall \mathbf{s}_i, \mathbf{s}_j \in D \subseteq \mathbb{R}^d, \quad \mathbf{h} = \mathbf{s}_i - \mathbf{s}_j, \tag{1.6}$$

where $\|\cdot\|$ *is a norm on* D.

The concept of global trace-semivariogram was already present in a pointwise formulation in Giraldo et al. [16], then Menafoglio et al. [29] provided the aforementioned definition which extends the previous one to more complex situations, e.g. when data can be embedded in a generic Sobolev space.

1.1.2 Kriging and Cokriging for real-valued data

Kriging [27] is a minimum-mean-squared-error method of spatial prediction. It (usually) depends on second order properties of the stochastic process $Z(\cdot)$ under study, which is known up to a set of observations $\boldsymbol{Z} = (Z_{\mathbf{s}_1}, \ldots, Z_{\mathbf{s}_n})$. Assuming a structure of spatial dependence between observations, the aim of kriging is to predict the response at an "unvisited" site $\mathbf{s}_0$ with a linear

combination of available data, that is

$$Z^{\star}_{\mathbf{s}_0} = \sum_{i=1}^{n} \lambda_i Z_{\mathbf{s}_i}. \tag{1.7}$$

The choice of weights for the interpolation is performed solving the following constrained minimization problem:

$$\begin{aligned} \min_{\lambda_1,\dots,\lambda_n \in \mathbb{R}} \quad & \mathbb{E}\left[(Z^{\star}_{\mathbf{s}_0} - Z_{\mathbf{s}_0})^2\right] \\ \text{subject to:} \quad & \mathbb{E}\left[Z^{\star}_{\mathbf{s}_0} - Z_{\mathbf{s}_0}\right] = 0. \end{aligned} \tag{1.8}$$

Cokriging is the multivariate extension of kriging in presence of secondary data that show spatial correlation with the primary variable, often preferred when the main attribute of interest is sparse while related secondary informations are abundant. Given the available data $\{\boldsymbol{Z}^{(k)} = (Z^{(k)}_{\mathbf{s}_1}, \dots, Z^{(k)}_{\mathbf{s}_{n_k}});\ k = 1, \dots, K\}$ measured at sample locations $\mathbf{s}_i, \dots, \mathbf{s}_{n_k}$ (which are not constrained to be in the same position nor the same amount for each variable) the cokriging predictor for the j-th component $Z^{(j)}_{\mathbf{s}_0}$ takes the following form:

$$Z^{(j)\star}_{\mathbf{s}_0} = \sum_{k=1}^{K} \sum_{i=1}^{n_k} \lambda^{(k)}_i Z^{(k)}_{\mathbf{s}_i}. \tag{1.9}$$

As for the univariate approach, optimal weights are computed solving minimization problem (1.8).

Depending on the degree of stationarity assumed for the underlying process, kriging can be applied through different methods (see Cressie [10] and Chilès and Delfiner [9] for a complete discussion) such as:

- *ordinary kriging*, in which a constant unknown mean is assumed throughout the whole domain;
- *universal kriging*, in which the process is modeled as sum of a deterministic term (*drift*), such as a linear trend model based on known covariates, and a zero-mean stochastic term (*residual*):

$$Z_{\mathbf{s}} = m_{\mathbf{s}} + \delta_{\mathbf{s}} = \sum_{l=0}^{L} \beta_l f_l(\mathbf{s}) + \delta_{\mathbf{s}}. \tag{1.10}$$

Both methods can be extended to cokriging, assuming a similar decomposition for each component k of the multivariate process:

$$Z^{(k)}_{\mathbf{s}} = m^{(k)}_{\mathbf{s}} + \delta^{(k)}_{\mathbf{s}} = \sum_{l=0}^{L} \beta^k_l f^k_l(\mathbf{s}) + \delta^{(k)}_{\mathbf{s}}, \quad k = 1, \dots, K, \tag{1.11}$$

which includes the ordinary cokriging case when $L = 0$ and $f^k_0(\mathbf{s}) = 1$ for each location $\mathbf{s}$.

1.1.3 Kriging and Cokriging for functional data review

The first attempt in functional geostatistics is proposed by Goulard and Voltz [19], who consider functions known up to a finite and small set of points. Observations are thus embedded in a classic multivariate framework, and two approaches are presented: either performing a cokriging prediction and then reconstructing the shape of the whole unsampled curve fitting a parametric model (*Cokrige first, Fit later Predictor* or CFP) or fitting a parametric model over the whole sample of curves and predicting the parameters at an unsampled site (*Fit first, Cokrige later Predictor* of FCP). While the former method relies strongly on the assumption that prediction of individual values would honour the shape of the curve and constrains the analysis only on equally-sampled curves, the latter assumes that all curves can be satisfactorily fitted with the same parametric model, though relaxing the need of equal sampling for all observations in the sample. Goulard and Voltz proposed also a fully-functional approach to kriging (*Curve Kriging Predictor* or CKP), defining the predictor for curves in a temporal domain T as

$$\mathcal{X}^{\star}_{\mathbf{s}_0}(t) = \sum_{i=1}^{n} \lambda_i(t)\mathcal{X}_{\mathbf{s}_i}(t), \quad t \in T, \tag{1.12}$$

and carrying out the computation of empirical variogram through numerical integration or fitting a parametric model.

CKP approach was recovered and developed by Giraldo et al. [17] who improved the smoothing process by means of a functional extension of Leave-One-Out Cross-Validation [35] along with the introduction of basis representation for both observations and weights, namely

$$\mathcal{X}_{\mathbf{s}_i}(t) = \sum_{l=1}^{L} a_{il}B_l(t), \quad i = 1, \ldots, n,\ t \in T, \tag{1.13}$$

$$\lambda_i(t) = \sum_{l=1}^{L} b_{il}B_l(t), \quad i = 1, \ldots, n,\ t \in T, \tag{1.14}$$

where $\{B_l(t)\}_{l=1}^{L}$ are appropriate basis functions (e.g. Fourier basis functions or B-splines). The use of finite dimensional representation allows to reduce functional prediction to a multivariate problem, resembling therefore FCP technique, while the choice for a proper number L of basis functions is performed through Cross-Validation or other standard approaches in FDA framework (see Ramsay and Silverman [32] for a complete discussion).
Another approach exploiting basis expansion is the one of Nerini et al. [30], in which the proposed expansion for weights is the following:

$$\lambda_i(t, v) = \sum_{j=1}^{L}\sum_{l=1}^{L} c^i_{jl}B_j(t)B_l(v), \quad t, v \in T. \tag{1.15}$$

This choice of weights allows to build an infinite-dimensional extension to cokriging predictor, namely

$$\mathcal{X}^*_{\mathbf{s}_0}(t) = \sum_{i=1}^{n} \int_T \lambda_i(t,v)\mathcal{X}_{\mathbf{s}_i}(v)dv, \quad t \in T, \tag{1.16}$$

which is known as *cokriging predictor based on functional data*. In both cases, however, the choice of basis expansion could produce difficulties in estimating unidirectional (marginal) variograms and consequently fitting Linear Models of Coregionalization [9] due to high dimension [20].

Bohorquez et al. [3] introduced a representation for curves through Empirical Functional Principal Components [21], which allows to work with an orthogonal basis, i.e. the eigenfunctions $\{\xi_k(t)\}_{k=1}^K$ of the covariance operator of $\mathcal{X}_{\mathbf{s}}$

$$C(\mathcal{Y}) = \mathbb{E}[\langle \mathcal{X}_{\mathbf{s}}, \mathcal{Y}\rangle \mathcal{X}_s], \quad \mathcal{Y} \in L^2, \tag{1.17}$$

and to represent curves with their associates principal components scores, namely

$$f_j(\mathbf{s}_i) = \langle \mathcal{X}_{\mathbf{s}_i}, \xi_j\rangle, \quad j = 1,\dots,K,\ i = 1,\dots,n. \tag{1.18}$$

This approach reduces significantly the number of basis functions needed to represent observations while keeping a fairly good approximation of them, mostly due to the suitable property of variance maximization along those directions.

After these attempts to coerce functional observations into a classic multivariate setting, several authors proposed a much general framework in which functional observations are considered as elements of suitable abstract spaces. This new paradigm known as *Object-Oriented Data Analysis* or OODA (see Marron and Alonso [26] for a recent review), although more complex from the technical point of view, allow to exploit all the information embedded in data incorporating raw observations in the estimation process, while former techniques required "lossy" data representations.

Following this approach, Menafoglio et al. [29] developed the framework for functional random fields shown in Section 1.1.1, recovering the CKP predictor (1.12) and solving the original minimization problem (1.8) in light of the new setting. Another improvement was the proposal of an iterative algorithm to overcome the issue of estimating both drift component and spatial dependence structure of a non-stationary process, which extended the approach of Caballero et al. [8] and Ignaccolo et al. [22].

Finally, Bohorquez et al. [4] and Grujic et al. [20] extended the analysis to multivariate functional random fields, namely

$$\{\boldsymbol{\Xi}_{\mathbf{s}},\ s \in D \subseteq \mathbb{R}^d\}, \tag{1.19}$$

where $\mathbf{\Xi_s} = (\mathcal{X}_\mathbf{s}^1(t), \dots, \mathcal{X}_\mathbf{s}^P(t))$ are elements of $\mathcal{H}^P = \mathcal{H} \oplus \cdots \oplus \mathcal{H}$, direct sum of abstract spaces (e.g., Hilbert or Sobolev space). While the former explored only the case of constant mean processes, the latter embraced the lesson of Menafoglio et al. [29] and developed a complete framework for universal co-kriging of functional data, which can benefit from secondary informations in prediction that are not even constrained to belong to the same abstract space, e.g. mixing functional observations with scalar ones. For the development of our analysis we will follow the latter approach, which will be detailed in Chapter 2, as it allows to preserve all the information within observations, giving us a greater flexibility in modeling spatial behaviours and relationship between primary and secondary data.

1.2 Meta-modeling in computer simulation

Since the beginning of computer simulation as a way to overcome limits of physical experimentation, often too complex or even infeasible for some processes, assessing the relationship between inputs and outputs of a simulation has been of primary importance. Mathematical models (and their computer counterparts) have reached a considerable level of complexity, which often implies a significant computational effort: this, paired with the uncertainty over input parameters involved in modeling, has made the exploration of the whole input space an infeasible yet necessary step for the majority of numerical models exploited nowadays.

1.2.1 Surrogate modeling

One of the most developed approaches to the issue relies on the construction of meta-models (also known as surrogate models), approximations of the full numerical model built from a small set of simulations, which allow to obtain a "cheap" response (in terms of computational effort) useful in several processes as prediction, uncertainty quantification and parameter optimization. One of the approaches first proposed in the literature on this matter is the response surfaces method [6], which entails fitting scalar simulation outputs with a polynomial (usually of second-order) through linear regression.

In recent years the literature on this matter has been largely developed, with several proposals spacing from improvements to the aforementioned response surfaces to cutting-edge methods involving machine learning and neural networks (see Yeten et al. [39], Forrester and Keane [15] and Aliyev and Durlofsky [2] for an overview of principal methodologies). Among these techniques kriging-based approaches, exploiting spatial dependence of outputs with respect to their input configurations' mutual distances, have gained great im-

portance as an effective way to reduce computational effort and the overall number of simulations of the original code.

1.2.2 Multi-fidelity modeling

An important improvement in meta-modeling theory was the introduction of the concept of multi-fidelity simulation. This paradigm consists on building several copies of the same code with different levels of complexity, e.g. varying the amount of underlying modeled physics or the size of single discrete units of computation. Therefore it is possible to provide for the same input configuration several outputs mutually correlated and with different demand for computational resources. In this sense, each level provides an improving approximation of the underlying processes involved, going from a broader and computationally efficient one (usually indicated as low-fidelity or *lo-fi*) to the finest and most computationally intensive available (high-fidelity or *hi-fi*).

Kennedy and O'Hagan [23] proposed to build a surrogate model exploiting several levels of code and assuming a structure of covariance between them, giving rise to a "dependency chain" in which each higher-fidelity level is related to the subsequent one by an autoregressive model. The resulting prediction of a hi-fi output at an unsampled configuration is a function of both other existing hi-fi samples and lower level ones obtained from the same input configuration. This technique shares, albeit from a bayesian point of view, the same theoretical framework of kriging, leading to a similar predictor and providing the same estimate of the variance.

On these initial assumptions several authors have studied the multi-fidelity approach as a technique to further reduce the computational effort of building a surrogate, extending the analysis from the original scalar problem by Kennedy and O'Hagan to multi-dimensional outputs and even functional ones in several context such as fluid dynamics or geosciences (see e.g. Thenon et al. [37] for an application in reservoir engineering).

In this work we focus the analysis on functional outputs in presence of two levels of fidelity in computer code, building the functional cokriging predictor described by Grujic et al. [20]. This approach is used to estimate outputs across input space and relative uncertainty over input parameters. Our choice is motivated by three main aspects:

- great flexibility of cokriging with respect to other methods, since the technique allows to study both stationary and non-stationary processes;
- the property of being an exact interpolator, i.e, predicting the response at any sampled location produce the simulation output itself, which is desirable when dealing with totally deterministic algorithms;

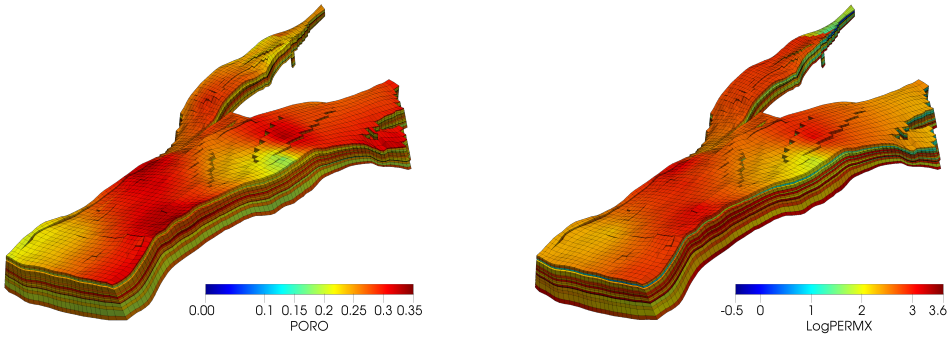

Figure 1.2: Example of grid discretization of Norne reservoir model, coloured by cells porosity (left) and log-permeability (right).

- the possibility of working with functional observation without any preprocessing step, while other methods require scalar or at most vectorial output, which often entails a higher level of information loss.

1.3 Reservoir simulation of geological models

Simulation of an oil reservoir refers to the construction of a mathematical model, i.e. a set of equations subject to certain assumptions, that describes the physical processes taking place in a reservoir. These models are usually formulated in terms of coupled Ordinary or Partial Differential Equations (ODEs/PDEs) over a continuous domain, which can be subsequently discretized into a grid representation (see Figure 1.2 for an example) and solved with schemes such as Finite Elements [24] or Finite Volume methods [14] in order to produce an approximation of the solution for the general equations.

1.3.1 Mathematical model

The theoretical framework of reservoir modeling is the description of fluid flow in porous media, which is characterized by two main equations:

- macroscopic fluid mass conservation:

$$\frac{\partial (\Phi \rho)}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = \tilde{m}, \tag{1.20}$$

  where Φ is the porosity of the reservoir, ρ and $\mathbf{u}$ respectively the fluid's density and macroscopic velocity and $\tilde{m}$ a source/sink term;

- Darcy's Law of fluid flow in porous media [11]:

$$\mathbf{u} = -\frac{\mathbf{K}}{\mu} \left(\nabla p - \rho \mathbf{g} \right), \tag{1.21}$$

where $\mathbf{u}$ is fluid's velocity, ρ, p and μ respectively its density, pressure and viscosity, $\mathbf{K}$ the symmetric tensor of *absolute permeability* (or *hydraulic conductivity*) [1] and $\mathbf{g}$ the gravity vector.

The two equations coupled under the assumption of constant density allow to derive the pressure equation:

$$-\nabla \cdot \left[\frac{\mathbf{K}}{\mu}\left(\nabla p - \rho \mathbf{g}\right)\right] = \tilde{q}, \tag{1.22}$$

which can provide the steady-state behaviour of the model in terms of pressure and velocity in presence of a single-phase flow (e.g., water, oil or gas).

The equations describing two-phase flow can be derived writing Darcy's law for each phase, i.e., in absence of gravity:

$$\mathbf{u}_j = -\frac{k_{rj}}{\mu_j}\nabla p_j, \tag{1.23}$$

where j refers to the phase and k_{rj} is the *relative permeability* [1] of the j-th phase ($j = w$ for water and $j = o$ for oil). The former equation is paired with mass conservation for each phase expressed in terms of saturation (volume fraction) S_j:

$$\frac{\partial\left(\Phi \rho_j S_j\right)}{\partial t} + \nabla \cdot \left(\rho_j \mathbf{u_j}\right) = \tilde{m}_j, \tag{1.24}$$

which, under the assumption of constant density and negligible capillary pressure (i.e. $p_c(S_w) = p_w - p_o = 0$), gives the following system of equations:

$$\begin{aligned} \frac{\partial S_w}{\partial t} + \nabla \cdot \left[\mathbf{u}_t f\left(S_w\right)\right] &= \tilde{q}_w \\ -\nabla \cdot \left[\lambda_t\left(S_w\right)\mathbf{K}\nabla p\right] &= \tilde{q}_t, \end{aligned} \tag{1.25}$$

where $\mathbf{u}_t$ is the total fluid velocity, $\lambda_t = \frac{k_{rw}}{\mu_w} + \frac{k_{ro}}{\mu_o}$ is the total mobility and $f(S_w)$ is the Buckley-Leverett fractional fluid function [7]. Another mathematical model for reservoir simulation is the black-oil model [38], which generalizes the aforementioned ones to situations involving a gaseous phase and the presence of dissolved gas into oil. In this work we will exploit the latter model, since our interest is to simulate processes in which are involved both water, oil and gas phases.

1.3.2 Upscaling techniques

Given the fine scale nature of field properties, such as rock porosity or permeability, realistic reservoir models involve a large number of computational cells. This fine scale description could make the simulation costly from the computational point of view, while the uncertainty over parameters (often generated from a few field measurements) should require several runs to produce a correct estimate of the output. One way to reduce single-run computational cost

is to consider a coarse realization of the reservoir with few cells, estimating the equivalent properties value of a coarse scale cell from the fine-scale cells laying within. We distinguish between three main approaches:

- purely local procedures, in which coarse scale parameters are computed by considering only the fine scale region corresponding to the target coarse block;
- global procedures, in which the whole fine scale model is simulated for the calculation of coarse scale parameters;
- extended local procedures, in which a region corresponding to each coarse scale plus a border (or ring) is considered when computing upscaled parameters.

Upscaling procedures largely depend on the setting of reservoir and types of flow considered: while single-phase flow needs the upscaling of porosity and absolute permeability, two-phase flow requires also relative permeability to be upscaled. However, it is possible to obtain an accurate coarse scale model for a two-phase flow upscaling only porosity and absolute permeability, neglecting the contribution of relative permeability in the procedure and leading to a more efficient computation of the coarse model: some theoretical justifications to this approach are presented in Durlofsky [13]. The adopted upscaling scheme in this work is the following:

- porosity on the coarse scale will be computed simply averaging fine scale values over each target coarse block, since it suffices to ensure that pore volume is conserved between different scales;
- permeability will be upscaled with a single-phase equivalent volume flow approach (see Rizzo [34, Chapter 2] for a technical discussion), therefore taking into account only absolute permeability, which is consistent with the aforementioned results.

Chapter 2

Universal Trace-Cokriging for functional data

The aim of this chapter is to present the problem of optimal spatial prediction for multivariate functional random fields and develop a universal Trace-Cokriging method from the theoretical point of view, detailing the mathematical formulation of the predictor, the analytical solution to the problem and the available approaches to parameters inference. In this work we will follow the approach of Menafoglio et al. [29] and Grujic et al. [20] for the development of the predictor, introducing a cross-validation approach to range optimization within the fitting of valid covariance models.

2.1 Theoretical framework

Let $H_k, k = 1, \dots, K$ a separable Hilbert space endowed with the inner product $\langle \cdot, \cdot \rangle_{H_k}$ and D a Euclidean spatial domain in $\mathbb{R}^d, d \geq 1$. Given a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ we indicate by $\mathcal{X}_{\mathbf{s}}^{(k)}$, where $\mathbf{s}$ is a spatial index over D, a random element on $(\Omega, \mathcal{F}, \mathbb{P})$ in H_k. Since we will consider multivariate random fields, we denote with $\{\boldsymbol{\mathcal{X}}_{\mathbf{s}},\ \mathbf{s} \in D\}$ a multivariate random process on $(\Omega, \mathcal{F}, \mathbb{P})$ taking values in the Cartesian space $H^K = H_1 \times H_2 \times \dots H_K$: each element $\boldsymbol{\mathcal{X}}_{\mathbf{s}}$ is a vector of K random elements in $H_1, \dots, H_K$ respectively:

$$\boldsymbol{\mathcal{X}}_{\mathbf{s}} = \left(\mathcal{X}_{\mathbf{s}}^{(1)}, \dots, \mathcal{X}_{\mathbf{s}}^{(K)}\right)^T. \tag{2.1}$$

To define first and second order properties of the field we proceed by analogy with the classical framework described in Chapter 1, calling $\boldsymbol{m}_{\mathbf{s}} \in H^K$ the spatial mean of the process at location $\mathbf{s}$ in D, that is:

$$\boldsymbol{m}_{\mathbf{s}} = \mathbb{E}\left[\boldsymbol{\mathcal{X}}_{\mathbf{s}}\right] = \left(m_{\mathbf{s}}^{(1)}, \dots, m_{\mathbf{s}}^{(K)}\right)^T, \quad m_{\mathbf{s}}^{(k)} = \mathbb{E}\left[\mathcal{X}_{\mathbf{s}}^{(k)}\right], \tag{2.2}$$

and defining the map $C : D \times D \to \mathbb{R}^{K\times K}$ that determines the trace-covariograms and cross-trace-covariograms as follows:

$$\begin{aligned}(\mathbf{s}, \boldsymbol{u}) &\mapsto C(\mathbf{s}, \boldsymbol{u}) \in \mathbb{R}^{K\times K} \\ C_{kl}(\mathbf{s}, \boldsymbol{u}) &= \mathbb{E}\left[\langle \mathcal{X}_{\mathbf{s}}^{(k)} - m_{\mathbf{s}}^{(k)}, \mathcal{X}_{\boldsymbol{u}}^{(l)} - m_{\boldsymbol{u}}^{(l)} \rangle\right].\end{aligned} \tag{2.3}$$

In this work we assume that every element $\mathcal{X}_{\mathbf{s}}^{(k)}$ of the multivariate process $\boldsymbol{\mathcal{X}}_{\mathbf{s}}$ is non stationary, i.e., it can be represented as a sum of a deterministic mean (drift) and a zero-mean globally second order stationary residual

$$\mathcal{X}_{\mathbf{s}}^{(k)} = m_{\mathbf{s}}^{(k)} + \delta_{\mathbf{s}}^{(k)}, \tag{2.4}$$

where drift term is assumed to be non-constant within space D and it is modeled as a functional linear model

$$m_{\mathbf{s}}^{(k)} = \sum_{l=0}^{L} a_l^{(k)} f_l(\mathbf{s}) \tag{2.5}$$

where $a_l^{(k)}$ are (functional) coefficients in H_k and $f_l(\cdot)$ scalar regressors known over the entire domain D. Furthermore, we assume that residuals are globally second order stationary in the sense of Menafoglio et al. [29], that is, the multivariate trace-covariogram structure depends only on the increments between locations:

$$C(\mathbf{s}, \mathbf{u}) = \tilde{C}(\mathbf{s} - \boldsymbol{u}) \quad \forall \mathbf{s}, \boldsymbol{u} \in D, \tag{2.6}$$

for some map $\tilde{C} : D \to \mathbb{R}^{K\times K}$, which for ease of notation will be denoted hereafter simply by C.

2.2 Predictor equations

Consider a series of measurement locations $\mathbf{s}_1, \ldots, \mathbf{s}_{N_j}$ $(j = 1, \ldots, K)$ over the domain D and the partial observation $\mathcal{X}_{\mathbf{s}_1}^{(j)}, \ldots, \mathcal{X}_{\mathbf{s}_{N_j}}^{(j)}$ of the j-th component of the multivariate process at these locations. Under the former assumption, the aim of trace-cokriging is to predict the k-th element $\mathcal{X}_{\mathbf{s}_0}^{(k)}$ of $\boldsymbol{\mathcal{X}}_{\mathbf{s}}$ at a target location $\mathbf{s}_0$ in D with the best linear unbiased predictor within the class of linear predictors of the form:

$$\mathcal{X}_{\mathbf{s}_0}^{(k)\lambda} = \sum_{j=1}^{K} \sum_{i=1}^{N_j} \lambda_{ji} \mathcal{X}_{\mathbf{s}_i}^{(j)}. \tag{2.7}$$

In order to find the optimal weight $\lambda_{ji}^{\star},\ j = 1, \ldots, K,\ i = 1, \ldots, N_j$, we proceed by solving the constrained optimization problem of minimizing mean square error of prediction under the unbiasedness constraint, that is:

$$\begin{aligned}\min_{\substack{\lambda_{ji}\in\mathbb{R} \\ j=1,\ldots,K \\ i=1,\ldots,N_j}} \quad & \mathbb{E}\left[\left\|\mathcal{X}_{\mathbf{s}_0}^{(k)\lambda} - \mathcal{X}_{\mathbf{s}_0}^{(k)}\right\|^2\right] \\ \text{subject to:} \quad & \mathbb{E}\left[\mathcal{X}_{\mathbf{s}_0}^{(k)\lambda} - \mathcal{X}_{\mathbf{s}_0}^{(k)}\right] = 0.\end{aligned} \tag{2.8}$$

Recalling the random element decomposition (2.4) and (2.5) we can write

$$\mathbb{E}\left[\mathcal{X}_{\mathbf{s}_0}^{(k)\lambda}\right] = \sum_{j=1}^{K}\sum_{i=1}^{N_j}\lambda_{ji}m_{\mathbf{s}_i}^{(j)}, \tag{2.9}$$

so that the unbiasedness constraint in (2.8) reads as

$$\begin{aligned}&\sum_{i=1}^{N_k}\lambda_{ki}f_l(\mathbf{s}_i) = f_l(\mathbf{s}_0) \quad \forall l = 0,\dots,L;\\ &\sum_{i=1}^{N_j}\lambda_{ji}f_l(\mathbf{s}_i) = 0 \quad j \neq k,\ \forall l = 0,\dots,L.\end{aligned} \tag{2.10}$$

The objective functional in (2.8) can be expanded in terms of the trace- and cross-trace-covariograms as

$$\begin{aligned}\mathbb{E}\left[\left\|\mathcal{X}_{\mathbf{s}_0}^{(k)\lambda} - \mathcal{X}_{\mathbf{s}_0}^{(k)}\right\|^2\right] = {} & C_{kk}(\mathbf{0})\\ & + \sum_{j=1}^{K}\sum_{i=1}^{N_j}\sum_{j'=1}^{K}\sum_{i'=1}^{N_j'}\lambda_{ji}\lambda_{j'i'}C_{jj'}(\mathbf{s}_i - \mathbf{s}_{i'})\\ & - 2\sum_{j=1}^{K}\sum_{i=1}^{N_j}\lambda_{ij}C_{kj}(\mathbf{s}_i - \mathbf{s}_0),\end{aligned} \tag{2.11}$$

while performing a Lagrangian relaxation, introducing $K(L+1)$ multipliers to account for the constraint, leads to the following equation:

$$\begin{aligned}\mathbb{E}\left[\left\|\mathcal{X}_{\mathbf{s}_0}^{(k)\lambda} - \mathcal{X}_{\mathbf{s}_0}^{(k)}\right\|^2\right] = {} & C_{kk}(\mathbf{0})\\ & + \sum_{j=1}^{K}\sum_{i=1}^{N_j}\sum_{j'=1}^{K}\sum_{i'=1}^{N_j'}\lambda_{ji}\lambda_{j'i'}C_{jj'}(\mathbf{s}_i - \mathbf{s}_{i'})\\ & - 2\sum_{j=1}^{K}\sum_{i=1}^{N_j}\lambda_{ij}C_{kj}(\mathbf{s}_i - \mathbf{s}_0)\\ & + 2\sum_{l=0}^{L}\mu_{kl}\left(\sum_{i=1}^{N_k}\lambda_{ki}f_l(\mathbf{s}_i) - f_l(\mathbf{s}_0)\right)\\ & + 2\sum_{l=0}^{L}\sum_{\substack{j=1\\ j\neq k}}^{K}\mu_{jl}\left(\sum_{i=1}^{N_j}\lambda_{ji}f_l(\mathbf{s}_i)\right).\end{aligned} \tag{2.12}$$

After taking partial derivatives of (2.12) with respect to variables λ and μ we obtain the following system of linear equations:

$$\begin{aligned}
&\sum_{j=1}^{K}\sum_{i=1}^{N_j}\lambda_{ji}C_{jj'}\left(\mathbf{s}_i-\mathbf{s}_{i'}\right)+\sum_{l=0}^{L}\mu_{j'l}f_l\left(\mathbf{s}_i\right)=C_{j'k}\left(\mathbf{s}_{i'}-\mathbf{s}_0\right),\\
&\quad\forall j'=1,\ldots,K,\ \forall i'=1,\ldots,N_{j'};\\
&\sum_{i=1}^{N_k}\lambda_{ki}f_l(\mathbf{s}_i)=f_l(\mathbf{s}_0)\quad\forall l=0,\ldots,L;\\
&\sum_{i=1}^{N_j}\lambda_{ji}f_l(\mathbf{s}_i)=0\quad j\neq k,\ \forall l=0,\ldots,L.
\end{aligned}\tag{2.13}$$

When $k=1$, system (2.13) can be expressed in matrix form as follows:

$$\begin{bmatrix}
\mathbf{C}_{11} & \mathbf{C}_{12} & \cdots & \mathbf{C}_{1K} & \boldsymbol{F}_1 & \mathbf{0} & \cdots & \mathbf{0}\\
\mathbf{C}_{21} & \mathbf{C}_{22} & \cdots & \mathbf{C}_{2K} & \mathbf{0} & \boldsymbol{F}_2 & \cdots & \mathbf{0}\\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots\\
\mathbf{C}_{K1} & \mathbf{C}_{K2} & \cdots & \mathbf{C}_{KK} & \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{F}_K\\
\boldsymbol{F}_1^T & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0}\\
\mathbf{0} & \boldsymbol{F}_2^T & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0}\\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0}\\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots\\
\mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{F}_K^T & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0}
\end{bmatrix}
\begin{bmatrix}
\boldsymbol{\lambda}_1\\ \boldsymbol{\lambda}_2\\ \vdots\\ \boldsymbol{\lambda}_K\\ \boldsymbol{\mu}_1\\ \boldsymbol{\mu}_2\\ \boldsymbol{\mu}_3\\ \vdots\\ \boldsymbol{\mu}_K
\end{bmatrix}=
\begin{bmatrix}
\boldsymbol{c}_{10}\\ \boldsymbol{c}_{20}\\ \vdots\\ \boldsymbol{c}_{K0}\\ \boldsymbol{f}_{01}\\ \mathbf{0}\\ \mathbf{0}\\ \vdots\\ \mathbf{0}
\end{bmatrix}\tag{2.14}$$

where

$$\left[\boldsymbol{C}_{mn}\right]_{ij}=\mathrm{Cov}\left(\mathcal{X}_{\mathbf{s}_i}^{(m)},\mathcal{X}_{\mathbf{s}_j}^{(n)}\right)=C_{mn}\left(\mathbf{s}_i-\mathbf{s}_j\right),$$

$$\boldsymbol{F}_j=\begin{bmatrix}
f_0\left(\mathbf{s}_1\right) & f_1\left(\mathbf{s}_1\right) & \cdots & f_L\left(\mathbf{s}_1\right)\\
f_0\left(\mathbf{s}_2\right) & f_1\left(\mathbf{s}_2\right) & \cdots & f_L\left(\mathbf{s}_2\right)\\
\vdots & \vdots & \vdots & \vdots\\
f_0\left(\mathbf{s}_{N_j}\right) & f_1\left(\mathbf{s}_{N_j}\right) & \cdots & f_L\left(\mathbf{s}_{N_j}\right)
\end{bmatrix},$$

$$\boldsymbol{\lambda}_j=\begin{bmatrix}\lambda_{j1}\\ \lambda_{j2}\\ \vdots\\ \lambda_{jN_j}\end{bmatrix},\ \boldsymbol{\mu}_j=\begin{bmatrix}\mu_{j0}\\ \mu_{j1}\\ \vdots\\ \mu_{jL}\end{bmatrix},$$

$$\boldsymbol{c}_{j0}=\begin{bmatrix}
C_{jk}\left(\mathbf{s}_1-\mathbf{s}_0\right)\\
C_{jk}\left(\mathbf{s}_2-\mathbf{s}_0\right)\\
\vdots\\
C_{jk}\left(\mathbf{s}_{N_j}-\mathbf{s}_0\right)
\end{bmatrix},\ \boldsymbol{f}_{0j}=\begin{bmatrix}
f_0\left(\mathbf{s}_0\right)\\
f_1\left(\mathbf{s}_0\right)\\
\cdots\\
f_L\left(\mathbf{s}_0\right)
\end{bmatrix}.$$

Solving the linear system with respect to λ's and μ's allow to obtain the desired best linear unbiased predictor $\mathcal{X}_{\mathbf{s}_0}^{(k)\star}=\sum_{j=1}^{K}\sum_{i=1}^{N_j}\lambda_{ji}^{\star}\mathcal{X}_{\mathbf{s}_i}^{(j)}$ with the associated

trace-variance given by

$$\sigma_k^2(\mathbf{s}_0) = C_{kk}(\mathbf{0}) - \sum_{j=1}^{K}\sum_{i=1}^{N_j}\lambda_{ji}C_{kj}(\mathbf{s}_i - \mathbf{s}_0) + \sum_{l=0}^{L}\mu_{kl}f_l(\mathbf{s}_0). \tag{2.15}$$

2.3 Parameters inference

The strategy for parameter inference can be analogous to that performed in conventional multivariate cokriging: functional regression [32] is used to estimate functional drift for each element of the multivariate functional data, e.g. performing ordinary least squares regression. Then, estimates of trace-auto and trace-cross covariances are computed on the estimated functional residuals and admissible covariance structures are fitted with the linear model of coregionalization (LMC, Goovaerts [18]).

Given the assumption of spatial dependence between observations, improvements on drift and residuals estimation can be obtained by a generalized least squares approach. However, the latter is possible only after an estimation of the structure of spatial dependence, so that an iterative algorithm needs to be performed in order to jointly estimate drift and spatial dependence: this procedure is described in Algorithm 1.

Algorithm 1. *Given a realization* $\boldsymbol{x}^{(k)} = (x^{(k)}_{\mathbf{s}_1}, \dots, x^{(k)}_{\mathbf{s}_n})$ *of the k-th component of the non-stationary multivariate functional random field* $\{\boldsymbol{\mathcal{X}}_{\mathbf{s}},\ \mathbf{s} \in D\}$ *representable as in* (2.4)*:*

1. *estimate drift vector* $\boldsymbol{m}$ *through Ordinary Least-Squares method and set* $\hat{\boldsymbol{m}}_{\mathbf{s}} := \hat{\boldsymbol{m}}^{OLS}_{\mathbf{s}}$*;*
2. *compute residual estimate* $\hat{\boldsymbol{\delta}} = (\delta_{\mathbf{s}_1}, \dots, \delta_{\mathbf{s}_n})$ *by difference* $\hat{\boldsymbol{\delta}} = \boldsymbol{x} - \hat{\boldsymbol{m}}$*;*
3. *estimate the trace-semivariogram* $\gamma(\cdot)$ *of the residual process* $\{\delta_{\mathbf{s}}, \mathbf{s} \in D \subseteq \mathbb{R}^d\}$ *from* $\hat{\boldsymbol{\delta}}$*, fit a valid LMC* $\gamma(\cdot, \hat{\boldsymbol{\theta}})$ *and derive the estimate* $\hat{\Sigma}$ *of* Σ*;*
4. *estimate drift vector* $\hat{\boldsymbol{m}}$ *through Generalized Least-Squares method taking as covariance matrix* $\hat{\Sigma}$*;*
5. *repeat 2.-4. until convergence has been reached.*

After drift estimation it is possible to proceed with the estimate of dependence structure in the stationary case, obtained by replacing to the observations the estimated residuals computed with Algorithm 1. In this case auto and cross-covariance estimation can be performed by means of trace-variography approach by Menafoglio et al. [29], which formulate the following estimators:

- trace variogram estimator:

$$\gamma_{k,k}(\boldsymbol{h}) = \frac{1}{2|N(\boldsymbol{h})|}\sum_{(i,j)\in N(\boldsymbol{h})}\left\|\mathcal{X}^{(k)}_{\mathbf{s}_i} - \mathcal{X}^{(k)}_{\mathbf{s}_j}\right\|^2; \tag{2.16}$$

- trace cross-variogram estimator:

$$\gamma_{k,l}(\boldsymbol{h}) = \frac{1}{2|N(\boldsymbol{h})|} \sum_{(i,j)\in N(\boldsymbol{h})} \left\langle \mathcal{X}^{(k)}_{\mathbf{s}_i} - \mathcal{X}^{(k)}_{\mathbf{s}_j}, \mathcal{X}^{(l)}_{\mathbf{s}_i} - \mathcal{X}^{(l)}_{\mathbf{s}_j} \right\rangle; \tag{2.17}$$

where $N(\boldsymbol{h})$ denotes the set of pairs (i, j) approximately separated by a vector $\boldsymbol{h}$, i.e., such that $\mathbf{s}_i - \mathbf{s}_j \sim \boldsymbol{h}$. In practice, inference and fitting over high dimensional input spaces is limited to omni-directional variograms due to the well-known curse of dimensionality [12]. Other estimators are available in literature for cross-variograms estimation (e.g. pseudo trace cross-variogram estimator [20]), the choice of which depends on the assumptions of the statistical model and the design of the experiment under study. In this work we choose the estimator defined in Equation (2.17) given the isotopic nature of data in reservoir experiments, i.e., the set of locations in which the primary variable is measured is a subset of the one in which secondary variables measurements are available.

The process of fitting trace-variogram and trace cross-variogram with a valid LMC, i.e., the estimate of optimal parameters value within a given family of valid models, is usually performed through maximum likelihood or least squares procedures ([10, Sect 2.6]). When dealing with high dimensional input spaces this task can be difficult due to sparsity of available sampled locations. This issue, especially present in cases when data come from computationally expensive computer code, often results in the lack of good fit and predictive capability of geostatistical models. In the following analysis we will consider a cross-validation approach to variogram fitting, in particular we will select a family of valid models and focus on optimal range estimation. We discard sill optimization since it is possible to show that both kriging and cokriging prediction is invariant with respect to the choice of sill, so that we consider this parameter fixed. The procedure we adopt is detailed in Algorithm 2.

Algorithm 2. *Given a realization* $\boldsymbol{x} = (x_{\mathbf{s}_1}, \dots, x_{\mathbf{s}_n})$ *of the multivariate functional random field* $\{\boldsymbol{\mathcal{X}}_{\mathbf{s}},\ \mathbf{s} \in D\}$*, a valid LMC* $\gamma(\cdot, r)$ *and a set of* N_r *candidate range values* $r_1, \dots, r_{N_r}$*:*

1. *split available training data into* K *subsets (*folds*)* $T_1, \dots, T_K$*;*
2. *for each range* r_j*, build* K *cokriging predictors* $\mathcal{X}^{\star(-k)}_{\mathbf{s}}$ *with training data* $\boldsymbol{\mathcal{X}} \setminus T_k$ *and covariance model* $\gamma(\cdot, r_j)$*;*
3. *compute the average cross-validation sum of squared errors*

$$SSE_j = \sum_{k=1}^{K} SSE_j^{(k)} = \frac{1}{K} \sum_{K=1}^{K} \sum_{i\in T_k} \left\| \mathcal{X}^{\star(-k)}_{\mathbf{s}_i} - \boldsymbol{x}_{\mathbf{s}_i} \right\|^2 \tag{2.18}$$

4. *choose* $j^\star$ *such that* $j^\star = \arg\min_{j=1,\dots,N_r} SSE_j$*.*

Chapter 3

Synthetic reservoir models generation

The aim of this Chapter is to present the set of synthetic reservoir models used in testing our approach. All scenarios are derived from the model *SPE1* of Society of Petroleum Engineers (SPE) project for comparative solutions and benchmarks. For each scenario we will detail field properties, simulation scheme and the upscaling procedure performed to obtain coarser realization necessary to following testing. All the simulations performed in this work exploit the reservoir simulator *OPM Flow* [33], a fully-implicit black-oil simulator. Upscaling is performed with the `upscaling` module of *MRST* reservoir simulator [25], a set of routines for single and two-phase upscaling. In order to solve the fluid flow problem, several input data must be set, distinguished into two type: external data and field data. Some examples of external data are the injection and production well positions, which can be varied for feasibility studies over new reservoirs or to optimize the oil production in the existing ones. Field data are quantities such as porosity and permeability, which characterize the reservoir.

3.1 SPE1 model

The SPE Comparative Solution Project (`www.spe.org/web/csp`) is a comparative solution projects organised by the Society of Petroleum Engineers. The purpose of the projects has been to provide benchmark datasets which can be used to compare the performance of different simulators or algorithms. In particular, SPE1 reservoir model is an example of black oil reservoir simulation derived by a comparison study in Odeh [31]. It is also part of the open access datasets of the OPM project (available at `https://github.com/OPM/opm-data.git`). The structure is a regular Cartesian grid of $100 \times 100 \times 3$

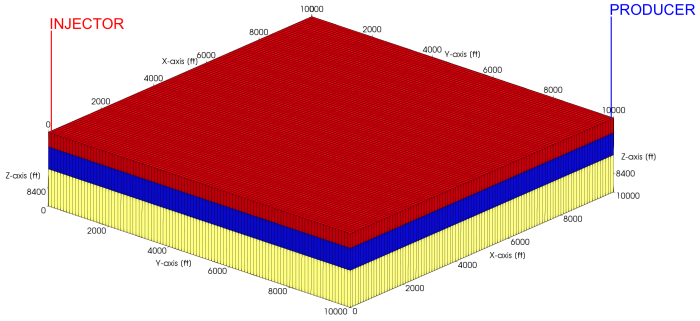

Figure 3.1: SPE1 model grid coloured by layers quote, dimensions are expressed in field units (feet).

Property	Value
Initial reservoir pressure (psia @8400ft)	4800
Gas injection rate (MMSCF/D)	100
Maximum oil production rate (STB/D)	20000
Minumum oil rate (STB/D)	1000
Minimum flowing bottomhole pressure (psi)	1000
Wellbore radius (ft)	0.25
Skin (–)	0
Capillary pressure (psia)	0
Reservoir temperature (°F)	200
Gas specific gravity (–)	0.792

Table 3.1: Reservoir properties of SPE1 model.

hexaedral elements (cells) of dimension 100 ft along x and y-axis with variable thickness (respectively 20, 30 and 50 ft) along the z-axis. The reservoir is provided with two wells, as it is portrayed in Figure 3.1:

- one gas injector located at Grid Point (1,1) on Layer 1,
- one oil producer at Grid Point (100,100) on Layer 3.

Data related to initial conditions of the reservoir and production constraints are listed in Table 3.1. The model is endowed with two phases, oil and gas, with the presence of dissolved gas into oil. The Pressure-Volume-Temperature (PVT) functions for all phases are listed in Table 3.2, 3.3, 3.4, 3.5 and 3.6, while Table 3.7 shows oil and gas relative permeability (respectively k_{ro} and k_{rg}) as functions of gas saturation level s_g. Since there are only two phases present in the simulation, water relative permeability k_{rw} is set to zero at any saturation level. All the aforementioned data are taken from Odeh [31].

Reservoir pressure (psia)	FVF (RB/MSCF)	Viscosity (cp)	Density (lbm/ft^3)	Pseudo Gas Potential (psia^2/cp)
14.7	166.666	0.008000	0.0647	0.0
264.7	12.093	0.009600	0.8916	0.777916e+07
514.7	6.274	0.011200	1.7185	0.267580e+08
1014.7	3.197	0.014000	3.3727	0.875262e+08
2014.7	1.614	0.018900	6.6806	0.270709e+09
2514.7	1.294	0.020800	8.3326	0.386910e+09
3014.7	1.080	0.022800	9.9837	0.516118e+09
4014.7	0.811	0.026800	13.2952	0.803963e+09
5014.7	0.649	0.030900	16.6139	0.112256e+10
9014.7	0.386	0.047000	27.9483	0.251845e+10

Table 3.2: Gas PVT Functions.

Reservoir pressure (psia)	FVF (RB/STB)	Viscosity (cp)	Density (lbm/ft^3)	Solution GOR (SCF/STB)
14.7	1.0620	1.0400	46.244	1.0
264.7	1.1500	0.9750	43.544	90.5
514.7	1.2070	0.9100	42.287	180.0
1014.7	1.2950	0.8300	41.004	371.0
2014.7	1.4350	0.6950	38.995	636.0
2514.7	1.5000	0.6410	38.304	775.0
3014.7	1.5650	0.5940	37.781	930.0
4014.7	1.6950	0.5100	37.046	1270.0
5014.7	1.8270	0.4490	36.424	1618.0
9014.7	2.3570	0.2030	34.482	2984.0

Table 3.3: Saturated Oil PVT Functions.

Reservoir pressure (psia)	FVF (RB/bbl)	Viscosity (cp)	Density (lbm/ft^3)	Gas/Water Ratio (SCF/bbl)
14.7	1.0410	0.3100	62.238	0.0
264.7	1.0403	0.3100	62.283	0.0
514.7	1.0395	0.3100	62.328	0.0
1014.7	1.0380	0.3100	62.418	0.0
2014.7	1.0350	0.3100	62.599	0.0
2514.7	1.0335	0.3100	62.690	0.0
3014.7	1.0320	0.3100	62.781	0.0
4014.7	1.0290	0.3100	62.964	0.0
5014.7	1.0258	0.3100	63.160	0.0
9014.7	1.0130	0.3100	63.959	0.0

Table 3.4: Saturated Water PVT Functions.

Reservoir pressure (psia)	FVF (RB/STB)	Viscosity (cp)	Density (lbm/ft^3)
4014.7	1.6950	0.5100	37.046
9014.7	1.5790	0.7400	39.768

Table 3.5: Undersaturated Oil PVT Functions.

Reservoir pressure (psia)	FVF (RB/bbl)	Viscosity (cp)	Density (lbm/ft^3)
4014.7	1.0290	0.3100	62.964
9014.7	1.0130	0.3100	63.959

Table 3.6: Undersaturated Water PVT Functions.

s_g	k_{rg}	k_{ro}
0.000	0.0000	1.0000
0.001	0.0000	1.0000
0.020	0.0000	0.9970
0.050	0.0050	0.9800
0.120	0.0250	0.7000
0.200	0.0750	0.3500
0.250	0.0125	0.2000
0.300	0.1900	0.0900
0.400	0.4100	0.0210
0.450	0.6000	0.0100
0.500	0.7200	0.0010
0.600	0.8700	0.0001
0.700	0.9400	0.0000
0.850	0.9800	0.0000
1.000	1.0000	0.0000

Table 3.7: Relative permeability data for Oil and Gas phases.

3.2 Rock characterization

Rock properties represent the core of our synthetic model, since we focus on establishing the relation between uncertainty over field parameters and the output of the simulation. Since porosity and permeability are the main properties characterizing reservoir behaviour, we developed a procedure to produce random fields of these two properties over our domain which represent admissible conditions for a realistic reservoir.

3.2.1 Porosity field

We generate an instance of a random field for porosity with the `gstat` package of R software, with given mean and variance, assuming a structure of spatial dependence over the domain. This approach allows to reproduce the general behaviour of rock, in which changes in porosity are gradual, instead of assuming a pure random noise process which would have been non-compliant with realistic conditions. Since the purpose of our analysis is the production of several instances of the model through upscaling procedures, we consider for porosity simulation a covariance model which is linear near the origin, such that upscaling will be effective while respecting the overall structure of the model. We choose an exponential covariance model, characterized by the following semivariogram function:

$$\gamma_{exp}(\boldsymbol{h}, S, r) = \begin{cases} 0 & \boldsymbol{h} = \mathbf{0}, \\ S\left(1 - \exp\left(\dfrac{\|\boldsymbol{h}\|}{r}\right)\right) & \boldsymbol{h} \neq \mathbf{0}, \end{cases} \tag{3.1}$$

in which $\boldsymbol{h} \in \mathbb{R}^3$ is the distance between to locations within the domain, S represent the variance of the process at great distance (also known as *sill*) and r is the *range* of the covariance model, i.e., a measure of how fast the spatial dependence scales with distance. We consider two locations being independent realization of the process when their distance is approximately $3r$, value often denoted as *effective range* (see Figure 3.2 for an example). In order to build a realistic porosity field we consider a geometric anisotropy for the covariance model. Geometric anisotropy entails that variance between different locations is a function of both magnitude and direction of the distance vector $\boldsymbol{h}$. This effect is obtained performing a linear transformation of $\boldsymbol{h}$, namely, taking a semivariogram function of the form

$$\tilde{\gamma}(\boldsymbol{h}) = \gamma(\|A\boldsymbol{h}\|), \quad \boldsymbol{h} \in \mathbb{R}^3, \tag{3.2}$$

where A is a 3×3 symmetric matrix whose eigenvectors represent the main directions of variation for the process, while the eigenvalues scale the sill parameters along their associated directions. The choice of an anisotropic random field is meant also to affect the upscaling procedure, which operates along

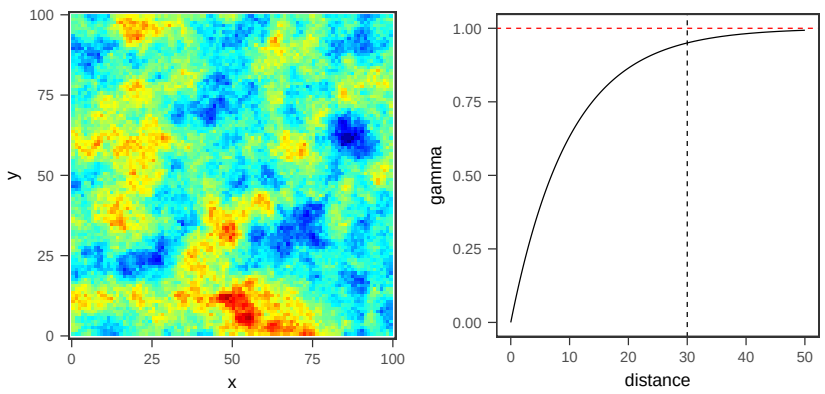

Figure 3.2: Left: Realization of exponential random field with $S = 1$ and $3r = 30$; Right: theoretical variogram function with sill (red dashed line) and effective range (black dashed line).

Cartesian axes, in order to better simulate the issue of upscaling a fine-scale field into a coarser one. For the simulation of the porosity fields we choose the anisotropy matrix A written as

$$A = \Lambda D \Lambda^T \tag{3.3}$$

where Λ is the rotation matrix

$$\Lambda = \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \boldsymbol{v}_3 \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{5\pi}{12}\right) & -\sin\left(\frac{5\pi}{12}\right) & 0 \\ \sin\left(\frac{5\pi}{12}\right) & \cos\left(\frac{5\pi}{12}\right) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

which perform a counter-clockwise rotation of the Cartesian axes of 75 degrees with respect to the positive x direction on the x-y plane, while D is the diagonal matrix

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.12 \end{bmatrix}$$

which perform a rescaling of the sill along second and third main direction. Figure 3.3 show graphically the transformation we operate on Cartesian axes.

3.2.2 Permeability field

The generation of permeability fields is usually obtained taking permeability values as function of porosity within the same cell. We choose to exploit a relation of the form

$$\log(K) = a\left(\Phi(\boldsymbol{x}) + \varepsilon(\boldsymbol{x})\right) + b \tag{3.4}$$

where K is permeability, Φ porosity and ε an independent zero-mean random error with equal covariance structure. Parameters a and b are deterministic

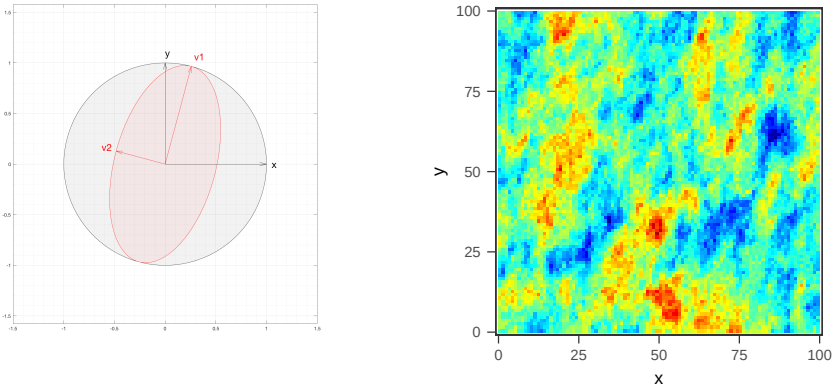

Figure 3.3: Left: Effect over Cartesian axes (black) of the linear transformation given by anisotropy matrix A (red); Right: realization of exponential anisotropic random field generated with matrix A.

and can be tuned to modify the resulting permeability field. The condition we choose to impose over these two parameters are the following:

- the expected value of the resulting permeability field must match some given value m, so that if $\Phi \sim N(\mu_\Phi, \sigma_\Phi^2)$ and $\varepsilon \sim N(0, \sigma_\varepsilon^2)$ with $\Phi \perp \varepsilon$ then
$$\log(K) \sim N\left(a\mu_\Phi + b, a^2\left(\sigma_\Phi^2 + \sigma_\varepsilon^2\right)\right) \tag{3.5}$$
therefore it must be
$$\mathbb{E}[K] = \exp\left(a\mu_\Phi + b + a^2\left(\frac{\sigma_\Phi^2 + \sigma_\varepsilon^2}{2}\right)\right) = m; \tag{3.6}$$

- the dispersion around the expected value μ_K is controlled by Chebyshev inequality
$$\mathbb{P}\left(\frac{|K - \mu_K|}{\sigma_K} \geq C\right) \leq \frac{1}{C^2}, \quad C \geq 2 \tag{3.7}$$
where σ_K is the standard deviation of permeability, so that fixing Chebyshev bound at 95% ($C \cong 4,5$) it must be
$$\sigma_K^2 = \exp\left(2a\mu_\Phi + 2b + a^2\left(\sigma_\Phi^2 + \sigma_\varepsilon^2\right)\right)\left(\exp\left(a^2\left(\sigma_\Phi^2 + \sigma_\varepsilon^2\right)\right) - 1\right) = \left(\frac{s}{C}\right)^2 \tag{3.8}$$
for some given value s.

Solving for the unknowns a and b allow to obtain the desired permeability field, which will be correlated with porosity depending on the ratio between σ_Φ^2 and σ_ε^2. Finally, since we consider permeability as an isotropic tensor, namely
$$K = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{bmatrix} \tag{3.9}$$
we will generate only k_x permeability values, assuming $k_y = C_{xy}k_x$ and $k_z = C_{xz}k_x$ for some constants $C_{xy}, C_{xz} \in (0,1]$.

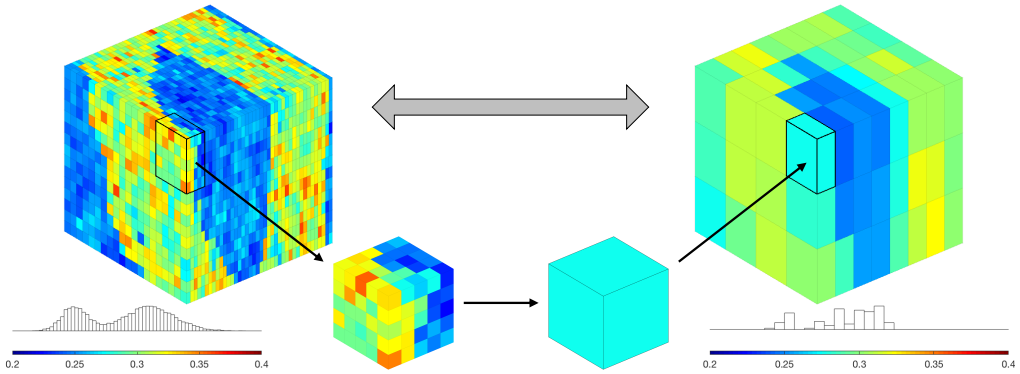

Figure 3.4: An example of local upcaling of porosity between fine level (left) and coarse one (right), with distribution histograms (bottom).

3.3 Parameters upscaling

Given a Cartesian fine grid with $n_x \times n_y \times n_z$ cells and a Cartesian coarse one with $n_x^c \times n_y^c \times n_z^c$, we define the directional scale factors as

$$s_x = \frac{n_x}{n_x^c}; \quad s_y = \frac{n_y}{n_y^c}; \quad s_z = \frac{n_z}{n_z^c}. \tag{3.10}$$

Since our model has a low $n_z = 3$, we will consider upscaled realizations with fixed scale factor $s_z = 1$. We choose to build two different coarse realizations of our model, namely:

- one with scale factors $s_x = 2, s_y = 2, s_z = 1$ denoted as *COARSE50*, with $50 \times 50 \times 3$ cells;
- one with scale factors $s_x = 4, s_y = 4, s_z = 1$ denoted as *COARSE25*, with $25 \times 25 \times 3$ cells.

These two realizations, coupled with the original model (hereafter denoted with name *FINE*) will be the reservoir realizations under study in the following analyses.

In order to produce the coarser realizations of SPE1 model we perform the upscaling of porosity and absolute permeability with a single-phase approach (as mentioned in Section 1.3.2). The core of the procedure is to define a relation between different levels of the grid, i.e., a map linking each coarse block to the fine ones lying within. Since we opt for even scale factors over x and y-axis, fine and coarse grids are matching in both cases, that is, cells faces of the coarse grid are aligned with fine grid ones. Moreover, the resulting partition is exactly uniform, having respectively 4 and 16 fine-scale cells per target coarse block. These two aspects are of primary importance when performing single-phase local upscaling, since the procedure aims at resolving the equivalent volume flow problem between two levels. An odd number of cells per coarse block or non-matching faces can therefore result in distorted upscaled fields. One of the

effects worth mentioning when performing local upscaling of field properties, as shown in Figure 3.4, is the reduction of the span and the tendency of values to cluster around the median of the fine-scale distribution.

3.4 Simulation outputs

For each level of the model we run the simulation over ten years with a time step of one month, recording at each time step the production curves recorded at the production well. Three main output curves are of interest in our model:

- Gas Oil Ratio over time (GOR), the ratio of the volume of gas that comes out of solution to the volume of oil at standard conditions, related to the initial pressure of the reservoir with respect to the *bubble point* (pressure at which dissolved gas liberates from oil); it is a dimensionless quantity in metric units, while in field units it is measured in Standard Cubic Feet per Stock Tank Barrels (SCF/STB);
- Oil Production Rate over time (OPR), measured in Stock Tank Barrels per day (STB/D), is the ratio at which oil is produced; it is a primary index of efficiency of a production plant, and within simulations it is usually constrained to be less or at most equal to the maximum plant production capacity;
- Oil Production Total over time (OPT), measured in STB, is the cumulative production of oil; most of the insights obtained from reservoir simulation exploit OPT curves to compute economic indexes (such as Net Present Value) for decision making.

Since our model is provided with only one production well the overall field behaviour coincides with the one of the well, so that from now on we will refer at each output curve with the prefix "F-" standing for *Field*.

3.5 Synthetic reservoir models

In order to test the functional cokriging approach detailed in Chapter 2 as a surrogate model for prediction we propose two synthetic models built on the SPE1 reservoir model. For each model we generate porosity and permeability layers with the procedure described in Section 3.2.1 and 3.2.2, build coarse instances through upscaling, establish a set of uncertain parameters within the simulation and produce the aforementioned functional outputs for each configuration over the input space. After the generation of the output curves, which will be the subject of spatial prediction, we test the capability of our method with respect to the Universal Trace Kriging approach by Menafoglio

et al. [29]. Universal Trace Kriging is a functional univariate method for spatial prediction of curves, which can be retrieved from our approach imposing only one level of output, i.e., it represents the situation in which no secondary data is available together with the target response.

3.5.1 Model A

The first model we propose for evaluation is built with a single instance of porosity field with given mean, sill and range, upon which permeability is computed separately for each layer of the model. This model is intended to reproduce the behaviour of a reservoir with homogeneous porosity layers. Table 3.8 summarizes the input parameters chosen for the initialization of this model. The resulting fields of porosity are displayed in Figure 3.5, while Figure 3.6 shows the distribution of permeability over the grid. After the generation of the fine scale fields we perform the upscaling procedure in order to obtain the COARSE50 and COARSE25 realizations of the reservoir model, whose first layer are compared to the corresponding FINE one in Figure 3.7 and 3.9. Also, Figure 3.8 and 3.10 show the effect of upscaling on the overall distribution of porosity and permeability: as expected, values tend to concentrate around modes of the distribution.

Parameter			Value
Φ (frac)		range	400.000
		sill	5.0E-4
		mean	0.225
k_x (md)	Layer 1	mean	500.000
		std. dev.	160.908
	Layer 2	mean	50.000
		std. dev.	17.200
	Layer 3	mean	200.000
		std. dev.	42.779
Scale factor k_x/k_y			1.000
Scale factor k_x/k_z			10.000
Rock compressibility (psi^{-1})			3.0E-6

Table 3.8: Input parameters for Model A reservoir.

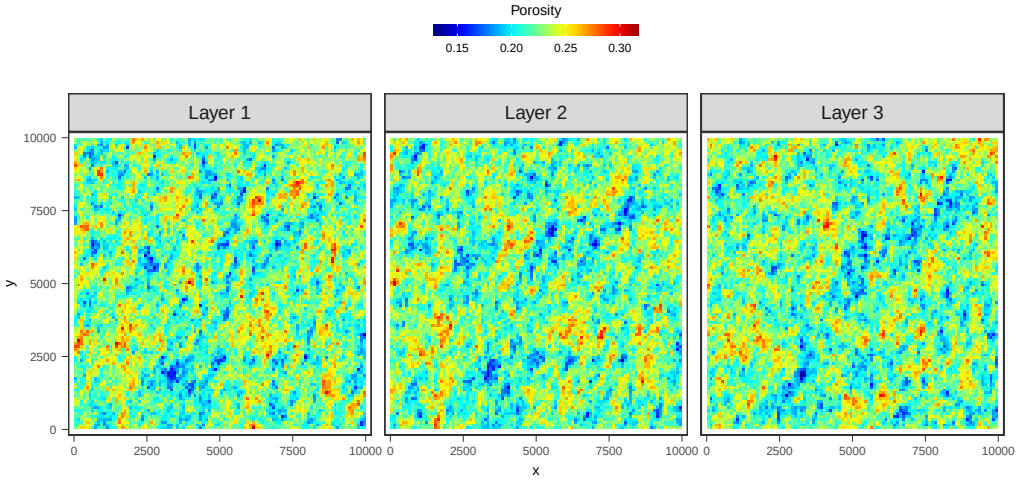

Figure 3.5: Porosity field for Model A, per layer.

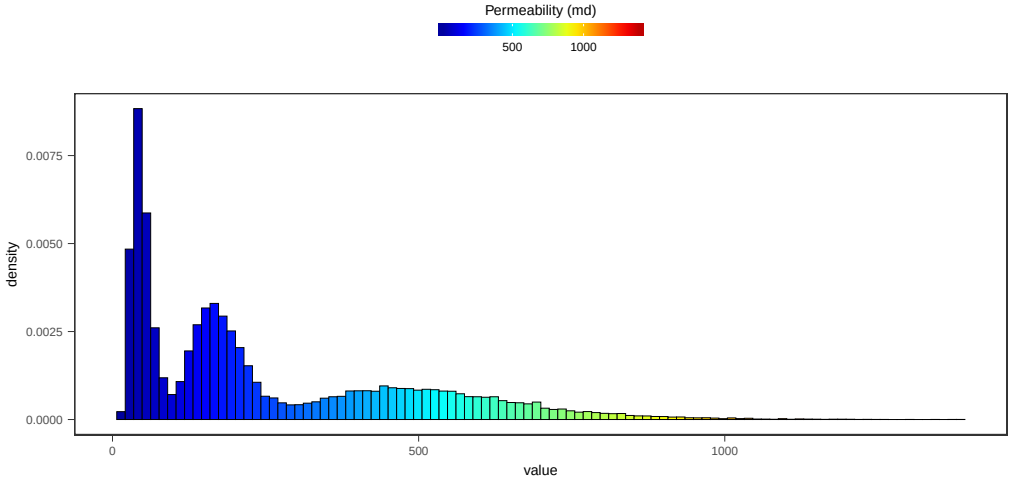

Figure 3.6: Histogram of permeability field for Model A.

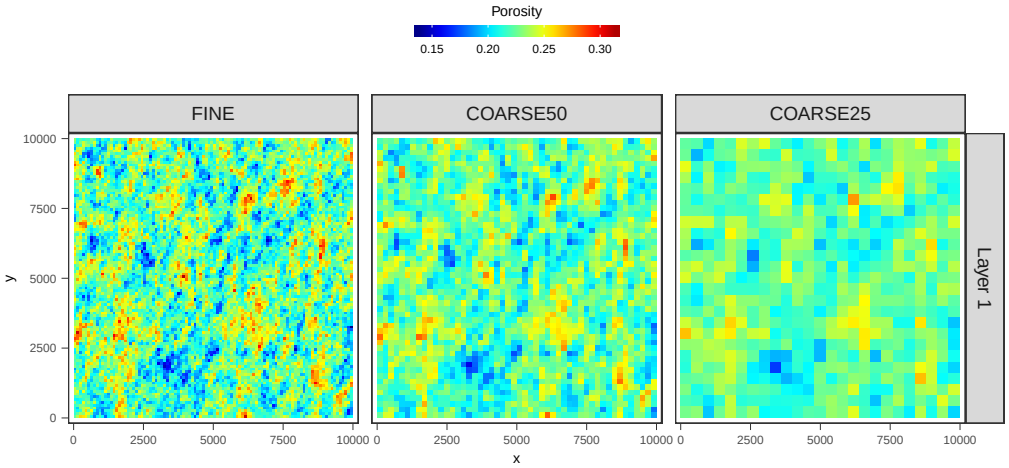

Figure 3.7: Comparison of Model A porosity in Layer 1 for three levels of grid refinement, from FINE (left) to COARSE25 (right).

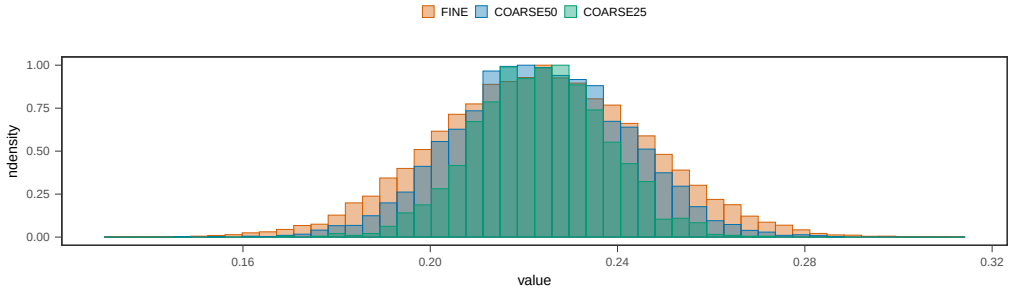

Figure 3.8: Histogram of Model A porosity distribution for three levels of grid refinement.

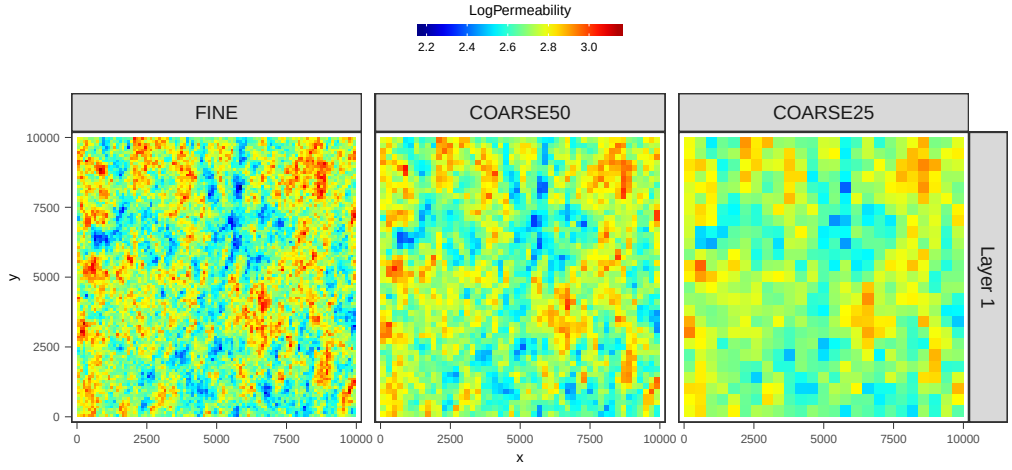

Figure 3.9: Comparison of Model A log-permeability (x-direction) in Layer 1 for three levels of grid refinement, from FINE (left) to COARSE25 (right).

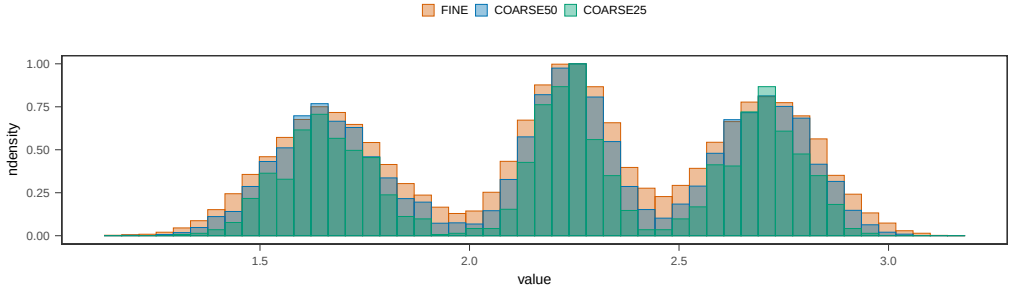

Figure 3.10: Histogram of Model A log-permeability (x-direction) distribution for three levels of grid refinement.

Parameter			Value
Φ (frac)		range	300.000
	Layer 1	sill	2.0E-4
		mean	0.050
	Layer 2	sill	2.5E-4
		mean	0.150
	Layer 3	sill	3.0E-4
		mean	0.250
K_x (md)	Layer 1	mean	500.000
		std. dev.	160.908
	Layer 2	mean	50.000
		std. dev.	17.200
	Layer 3	mean	200.000
		std. dev.	42.779
Scale factor k_x/k_y			1.000
Scale factor k_x/k_z			10.000
Rock compressibility (psi^{-1})			3.0E-6

Table 3.9: Input parameters for Model B reservoir.

3.5.2 Model B

The second model we propose for evaluation is built computing a single instance of random field with given range, upon which both porosity and permeability are computed separately for each layer of the model. The aim is to reproduce a reservoir with heterogeneous layers, whose behaviour will be closer to realistic reservoirs one. Table 3.9 summarizes the input parameters chosen for the initialization of this model. The resulting fields of porosity are displayed in Figure 3.11, while Figure 3.12 shows the distribution of permeability over the grid. Similarly with Model A, after the generation of the fine scale fields we perform the upscaling procedure in order to obtain the COARSE50 and COARSE25 realizations of the reservoir model. Figure 3.13 and 3.15 show the comparison of Layer 1 properties for the three levels. As for Model A, upscaling properties affects the overall distributions with a concentrating effect, as it can be seen in Figure 3.14 and 3.16. Being a more realistic realization of reservoir, this latter model will be object of two numerical experiments which will differ in the chosen set of uncertain parameters.

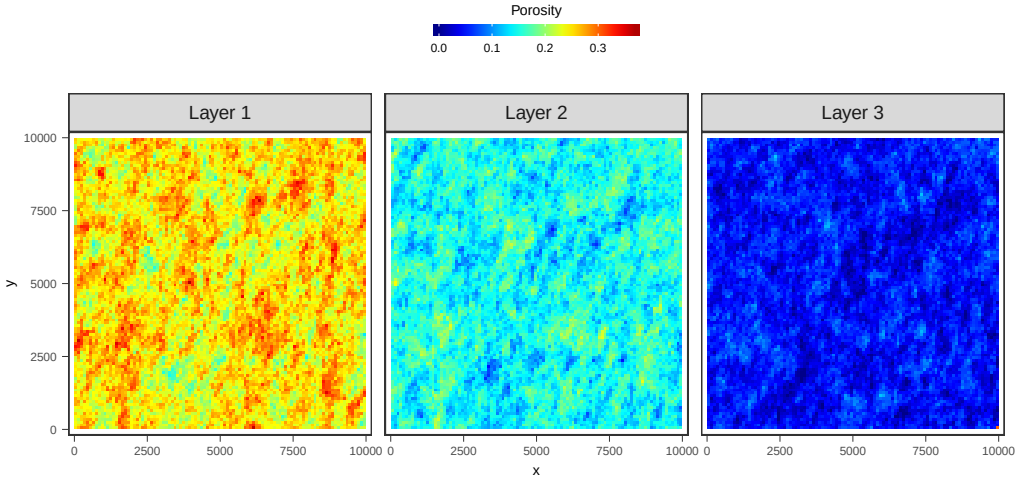

Figure 3.11: Porosity field for Test B, per layer.

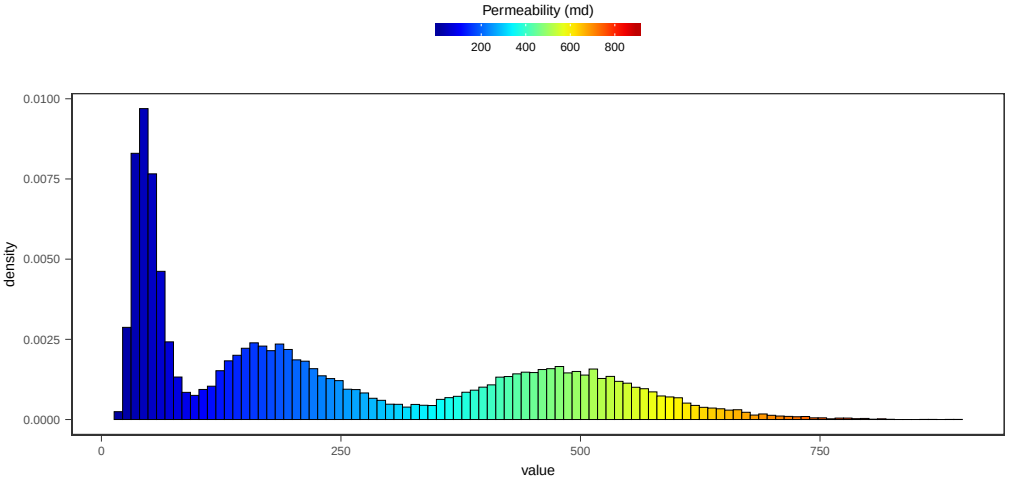

Figure 3.12: Histogram of permeability field for Test B.

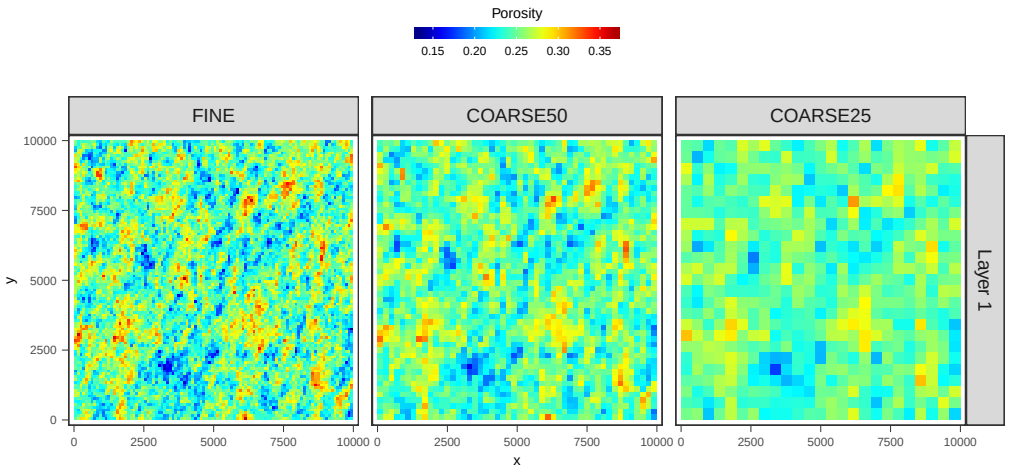

Figure 3.13: Comparison of Model B porosity in Layer 1 for three levels of grid refinement, from FINE (left) to COARSE25 (right).

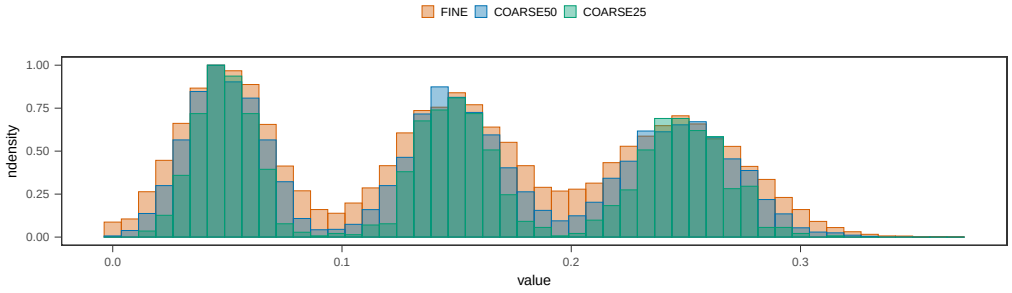

Figure 3.14: Histogram of Model B porosity distribution for three levels of grid refinement.

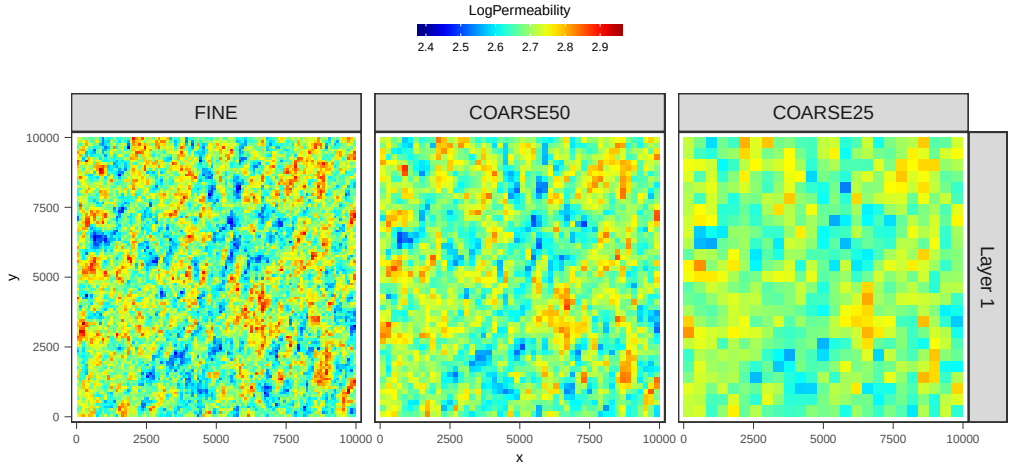

Figure 3.15: Comparison of Model B log-permeability (x-direction) in Layer 1 for three levels of grid refinement, from FINE (left) to COARSE25 (right).

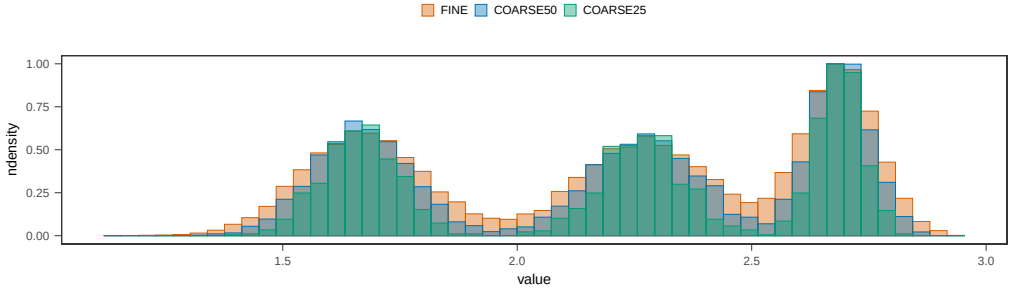

Figure 3.16: Histogram of Model B log-permeability (x-direction) distribution for three levels of grid refinement.

Chapter 4

Numerical results

The aim of this Chapter is to present the case studies we have developed in order to validate our approach. We test the Universal Trace-Cokriging predictor with three experiments, varying either the reservoir model under study or the characterization of the uncertainty over the simulation. We detail the construction of the dataset for each test: a description of the input space, the sampling strategy and the result of the numerical simulation. All numerical evaluations of reservoir models are performed on the HPC facilities of the Laboratory for Modeling and Scientific Computing MOX (`https://mox.polimi.it`). The construction of our predictor is performed with the R package `fdagstat` by Grujic et al., which extends the routines of `gstat` package to the functional framework. In order to validate our approach we develop an extensive comparison with the Universal Trace-Kriging method by Menafoglio et al. [29]. This latter approach does not account for secondary data, and represent the univariate version of the technique described in Chapter 2 (i.e. setting $K = 1$). It is thus fitted only on the fine-scale responses, i.e., without considering the low fidelity model.

4.1 Test 1

The first experiment we propose is realized with Model A and two uncertain parameters over the simulation. In order to choose the actual uncertainties we follow a common approach in literature (see e.g. Bottazzi and Della Rossa [5] and Grujic et al. [20]), considering constant multipliers over porosity and vertical permeability (z-axis) for all layers, while keeping fixed horizontal permeability along x and y directions. These two factors will be hereafter referred as *POROm* and *PERMZm*. Table 4.1 summarizes range and distribution of the uncertain parameters, which we set accordingly to previous works in this research area. The difference in range is due to the aforementioned exponential relationship between porosity and permeability, so that the latter need a wider

Layer	Parameter	Min	Max	Distribution
all	POROm (–)	0.8	1.2	Uniform
all	PERMZm (–)	0.5	2.0	Uniform

Table 4.1: Uncertain parameters for Test 1.

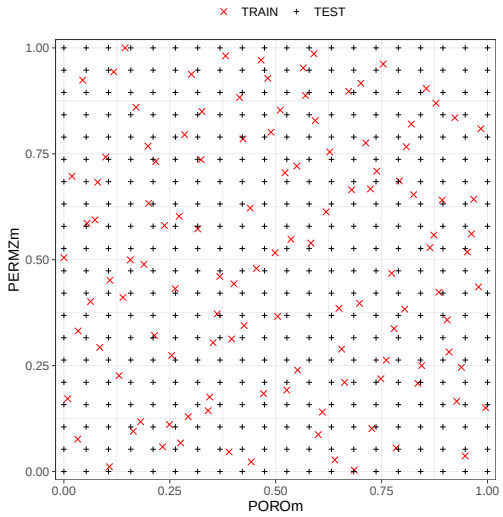

Figure 4.1: Input configurations for Test 1, train (red) and test (black) set; values are rescaled within the unit square.

range of variability in order to produce effective changes in the output. After establishing the uncertainties within the simulation, we generate the input configurations of the reservoir model in order to produce the functional outputs for the analysis. We build two distinct sets of data from the two dimensional input space given by the uncertain parameters (see Figure 4.1):

- a train set of 120 configurations, generated with a Latin Hypercube Sampling (LHS, McKay et al. [28]) approach and evaluated with all levels of flow simulation (FINE, COARSE50 and COARSE25);
- a test set of 400 configurations on a regular grid over the input space, evaluated only with FINE level flow simulation.

Table 4.2 summarizes the computational effort in simulating a single configuration at each fidelity level. These estimates, although computed in an uncontrolled environment and therefore not precise, suggest that the ability to incorporate lower fidelity outputs in place of extending the number of high fidelity ones could improve drastically the efficiency of the whole procedure.

The functional outputs of the numerical simulations for the highest fidelity level are portrayed in Figure 4.2, while Figure 4.3 shows the comparison of the outputs for a single configuration with respect to fidelity levels, which appear to be somewhat similar. Graphical inspection also reveals that both FGOR and FOPR curves are bounded functions, so that embedding them in the L^2

Fidelity Level	Min	p0.25	p0.50	p0.75	Max	mean
FINE	149.664	165.169	174.304	181.983	198.059	173.680
COARSE50	18.469	20.895	22.086	23.825	33.833	22.373
COARSE25	3.780	4.801	5.075	5.396	6.258	5.052

Table 4.2: Summary of computation time for Test 1; values are expressed in seconds.

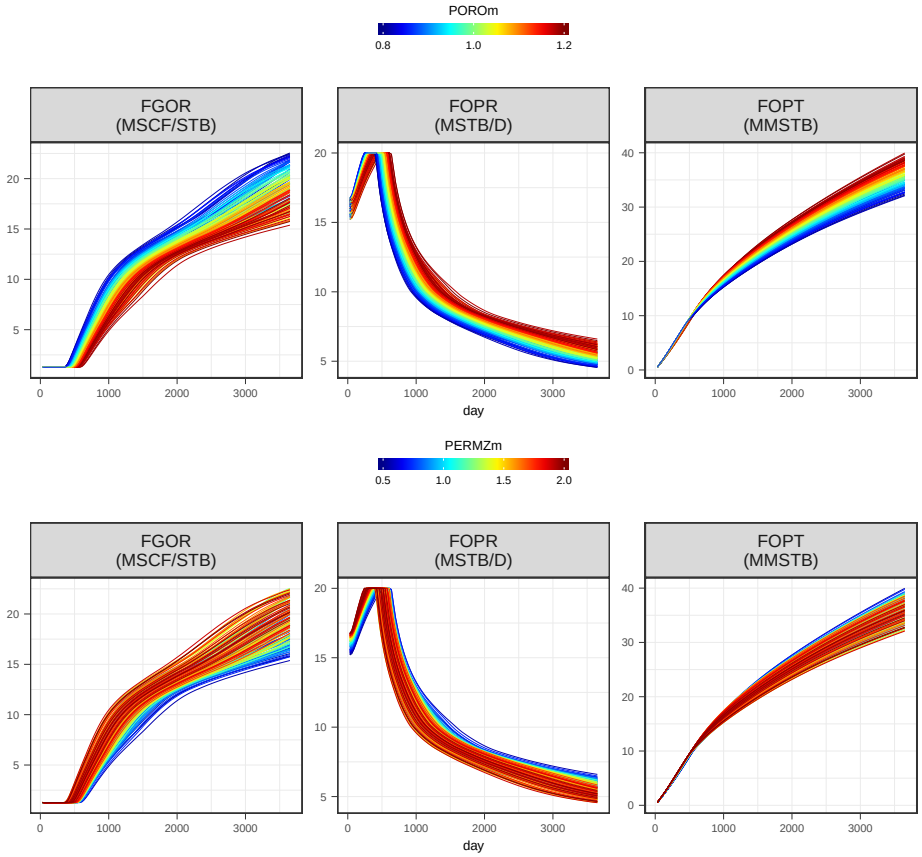

Figure 4.2: Functional outputs of Test 1 for FINE level coloured by parameters POROm (top) and PERMZm (bottom).

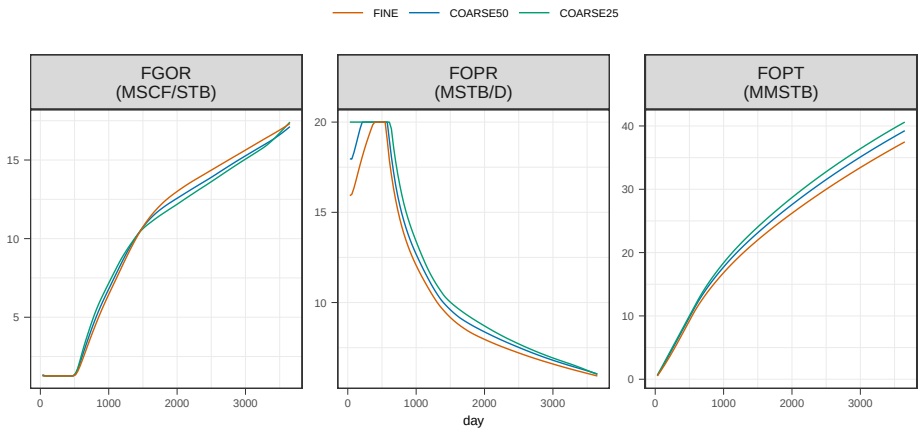

Figure 4.3: Comparison of functional outputs with respect to fidelity level for Test 1.

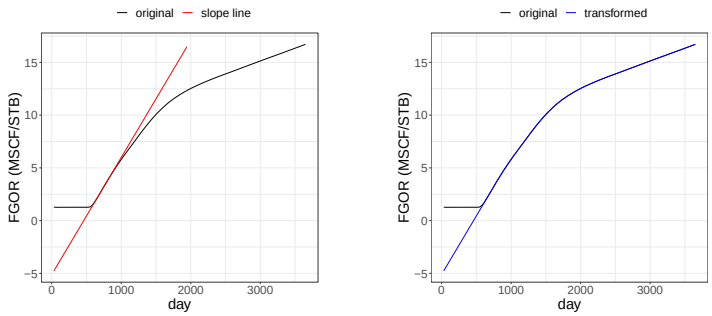

Figure 4.4: FGOR curve transformation procedure. Left: Original curve with straight line fitted through the early non-flat response. Right: The resulting curve.

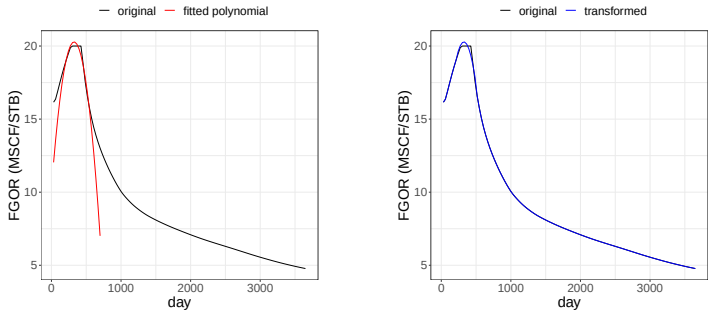

Figure 4.5: FOPR curve transformation procedure. Left: Original curve with second-order polynomial fitted on pre and post-plateau values. Right: The resulting curve.

space of square-integrable functions is not appropriate for our purposes. This is due to the fact that L^2 geometry is only suitable for unconstrained data presenting shifts in amplitude, while constraints within data entail a shift in phase. Following the approach of Grujic et al. [20] we propose the following ad-hoc procedures for the transformation of constrained functions:

- for FGOR curves, identify the time step whose derivative is non-zero, fit a simple regression model to early post-flat values and substitute the initial flat output with the regression solution;

- for FOPR curves, substitute the region containing constant values with a second-order polynomial fitted through regression of pre and post-plateau values.

These two procedures are visually explained respectively in Figure 4.4 and Figure 4.5.

The scheme we adopt to test our method is to compare the results of the prediction on test set varying the amount of fine-scale outputs in the train set. The procedure is the following:

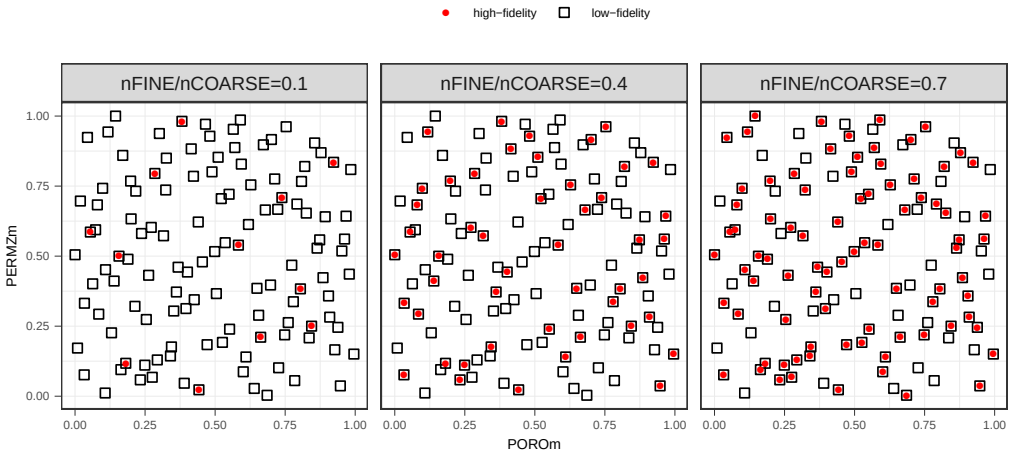

Figure 4.6: Left to right: Experimental designs with increasing high-fidelity evaluations (red dots) among low-fidelity (black squares) ones; *nFINE/nCOARSE* is the proportion of FINE evaluations with respect to COARSE ones.

1. choose the number of input hi-fi configurations for the construction of the predictor;
2. select the actual configurations over the whole train set with a Conditioned Latin Hypercube Sampling approach;
3. compute the Universal Trace-Kriging predictor with the selected subset;
4. extend the FINE subset of curves with the COARSE ones;
5. compute the Universal Trace-Cokriging predictor with the extended set;

In order to preserve coherence between training sets with different amount of hi-fi curves, we select each set of hi-fi curves such that it is a superset of the previous one. Figure 4.6 show three different realizations of the experimental design as the amount of high-fidelity outputs increases, where our selection criterion is clearly depicted: each red point in one plot is present in the right-adjacent one.

Two preliminary steps are necessary to construct both kriging and cokriging predictors, namely:

- the selection of a suitable form for the drift component of the functional process, usually a complete polynomial of given degree in the input variables;
- the choice of a fitting method for LMC parameters estimation from the auto and cross trace-variograms estimates, as discussed in Section 2.3.

The first step is usually data-driven, in that the separation of drift term is needed to recover a stationary residual term upon which the variogram es-

timate could be performed. Step-wise procedures to select the optimal drift term for a realization of non-stationary stochastic processes are available in literature (see e.g. Menafoglio et al. [29]), however in our analysis we decide to consider only first and second degree complete polynomials. This choice resides in the need of preserving as much information as possible within the available data, and to allow the drift parameters' estimation process to be effective: as the number of regression parameters exceeds the number of observations, the resulting least-square matrix becomes non-invertible. For what concerns the second step, we decide to test both least-squares and cross-validation methods for LMC parameter estimation, since there is no a-priori evaluation technique to select the best approach.

The two resulting statistical models are then used to predict the test set outputs and summarize the predictions by computing the sum of squared errors (SSE) of each prediction, that is:

$$SSE_i = \left\| \mathcal{X}_i^{(1)} - \mathcal{X}_i^{\star(1)} \right\|_{L^2}^2 ; \tag{4.1}$$

In order to better appreciate the magnitude of the error all SSE's are normalized by by the average squared norm of the entire test set (400 simulations):

$$SSE_i^n = \frac{SSE_i}{\frac{1}{400}\sum_{i=1}^{400} \left\| \mathcal{X}_i^{(1)} - \mu^{(1)} \right\|_{L^2}^2} \tag{4.2}$$

where $\mu^{(1)}$ is the mean of the test set. When dealing with transformed data, our choice is to compute the prediction SSE's with respect to the transformed unconstrained curves.

For each one of the four combinations of the aforementioned preliminary steps, we test the predictive capability of our method for the three output curves, considering either COARSE50 or COARSE25 as low-fidelity output. Therefore there are 24 different combinations for the test, distinguished by drift term degree, fitting method, target output curve and low-fi level. However, due to instability of the numerical routines within the process of variogram fitting and for solving the cokriging system, only a subset of these test are conclusive. In particular, only the combinations involving least-squares (LS) allow to actually compute the desired predictors, while using cross-validation results in instability while inverting the cokriging variance matrix. Also, the choice of second-order drift produces residuals with too much noise resulting in lack of convergence for the least-squares variogram fitting procedure. As a consequence we focus our interest in comparing the predictors built with first-order drift and least-squares, whose performances are depicted in terms of normalized SSE's distribution in Figure 4.7 (where secondary variable is

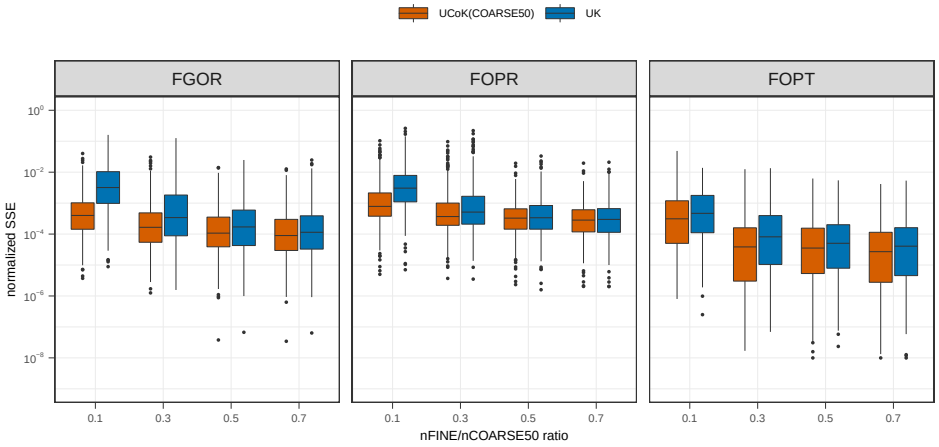

Figure 4.7: Comparison of Test 1 normalized SSE's distribution for increasing ratio of hi-fi response available; predictors are computed with LS variogram fitting, first-order polynomial drift and COARSE50 output as secondary variables.

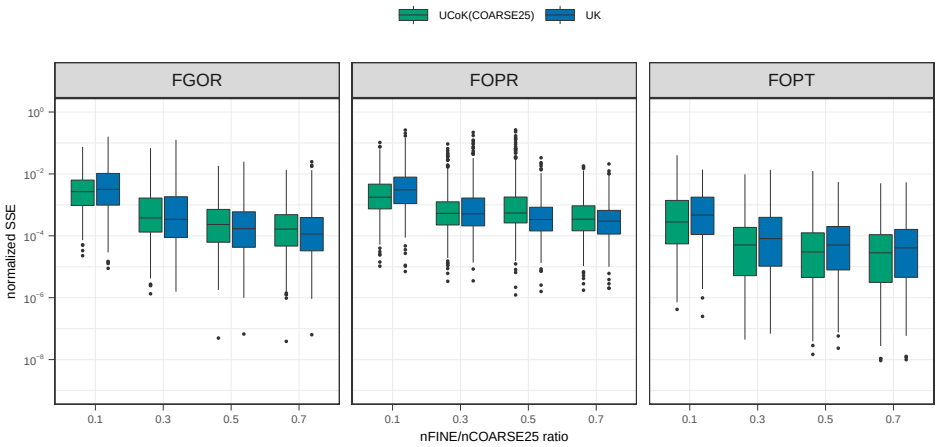

Figure 4.8: Comparison of Test 1 normalized SSE's distribution for increasing ratio of hi-fi response available; predictors are computed with LS variogram fitting, first-order polynomial drift and COARSE25 output as secondary variables.

COARSE50) and Figure 4.8 (where COARSE25 is used). At first glance, the performances of our method improve as the amount of hi-fi outputs available increases (as expected), while the gain with respect to the single-fidelity one decreases. Moreover, the gain in predictive capability is more evident for FGOR and FOPR curves, which present a higher level of uncertainty in output with respect to the initial configuration. Finally, we can observe that the choice of the COARSE25 response in computing the cokriging predictor leads to a worse performance overall: such results are probably due to the impactful simplification of the underlying model given by the upscaling procedure.

Layer	Parameter	Min	Max	Distribution
all	POROm (–)	0.8	1.2	Uniform
all	PERMZm (–)	0.5	2.0	Uniform

Table 4.3: Uncertain parameters for Test 2.

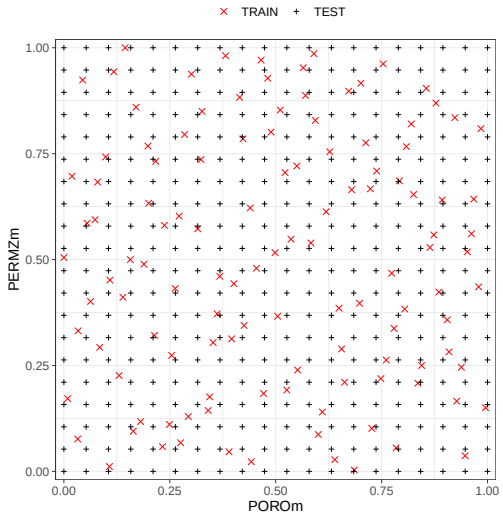

Figure 4.9: Input configurations for Test 2, train (red) and test (black) set; values are rescaled within the unit square.

4.2 Test 2

The second experiment we propose is realized with Model B, with the same uncertain parameters considered for Test 1. Table 4.3 summarizes their range and distribution. As for Test 1, we build two sets of data from the two dimensional input space given by the uncertain parameters (see Figure 4.9):

- a train set of 120 configurations, generated with Latin Hypercube Sampling and evaluated with all levels of flow simulation;
- a test set of 400 configurations on a regular grid over the input space, evaluated only with FINE level flow simulation.

The functional outputs of the numerical simulations for the highest fidelity level are portrayed in Figure 4.10, while Figure 4.11 shows the comparison of curves with respect to fidelity levels and Table 4.4 the summary of computation

Fidelity Level	Min	p0.25	p0.50	p0.75	Max	mean
FINE	288.179	304.025	313.409	326.150	357.650	315.088
COARSE50	37.074	40.415	41.852	42.951	48.114	41.886
COARSE25	6.781	7.620	8.043	8.608	10.969	8.202

Table 4.4: Summary of computation time for Test 2; values are expressed in seconds.

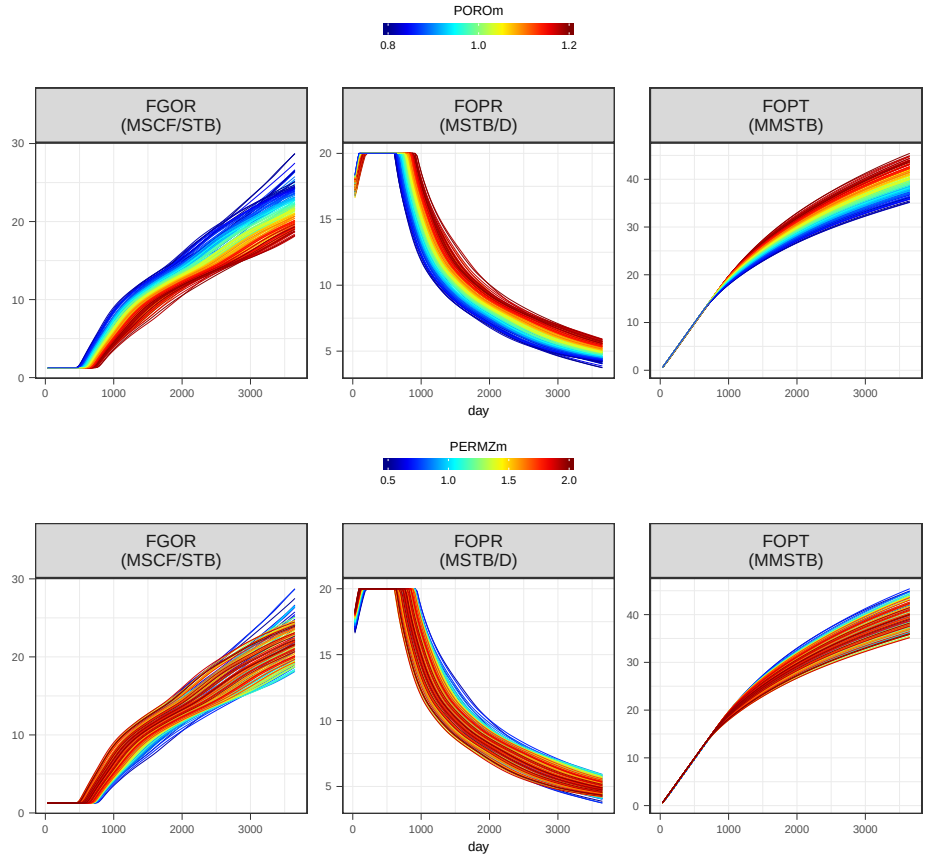

Figure 4.10: Functional outputs of Test 2 for FINE level, coloured by parameters POROm (top) and PERMZm (bottom).

times. We observe a slightly different shape of the outputs with respect to Test 1, even if the overall behaviour of the curves is similar. This is due to the nature of the underlying model which shares with the previous part of the implementation.

Unlike Test 1, in this case the cross-validation approach does not suffer from numerical instability when building the cokriging predictor. Moreover it allows to produce predictions also when a second-order polynomial is chosen as drift model, while the traditional LS approach suffers from a lack of convergence as in the previous test. We perform the same transformations of the FGOR and FOPR curves in order to remove the bounded behaviour, whose results are portrayed in Figure 4.12.

The results of the comparison for this test when the low-fidelity response is COARSE50 are presented in Figure 4.13 and 4.14, while Figure 4.15 and 4.16 show the ones obtained with COARSE25 outputs. Test 2 partially confirms the results of Test 1 in terms of predictive capability gain in conditions of high-fidelity outputs sparsity. This gain, although visually noticeable, appears reduced with respect to Test 1 results, probably due to the increased complexity of the simulation. While cross-validation allows to fit a valid model even when

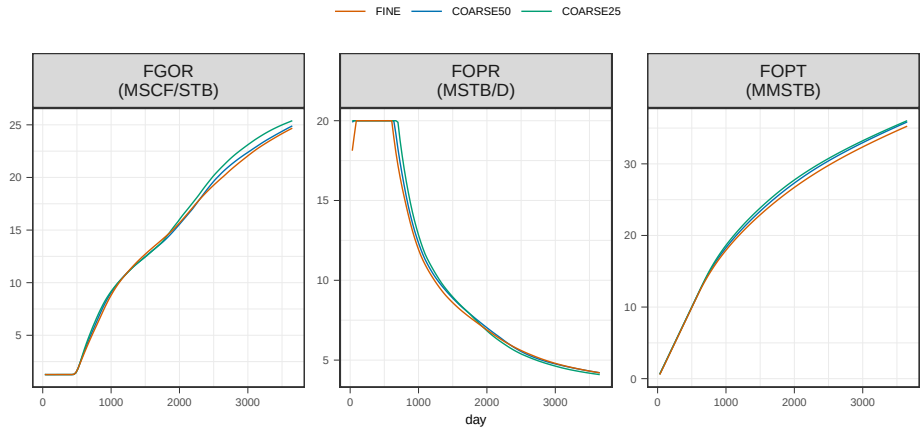

Figure 4.11: Comparison of functional outputs with respect to fidelity level for Test 2.

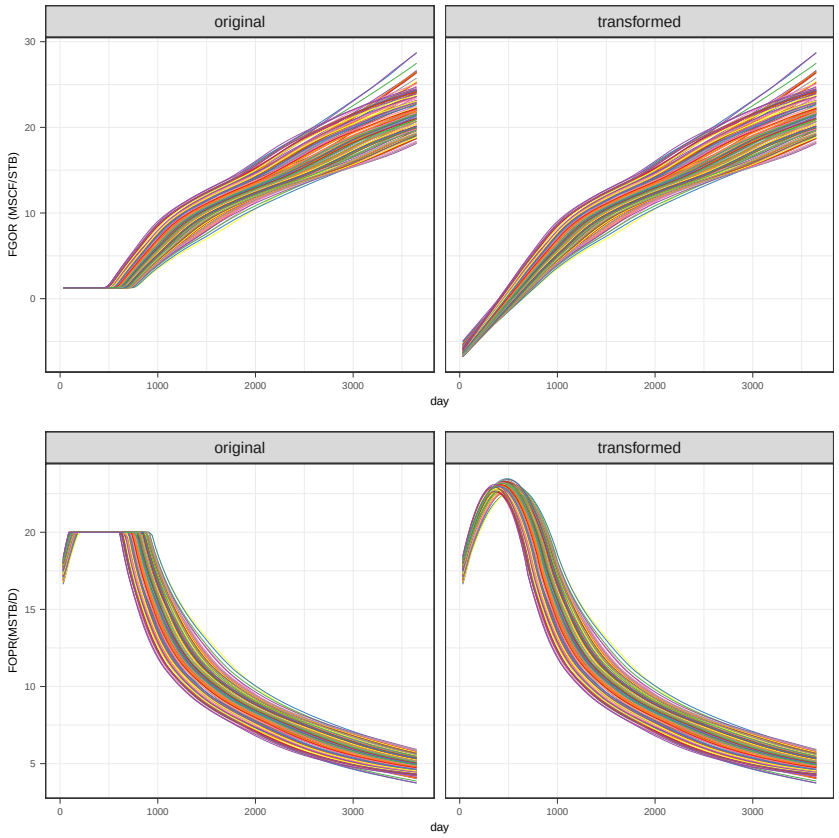

Figure 4.12: Transformation of FGOR (top) and FOPR (bottom) curves; colouring of curves here is only qualitative.

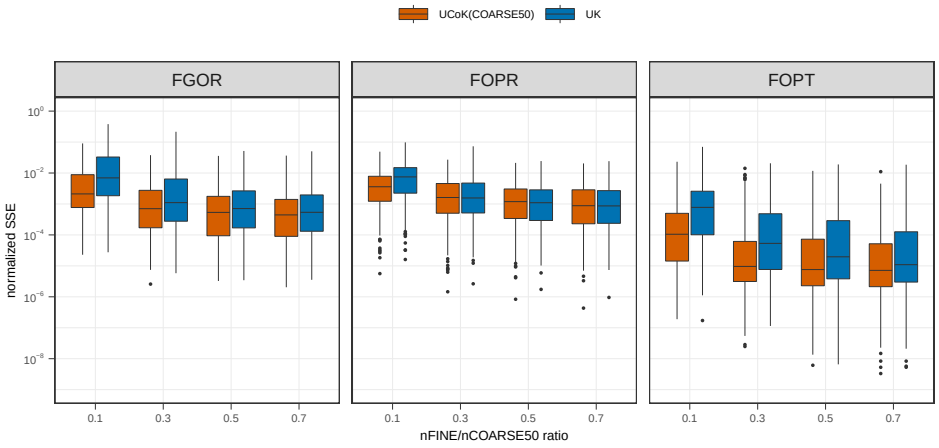

Figure 4.13: Comparison of Test 2 normalized SSE's distribution for for increasing ratio of hi-fi response available; predictors are computed with LS variogram fitting, first-order polynomial drift and COARSE50 output as secondary variables.

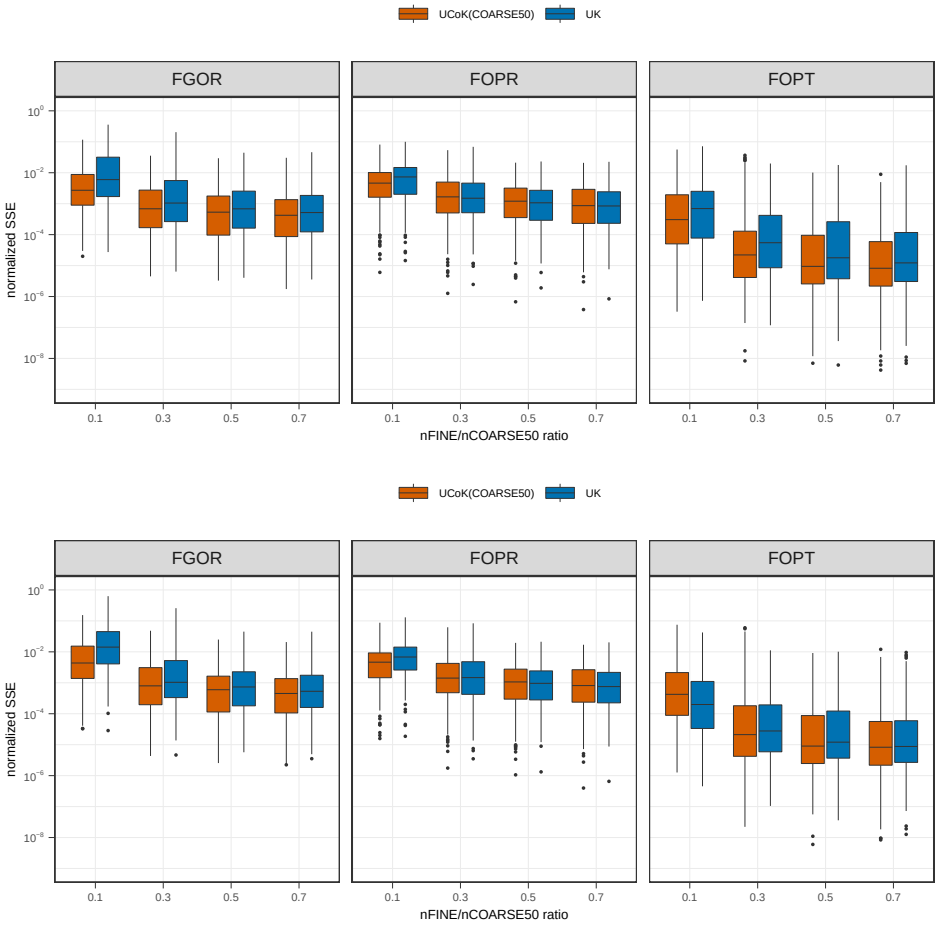

Figure 4.14: Comparison of Test 2 normalized SSE's distribution for increasing ratio of hi-fi response available; predictors are computed with cross-validation variogram fitting, first-order (top) and second-order (bottom) polynomial drift. Level COARSE50 is used as secondary variable.

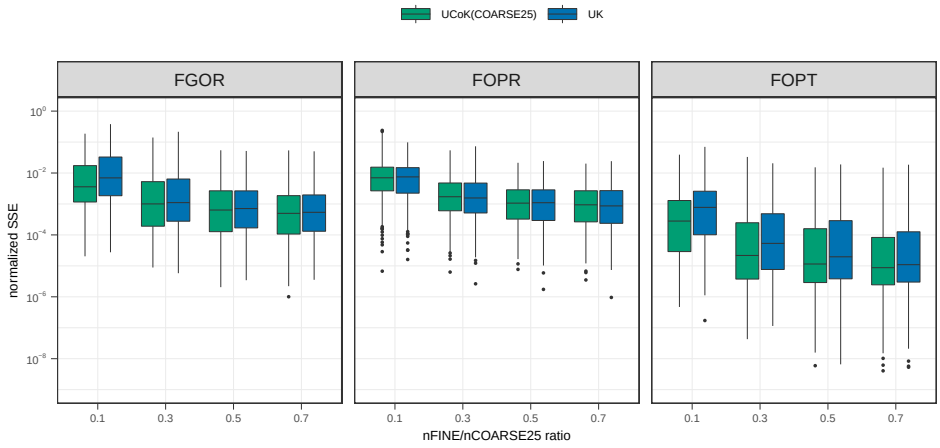

Figure 4.15: Comparison of Test 2 normalized SSE's distribution for increasing ratio of hi-fi response available; predictors are computed with LS variogram fitting, first-order polynomial drift and COARSE25 outputs as secondary variables.

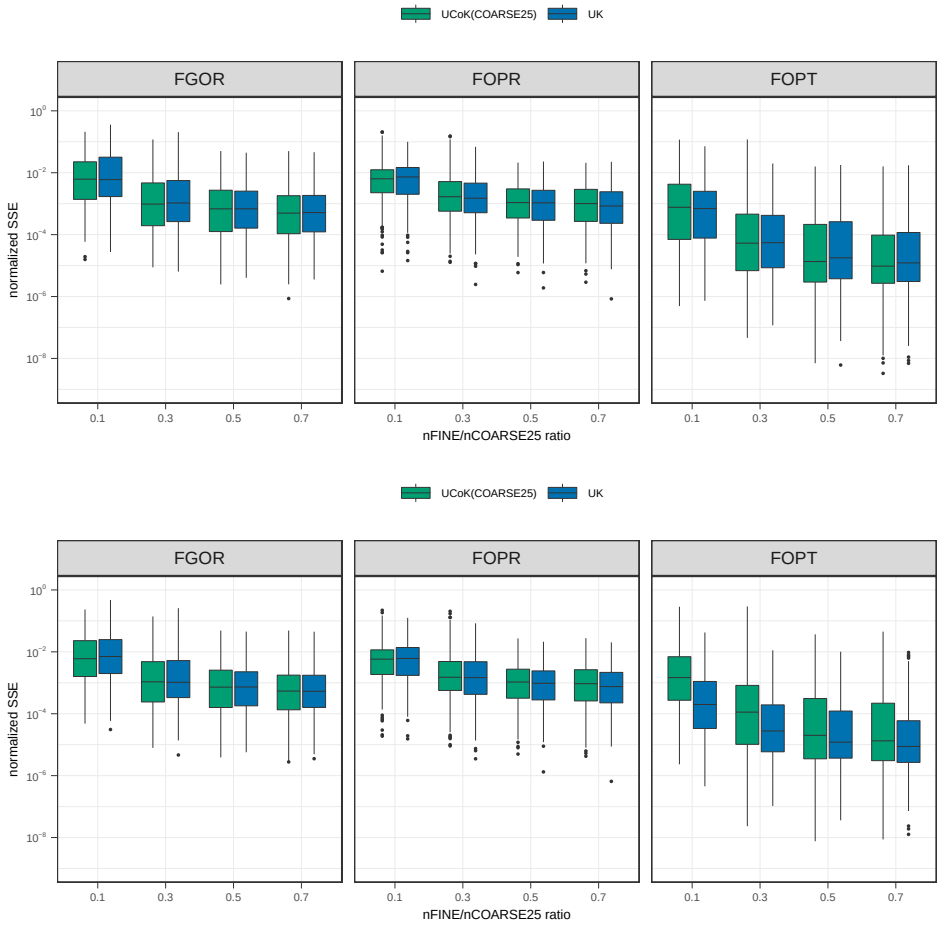

Figure 4.16: Comparison of Test 2 normalized SSE's distribution for increasing ratio of hi-fi response available; predictors are computed with cross-validation variogram fitting, first-order (top) and second-order (bottom) polynomial drift. Level COARSE25 is used as secondary variable.

Layer	Parameter	Min	Max	Distribution
1	POROm (–)	0.8	1.2	Uniform
1	PERMZm (–)	0.5	2.0	Uniform
2	POROm (–)	0.8	1.2	Uniform
2	PERMZm (–)	0.5	2.0	Uniform
3	POROm (–)	0.8	1.2	Uniform
3	PERMZm (–)	0.5	2.0	Uniform

Table 4.5: Uncertain parameters for Test 3.

Fidelity Level	Min	p0.25	p0.50	p0.75	Max	Mean
FINE	162.441	173.053	177.611	183.197	459.796	203.995
COARSE50	21.479	23.151	23.934	24.713	29.967	24.096
COARSE25	12.724	14.003	15.164	16.113	17.400	15.099

Table 4.6: Summary of computation time for Test 3; values are expressed in seconds.

second-order polynomial drift is chosen, the resulting predictor does not improve its performances with respect to the first-order one. In fact, Figure 4.14 and 4.16 show clearly that testing our predictor with second-order drift over FOPT curves as target results in worse performances for both COARSE50 and COARSE25 levels with respect to first-order one. As for Test 1, the use of the lowest fidelity level produces worse predictions for both LS and cross-validation approaches. Finally, we observe that performances on FOPT curves prediction improve with respect to Test 1 in terms of median error, while results on FOPR outputs are equivalent to the single-fidelity predictor.

4.3 Test 3

The last numerical experiment we perform exploits again reservoir model B with an approach to uncertainty that is more adherent to common practice in reservoir simulation. We consider multipliers for porosity and vertical permeability in each layer for a total of six uncertain parameters, whose range and distribution are described in Table 4.5. Dealing with a 6-dimensional space a uniform sampling approach is infeasible; hence both train and test set will be sampled with a space filling design (Latin Hypercube). We build the usual two sets of data from the 6-dimensional input space given by the uncertain parameters (see Figure 4.17 for the marginal two-dimensional representations):

- a train set of 100 configurations, evaluated as previous tests with all levels of flow simulation;
- a test set of 400 configurations, evaluated only with FINE level flow simulation.

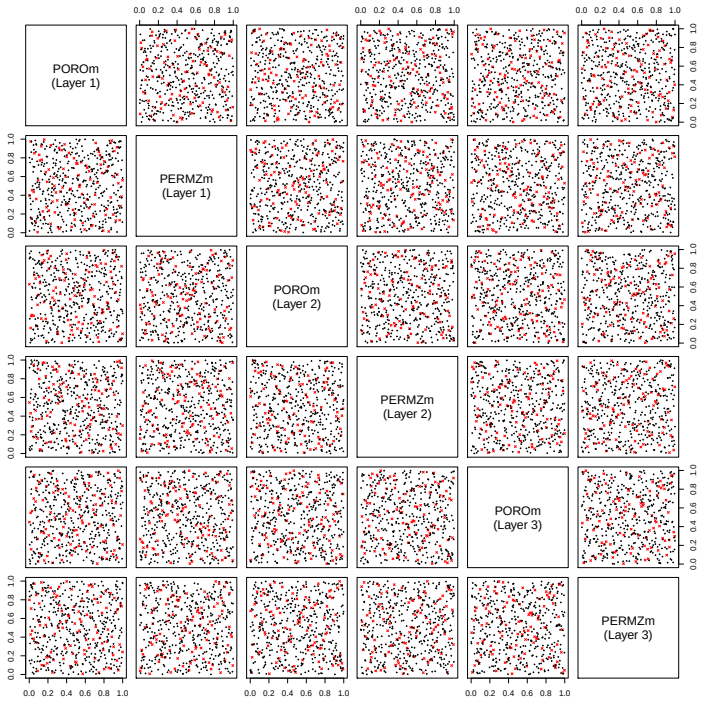

Figure 4.17: Pairs plot of input configurations for Test 3, train (red) and test (black) set; values are rescaled within the unit hypercube.

Figure 4.18 shows the comparison between fidelity levels, while FINE output curves are portrayed in Figure 4.19 and 4.20 with colouring based on every uncertain parameter; computation times are summarized in Table 4.6.

After transforming the FGOR and FOPR outputs accordingly to previously described procedures, we construct the comparison between our method and the single-fidelity approach. Due to the increased number of dimension the approach of variogram fitting through least-squares results in a lack of convergence for every combination. Moreover, since our main interest is to compare the two approaches for reduced amounts of hi-fi observations, a second order

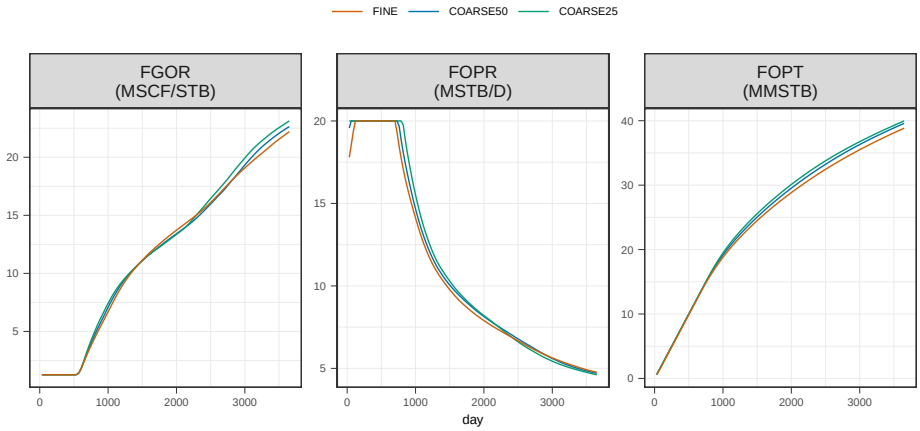

Figure 4.18: Comparison of functional outputs with respect to fidelity level for Test 3.

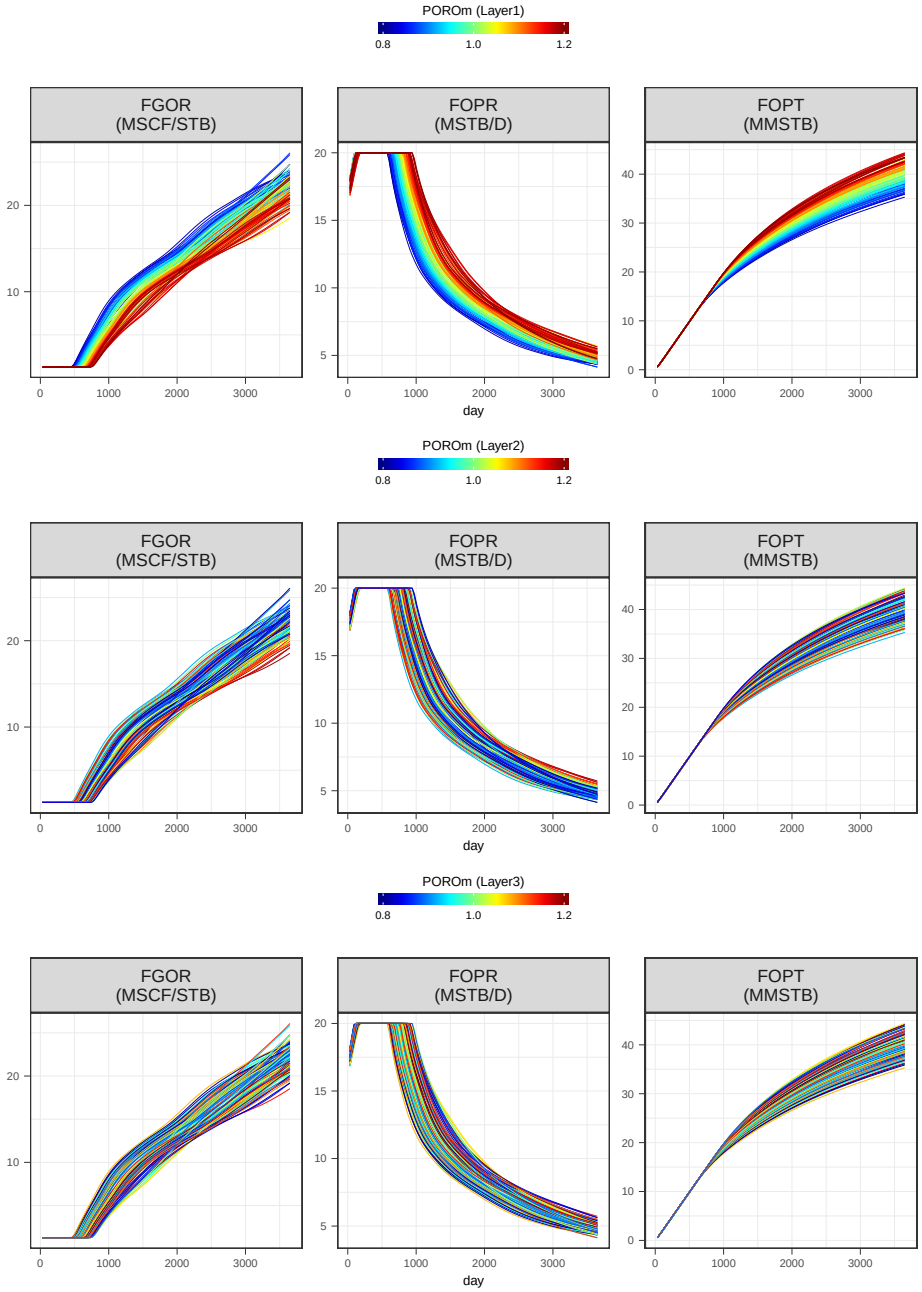

Figure 4.19: Functional outputs of Test 3 for FINE level coloured by parameter POROm in Layer 1 (top), Layer 2 (middle) and Layer 3 (bottom).

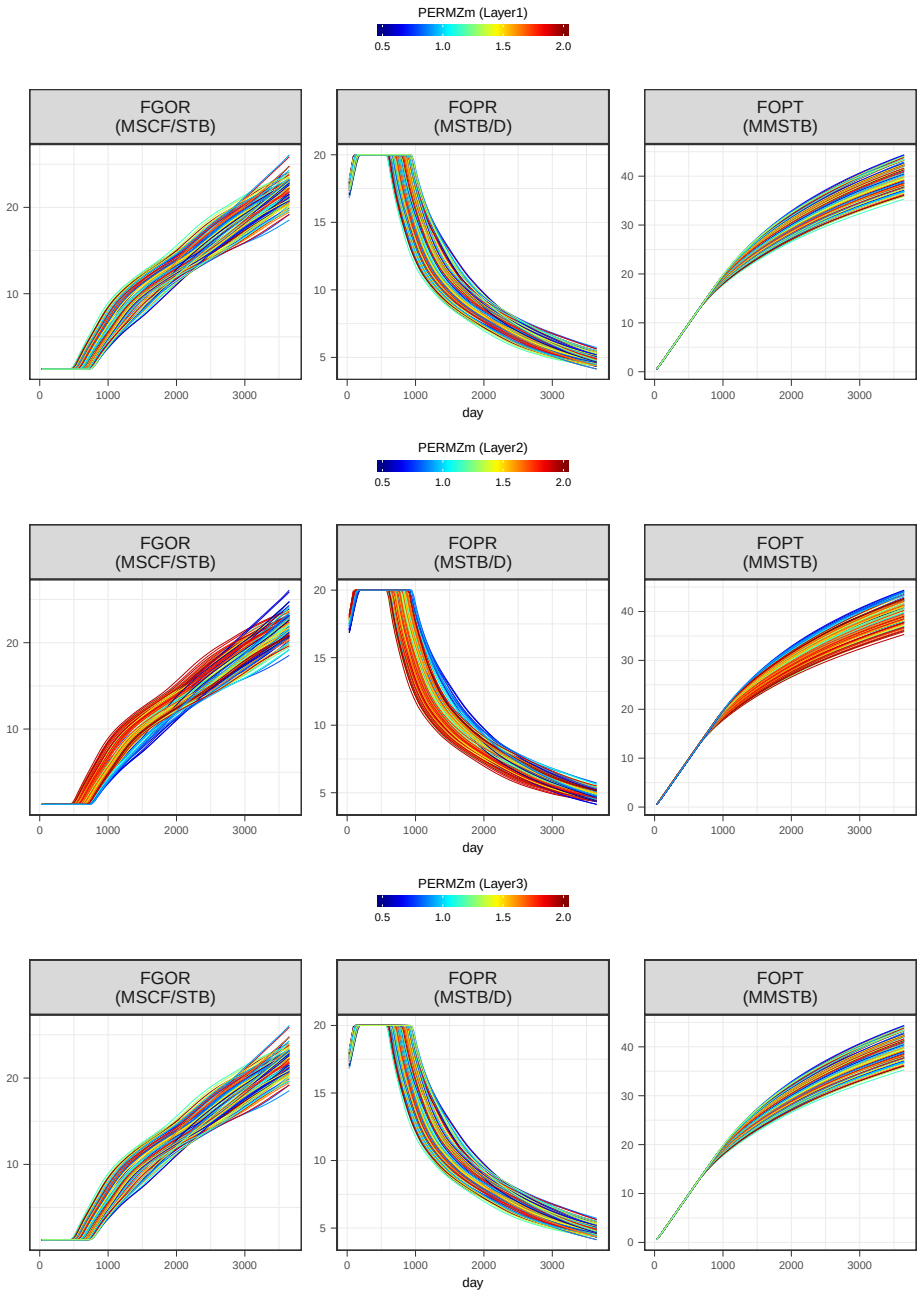

Figure 4.20: Functional outputs of Test 3 for FINE level coloured by parameter PERMZm in Layer 1 (top), Layer 2 (middle) and Layer 3 (bottom).

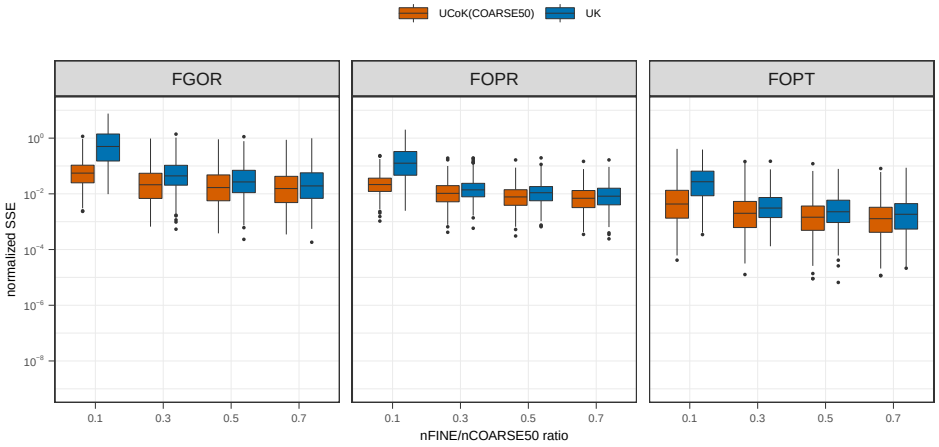

Figure 4.21: Comparison of Test 3 normalized SSE's distribution for increasing ratio of hi-fi response available; predictors are computed with cross-validation variogram fitting, first-order polynomial drift and COARSE50 outputs as secondary variables.

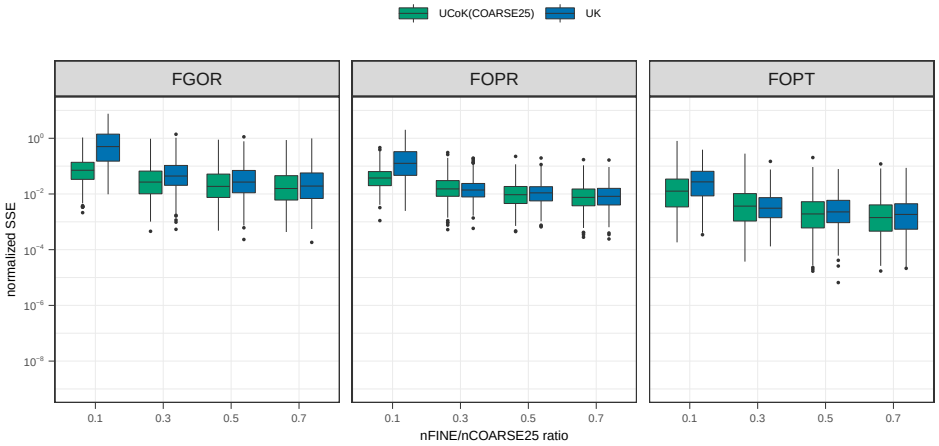

Figure 4.22: Comparison of Test 3 normalized SSE's distribution for increasing ratio of hi-fi response available; predictors are computed with cross-validation variogram fitting, first-order polynomial drift and COARSE25 outputs as secondary variables.

polynomial drift is undesirable due to the number of regression coefficients necessary for modeling. We therefore opt for a first-order polynomial drift and cross-validation variogram fitting, which produces the distributions of normalized SSE's summarized in Figure 4.21 when COARSE50 is the secondary variable and Figure 4.22 when COARSE25 is. The results are in agreement with previous tests; moreover median error for FGOR and FOPR predictors at 10% ratio between hi-fi and lo-fi observations decreases of almost one degree of magnitude. This gain in performances is due to the increased dimensionality of the uncertainty space, in which FINE observations alone are not enough to characterize the spatial dependence between outputs. One last aspect of the comparison of the two methods is worth mentioning: if we compare median normalized SSE's values across the different FINE-COARSE ratios, we observe

that our method allows obtaining similar results with respect to its univariate alternative with fewer high-fidelity outputs. This result is of particular importance, since the computational time needed for the construction of the two predictors is strictly related to the amount of simulated curves. In particular, from the inspection of FGOR results displayed in Figure 4.22 we can conclude that UCoK performance with 30% of FINE curves available is similar to the ones of UK predictor built with 70% of such curves. Assuming the mean times of computation reported in Table 4.6 as a fair indicator of the actual computational effort for one simulation run at each level of fidelity, we can obtain an estimate of the computational cost needed to build each predictor. In the aforementioned case, as the total amount of COARSE curves is 100, the estimated cost of the 30%-UCoK predictor is $203.995 \times 30 + 15.099 \times 100 = 7629.75$ seconds, while the estimated cost of the equivalent single-fidelity predictor (with 70% FINE curves) is $203.995 \times 70 = 14279.65$. The ratio of the two values is $7629.75/14279.65 \simeq 0.53$, that is, in order to get a similar predictive result the multifidelity approach is almost twice as fast as the univariate one. The incorporation of secondary data in form of proxy solution therefore improves the overall prediction, while being computationally less demanding in terms of resources.

Conclusions

The aim of this work has been the development of a multi-fidelity approach to surrogate modeling, exploiting a functional cokriging statistical model within the research area of reservoir simulation. In this framework, the lack of knowledge in modeling parameters often results in high-dimensionality of the input uncertainty space. In order to prevent the issues of fitting a valid model to variogram estimates through least-squares in such ample spaces, we have established a cross-validation procedure for variogram fitting. This approach, although limited to a single family of valid models and range optimization, allows us to estimate valid covariance matrices even in conditions of high sparsity of sampled locations. We have also developed an extensive procedure to compare the aforementioned method with the most recent geostatistical single-fidelity approach available in literature. We tested the predictive performances of our approach on three types of output curves in the reservoir simulation framework with different levels of variability with respect to uncertainty on input. The results of our numerical experiments allow us to draw the following considerations:

- The transformation procedures applied to constrained curves represent an effective tool in lack of a better geometry in which to embed such functions. However, the choice of a quadratic substitution for FOPR curves could be improved, e.g. by taking into account the degree of smoothness in correspondence to junctions with a spline-based approach.

- The impact of drift estimation on the prediction is decisive, since the choice of a too consistent deterministic component could result in the loss of the spatial dependence structure in the residuals.

- Our method performs better than its single-fidelity counterpart when the FINE-COARSE ratio is low, and the gain in performances is more conspicuous as the dimensionality of the input space increases. This is probably due to the difficult characterization of simulation outputs in presence of a low number of high-fidelity data, accentuated by the increased sparsity typical of larger input spaces.

- The predictive capability of the model is affected by the level of ap-

proximation of the low-fidelity model: a significant simplification of the fine-grid simulation, although computationally more efficient, results in the inability to bring information on the high-fidelity output.

- An important issue of the prediction is computing the numerical solution of the kriging system, since the condition number of the covariance matrix increases sometimes dramatically when considering multivariate processes. A possible explanation resides in the choice of the experimental design: whenever two points in the input space are too close together, the corresponding rows of the covariance matrix become almost linearly dependent, making the whole matrix close to be singular. When this occurs the result is the inability to compute a valid solution, restricting the potential applicability of the method.

In light of these considerations, we now discuss on some possible future development in this context:

- a comparison study on real scenarios, considering more complex reservoir models and more precise uncertainty characterizations, would be a first step in demonstrating the capability of our method in the surrogate modeling field;
- the extension of cross-validation procedure to different families of valid models would allow to cope with more general spatial dependence structures (e.g., quadratic near the origin, discontinuous or characterized by anisotropy);
- the implementation of more reliable solving method of the cokriging system, as discussed before, may overcome the numerical instability introduced with the addition of secondary variables to the statistical model;
- since we focused on exploring the uncertainty input space without actually considering how different factor affect the variability in the resulting simulation, we suggest the realization of a study on variance-based sensitivity analysis for functional data (e.g. extending the theory of Sobol indices [36]).

To sum up, we believe that these analyses establish a good starting point for the adoption of functional cokriging methods in the uncertainty quantification field. Also, the great generality with respect to admissible outputs and the flexibility in modeling stochastic processes allow our techniques to be extended to a broader set of numerical experiments.

Bibliography

[1] T. Ahmed. *Working guide to reservoir rock properties and fluid flow.* Oxford: Gulf Professional, 2009.

[2] E. Aliyev and L. J. Durlofsky. 'Multilevel Field Development Optimization Under Uncertainty Using a Sequence of Upscaled Models'. In: *Mathematical Geosciences* 49.3 (2017), pp. 307–339.

[3] M. Bohorquez, R. Giraldo, and J. Mateu. 'Optimal sampling for spatial prediction of functional data'. In: *Statistical Methods & Applications* 25.1 (2016), pp. 39–54.

[4] M. Bohorquez, R. Giraldo, and J. Mateu. 'Multivariate functional random fields: prediction and optimal sampling'. In: *Stochastic Environmental Research and Risk Assessment* 31.1 (2017), pp. 53–70.

[5] F. Bottazzi and E. Della Rossa. 'A Functional Data Analysis Approach to Surrogate Modeling in Reservoir and Geomechanics Uncertainty Quantification'. In: *Mathematical Geosciences* 49.4 (2017), pp. 517–540.

[6] G. E. P. Box and K. B. Wilson. 'On the Experimental Attainment of Optimum Conditions'. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 13.1 (1951), pp. 1–45.

[7] S. E. Buckley and M. C. Leverett. 'Mechanism of Fluid Displacement in Sands'. In: *Transactions of the AIME* 146.01 (1942), pp. 107–116.

[8] W. Caballero, R. Giraldo, and J. Mateu. 'A universal kriging approach for spatial functional data'. In: *Stochastic Environmental Research and Risk Assessment* 27.7 (2013), pp. 1553–1563.

[9] J.-P. Chilès and P. Delfiner. *Geostatistics: Modeling Spatial Uncertainty.* John Wiley & Sons, Inc., 1999.

[10] N. A. C. Cressie. *Statistics for Spatial Data.* John Wiley & Sons, Inc., 1993.

[11] H. P. G. Darcy. *Les fontaines publiques de la ville de Dijon.* Paris: Dalmont, 1856.

[12] L. De Cesare, D. E. Myers, and D. Posa. 'Estimating and modeling space–time correlation structures'. In: *Statistics & Probability Letters* 51.1 (2001), pp. 9–14.

[13] L. J. Durlofsky. 'Coarse scale models of two phase flow in heterogeneous reservoirs: volume averaged equations and their relationship to existing upscaling techniques'. In: *Computational Geosciences* 2.2 (1998), pp. 73–92.

[14] R. Eymard, T. Gallouët, and R. Herbin. 'Finite volume methods'. In: *Handbook of Numerical Analysis*. Elsevier, 2000, pp. 713–1018.

[15] A. I. J. Forrester and A. J. Keane. 'Recent advances in surrogate-based optimization'. In: *Progress in Aerospace Sciences* 45.1 (2009), pp. 50–79.

[16] R. Giraldo, P. Delicado, and J. Mateu. *Geostatistics for functional data: an ordinary kriging approach*. Tech. rep. Universitat Politècnica de Catalunya, Departament d'Estadística i Investigació Operativa, 2008.

[17] R. Giraldo, P. Delicado, and J. Mateu. 'Continuous Time-Varying Kriging for Spatial Prediction of Functional Data: An Environmental Application'. In: *Journal of Agricultural, Biological, and Environmental Statistics* 15.1 (2010), pp. 66–82.

[18] P. Goovaerts. *Geostatistics for Natural Resources Evaluation*. Oxford: Oxford University Press, 1997.

[19] M. Goulard and M. Voltz. 'Geostatistical Interpolation of Curves: A Case Study in Soil Science'. In: *Quantitative Geology and Geostatistics*. Ed. by A. Soares. Dordrecht: Springer Netherlands, 1993, pp. 805–816.

[20] O. Grujic, A. Menafoglio, G. Yang, and J. Caers. 'Cokriging for multivariate Hilbert space valued random fields: application to multi-fidelity computer code emulation'. In: *Stochastic Environmental Research and Risk Assessment* 32.7 (2018), pp. 1955–1971.

[21] L. Horváth and P. Kokoszka. *Inference for Functional Data with Applications*. Springer Series in Statistics. Springer New York, 2012.

[22] R. Ignaccolo, J. Mateu, and R. Giraldo. 'Kriging with external drift for functional data for air quality monitoring'. In: *Stochastic Environmental Research and Risk Assessment* 28.5 (2014), pp. 1171–1186.

[23] M. Kennedy and A. O'Hagan. 'Predicting the output from a complex computer code when fast approximations are available'. In: *Biometrika* 87.1 (2000), pp. 1–13.

[24] R. W. Lewis and P. M. Roberts. 'The Finite Element Method in Porous Media Flow'. In: *Fundamentals of Transport Phenomena in Porous Media*. Springer Netherlands, 1984, pp. 805–897.

[25] K.-A. Lie. *An Introduction to Reservoir Simulation Using MATLAB/GNU Octave*. Cambridge University Press, 2019.

[26] J. S. Marron and A. M. Alonso. 'Overview of object oriented data analysis'. In: *Biometrical Journal* 56.5 (2014), pp. 732–753.

[27] G. Matheron. 'Principles of geostatistics'. In: *Economic Geology* 58.8 (1963), pp. 1246–1266.

[28] 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'. In: *Technometrics* 21.2 (1979), p. 239.

[29] A. Menafoglio, P. Secchi, and M. Dalla Rosa. 'A Universal Kriging predictor for spatially dependent functional data of a Hilbert Space'. In: *Electron. J. Statist.* 7 (2013), pp. 2209–2240.

[30] D. Nerini, P. Monestiez, and C. Manté. 'Cokriging for spatial functional data'. In: *Journal of Multivariate Analysis* 101.2 (2010), pp. 409–418.

[31] A. S. Odeh. 'Comparison of Solutions to a Three-Dimensional Black-Oil Reservoir Simulation Problem'. In: *Journal of Petroleum Technology* 33.01 (1981), pp. 13–25.

[32] J. O. Ramsay and B. W. Silverman. *Functional Data Analysis*. Springer New York, 2005.

[33] F. A. Rasmussen et al. 'The Open Porous Media Flow Reservoir Simulator'. In: *arXiv e-prints*, arXiv:1910.06059 (2019). arXiv: `1910.06059 [cs.OH]`.

[34] C. Rizzo. '3D Upscaling of Reservoir Properties using the Mixed Finite Element Methodon Non-Matching Grids'. MA thesis. Politecnico di Milano, 2013.

[35] C. Sammut and G. I. Webb. 'Leave-One-Out Cross-Validation'. In: *Encyclopedia of Machine Learning*. Ed. by C. Sammut and G. I. Webb. Boston, MA: Springer US, 2010, pp. 600–601.

[36] I. M. Sobol. 'Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates'. In: *Mathematics and Computers in Simulation* 55.1-3 (2001), pp. 271–280.

[37] A. Thenon, V. Gervais, and M. Le Ravalec. 'Multi-fidelity meta-modeling for reservoir engineering - application to history matching'. In: *Computational Geosciences* 20.6 (2016), pp. 1231–1250.

[38] J. A. Trangenstein and J. B. Bell. 'Mathematical Structure of the Black-Oil Model for Petroleum Reservoir Simulation'. In: *SIAM Journal on Applied Mathematics* 49.3 (1989), pp. 749–783.

[39] B. Yeten, A. Castellini, B. Guyaguler, and W. H. Chen. 'A Comparison Study on Experimental Design and Response Surface Methodologies'. In: *SPE Reservoir Simulation Symposium*. Society of Petroleum Engineers, 2005.

Appendix A

Numerical tests codes

This Appendix present three original R codes developed for the numerical tests presented in Chapter 4. Our codes exploit the following existing packages:

- `fda` (`www.functionaldata.org`) and `fda.usc` (`www.jstatsoft.org`) for the representation through basis functions and the computation of functional norms;
- `fdagstat` (`github.com/ogru/fdagstat`) for the actual trace-cokriging predictor construction;
- `caret` (`github.com/topepo/caret/`) to construct folds in cross-validation procedures;
- `purrr` (`purrr.tidyverse.org`) for handling errors arising from computation.

A.1 TransformFGOR

`TransformFGOR()` function transforms FGOR curves whenever the initial output is flat, as described in Chapter 4. This behaviour is usually due to the input settings of the reservoir model: in conditions of unsaturated oil as the initial pressure at well bore is greater than the bubble point no gas evolves from the oil, inducing constant FGOR output.

```
1 TransformFGOR <- function(curves, tstep, offset=5, precision=1e-2,
      max.values=4, t.max=floor(0.8*length(tstep))) {
2
3   nstep <- length(tstep)
4   nobs  <- ncol(curves)
5
6   if( nstep != nrow(curves) ) stop('Number of evaluation per curve and
      timesteps must coincide. Exiting! ')
7
8   l     <- vector()
9   res   <- matrix(NA, nrow=nstep, ncol=nobs)
```

```
10
11   for( i in 1:nobs ) {
12
13     ## detect index of breakthrough point
14     l[i] <- max(which(abs(diff(curves[2:t.max,i])) <= precision))
15
16     ## if flat response is detected
17     if(l[i] != -Inf) {
18
19       ## add offset to improve estimation performances
20       l[i]     <- l[i] + offset
21
22       ## separate the remaining part of the curve
23       x.tmp    <- tstep[-(1:l[i])]
24       y.tmp    <- curves[-(1:l[i]),i]
25
26       ## regression on initial non-flat values
27       df.tmp   <- data.frame(y=y.tmp[1:max.values], x=x.tmp[1:max.values])
28       g        <- lm(y ~ x, data=df.tmp)
29
30       ## regression solution for initial flat response
31       x.est    <- tstep[1:l[i]]
32       y.est    <- predict(g, newdata=data.frame(x=x.est))
33
34       ## substitute regression solution to original curve flat response
35       res[ ,i] <- c(y.est, curves[-(1:l[i]),i])
36
37     } else {
38
39       ## no flat response
40       res[ ,i] <- curves[ ,i]
41     }
42   }
43
44   return(res)
45
46 }
```

A.2 TransformFOPR

`TransformFOPR()` function transforms FOPR curves whenever they reach the maximum production rate, as described in Chapter 4. This behaviour is due to the initial settings of the reservoir model: one of the parameters in input is the maximum production capacity of wells, whose value cannot be exceeded.

```
1  TransformFOPR <- function(curves, tstep, offset=4, upper.bound) {
2
3    nstep  <- length(tstep)
4    nobs   <- ncol(curves)
5
6    ## offset must be an even number
7    offset <- ifelse(offset %% 2 == 1, offset-1, offset)
8
9    if( nstep != nrow(curves) ) stop('Number of evaluation per curve and
         timesteps must coincide. Exiting!')
10
```

```
  l.min <- vector()
  l.max <- vector()
  res   <- matrix(NA, nrow=nstep, ncol=nobs)

  for( i in 1:nobs ) {

    if( all(curves[ ,1] < upper.bound) ) {
      ##  plateau is not reached
      res[ ,i] <- curves[ ,i]
    } else {

      ## detect indexes of plateau boundary
      l.min[i] <- min(which(curves[,i] >= upper.bound))
      l.max[i] <- max(which(curves[,i] >= upper.bound))

      if( l.min[i] == l.max[i] ) {
        ## false positive, plateau reached only at one timestep
        res[ ,i] <- curves[ ,i]
      } else {

        ## select pre and post-plateau indexes
        idx.pre  <- max((l.min[i]-offset),1):l.min[i]
        idx.post <- l.max[i]:(l.max[i]+offset)

        ## fit quadratic linear model to pre and post-plateau values
        x.tmp    <- tstep[c(idx.pre,idx.post)]
        y.tmp    <- curves[c(idx.pre,idx.post), i]
        df.tmp   <- data.frame(x=x.tmp, y=y.tmp)
        g        <- lm(y ~ poly(x, degree=2, raw=TRUE), data=df.tmp)

        if( l.min[i] == 1 ){
          ## plateau starts at first value (no ramp-up)
          x.est <- tstep[l.min[i]:(l.max[i]+offset/2)]
        } else {
          ## plateau occurs after ramp-up phase
          x.est <- tstep[(l.min[i]-offset/2):(l.max[i]+offset/2)]
        }

        y.est    <- predict(g, newdata=data.frame(x=x.est))

        ## substitute plateau with regression solution
        res[ ,i] <- curves[ ,i]
        res[tstep %in% x.est,i] <- y.est
      }
    }
  }

  return(res)

}
```

A.3 FitRangeCV

`FitRangeCV()` function allows to optimize the range parameter within a family of valid variogram models, computing the k-fold cross-validation SSE for each value of range and selecting the one whose associated error is minimum.

```
FitRangeCV <- function(coord, idx, tstep, curves.l1, curves.l2=NULL,
    drift.frml='~ .', model.name='Sph', n.lags=NULL, lag.max=NULL,
    n.folds=5, seed=NULL) {

  ## exit if required packages are missing, otherwise load them
  if(!require(fda)) stop()
  if(!require(fda.usc)) stop()
  if(!require(fdagstat)) stop()
  if(!require(purrr)) stop()
  if(!require(caret)) stop()

  diag.len <- sqrt(ncol(coords))

  ## check if seed is provided
  if( is.null(seed) ) stop('Cross-Validation requires setting a seed.
      Exiting!')

  ## compute r.min and r.max if not provided
  if( is.null(r.min) ) r.min <- 0.1*diag.len
  if( is.null(r.max) ) r.min <- 0.8*diag.len

  ## compute lag.max if not provided
  if( is.null(lag.max) ) {
    warning('Maximum lag missing, it will be computed (for high number of
        samples this could be slow)...')
    lag.max <- 0.6*diag.len
  }

  ## build folds and ranges sequence
  set.seed(seed)
  folds    <- caret::createFolds(y=idx, k=n.folds, list=TRUE)
  ranges   <- seq(r.min, r.max, by=0.05)
  res      <- NULL

  if( is.null(curves.l2) ) {

    ## only one variable provided, i.e., trace-kriging
    for( i in 1:length(ranges) ) {

      r   <- ranges[i]
      tmp <- NULL
      for( f in 1:n.folds ) {

        f.idx <- folds[[f]]
        frml  <- drift.frml

        ## build fdagstat object
        g.cv  <- fstat(NULL, vName="Fine",
            scalar=FALSE,Coordinates=coord[idx[-f.idx], ],
            Functions=as.data.frame(curves.l1[ ,idx[-f.idx]]))

        ## OLS estimate of drift component
        g.cv  <- estimateDrift(frml, g.cv, Intercept=TRUE)

        ## variogram estimate (not used for fitting)
        g.cv  <- fvariogram("~.", g.cv, Nlags=n.lags, LagMax=lag.max,
            ArgStep=mean(tstep), useResidual=TRUE, comments=FALSE)

        ## adding valid model with given range
        g.cv  <- fitVariograms(g.cv, model=vgm(psill=1, model=model.name,
```

```
          range=r), fitSills=FALSE, fitRanges=FALSE)
54        g.cv  <- addCovariance(g.cv, type='omni')
55
56        ## iterate GLS drift estimate and residuals computation
57        for ( j in 1:5 ) {
58          g.cv <- estimateDrift(frml, g.cv, .type="GLS", Intercept=TRUE)
59          g.cv <- fvariogram("~.", g.cv, Nlags=n.lags, LagMax=lag.max,
                ArgStep=mean(tstep), useResidual=TRUE, comments=FALSE)
60          g.cv <- fitVariograms(g.cv, model=vgm(psill=1, model=model.name,
                range=r), fitSills=FALSE, fitRanges=FALSE)
61          if( j == 5 ) {
62            ## rescale sill
63            mm <- max(sapply(g.cv$model$omni, function(el) el$psill ))
64            g.cv$model$omni$Fine$psill <- g.cv$model$omni$Fine$psill / mm
65          }
66          g.cv <- addCovariance(g.cv, type='omni')
67        }
68
69        ## prevent errors due to numerical instability of matrix inversion
70        prdct         <- function(g, a, b, c) { predictFstat(g,
              .newCoordinates=a, .what=b, .type=c) }
71        prdct2        <- purrr::possibly(prdct, otherwise = NA)
72        ck.pred       <- prdct2(g.cv, coord[idx[f.idx], ], "Fine", "UK")
73
74        ## compute prediction error
75        if( any(is.na(ck.pred)) ) {
76          tmp[f] <- NA
77        } else {
78          ## build basis representation to compute norms
79          sse.cv        <- NULL
80          normconst.tmp <- NULL
81          bspbasis      <- create.bspline.basis(range(tstep), nbasis=30)
82          real          <- Data2fd(argval=tstep, y=curves.l1[
                ,idx[f.idx]], basisobj=bspbasis)
83          pred          <- Data2fd(argval=tstep, y=ck.pred$Forecast,
                basisobj=bspbasis)
84
85          if( length(f.idx) == 1 ) {
86            ## one curve per fold, no need to normalize
87            tmp[f] <- fda.usc::norm.fd(real - pred)^2
88          } else {
89            ## compute mean function for SSE normalization
90            real.mean <- Data2fd(argval=tstep, y=rowMeans(curves.l1[
                  ,idx[f.idx]]), basisobj=bspbasis)
91            for( j in 1:ncol(curves.l1[ ,idx[f.idx]]) ) {
92              sse.cv[j]        <- fda.usc::norm.fd(real[j] - pred[j])^2
93              normconst.tmp[j] <- fda.usc::norm.fd(real[j] - real.mean)^2
94            }
95
96            ## normalize SSE
97            normconst.cv <- sum(normconst.tmp)/ncol(curves.l1[
                  ,idx[f.idx]])
98            sse.norm.cv  <- sse.cv/normconst.cv
99
100           ## compute mean SSE within fold
101           tmp[f]       <- mean(sse.norm.cv)
102         }
103       }
104     }
105
106     ## compute mean SSE between folds
```

```
107       res[i] <- mean(tmp, na.rm=TRUE)
108       cat(sprintf('r = %.2f | value = %.7f (NA = %d)\n', r, res[i],
              sum(is.na(tmp))))
109
110     }
111
112     ## select argmin of sse vector
113     range.best <- ranges[which.min(res)]
114     cat(sprintf('chosen r = %.2f \n', range.best))
115
116   } else {
117
118     ## two variables provided, i.e., trace-cokriging
119     for( i in 1:length(ranges) ) {
120
121       r   <- ranges[i]
122       tmp <- NULL
123       for( f in 1:n.folds ) {
124
125         f.idx <- folds[[f]]
126         frml  <- drift.frml
127
128         ## build fdagstat object
129         g.cv  <- fstat(NULL, vName="Fine", scalar=FALSE,
                Coordinates=coord_train[idx[-f.idx], ],
                Functions=as.data.frame(curves.l1[ ,idx[-f.idx]]))
130         g.cv  <- fstat(g.cv, vName="Coarse", scalar=FALSE,
                Coordinates=coord_train, Functions=as.data.frame(curves.l2))
131
132         ## OLS estimate of drift component
133         g.cv  <- estimateDrift(frml, g.cv, Intercept=TRUE)
134
135         ## variogram estimate (not used for fitting)
136         g.cv  <- fvariogram("~.", g.cv, Nlags=n.lags, LagMax=lag.max,
                ArgStep=mean(tstep), useResidual=TRUE, comments=FALSE)
137
138         ## adding valid model with given range
139         g.cv  <- fitVariograms(g.cv, model=vgm(psill=1, model=model.name,
                range=r), fitSills=FALSE, fitRanges=FALSE)
140         g.cv  <- addCovariance(g.cv, type='omni')
141
142         ## iterate GLS drift estimate and residuals computation
143         for ( j in 1:5 ) {
144           g.cv <- estimateDrift(frml, g.cv, .type="GLS", Intercept=TRUE)
145           g.cv <- fvariogram("~.", g.cv, Nlags=20, LagMax=0.60,
                  ArgStep=30.4, useResidual=TRUE, comments=FALSE)
146           g.cv <- fitVariograms(g.cv, model=vgm(psill=1, model=model.name,
                  range=r), fitSills=FALSE, fitRanges=FALSE)
147           if( j == 5 ) {
148             ## rescale sills
149             mm <- max(sapply(g.cv$model$omni, function(el) el$psill ))
150             g.cv$model$omni$Fine$psill        <-
                    g.cv$model$omni$Fine$psill / mm
151             g.cv$model$omni$Coarse$psill      <-
                    g.cv$model$omni$Coarse$psill / mm
152             g.cv$model$omni$Fine.Coarse$psill <-
                    g.cv$model$omni$Fine.Coarse$psill / mm
153           }
154           g.cv <- addCovariance(g.cv, type='omni')
155         }
156
```

```
157         ## prevent errors due to numerical instability of matrix inversion
158         prdct   <- function(g, a, b, c) { predictFstat(g,
                .newCoordinates=a, .what=b, .type=c) }
159         prdct2  <- purrr::possibly(prdct, otherwise = NA)
160         ck.pred <- prdct2(g.cv, coord_train[idx[f.idx], ], "Fine", "UcoK")
161
162         ## compute prediction error
163         if( any(is.na(ck.pred)) ) {
164           tmp[f] <- NA
165         } else {
166           ## build basis representation to compute norms
167           sse.cv        <- NULL
168           normconst.tmp <- NULL
169           bspbasis      <- create.bspline.basis(range(tstep), nbasis=30)
170           real          <- Data2fd(argval=tstep, y=curves.l1[
                  ,idx[f.idx]], basisobj=bspbasis)
171           pred          <- Data2fd(argval=tstep, y=ck.pred$Forecast,
                  basisobj=bspbasis)
172
173           if( length(f.idx) == 1 ) {
174             ## one curve per fold, no need to normalize
175             tmp[f] <- fda.usc::norm.fd(real - pred)^2
176           } else {
177             ## compute mean function for SSE normalization
178             real.mean <- Data2fd(argval=tstep, y=rowMeans(curves.l1[
                    ,idx[f.idx]]), basisobj=bspbasis)
179             for( j in 1:ncol(curves.l1[ ,idx[f.idx]]) ) {
180               sse.cv[j]        <- fda.usc::norm.fd(real[j] - pred[j])^2
181               normconst.tmp[j] <- fda.usc::norm.fd(real[j] - real.mean)^2
182             }
183
184             ## normalize SSE
185             normconst.cv <- sum(normconst.tmp)/ncol(curves.l1[
                    ,idx[f.idx]])
186             sse.norm.cv  <- sse.cv/normconst.cv
187
188             ## compute mean SSE within fold
189             tmp[f]       <- mean(sse.norm.cv)
190           }
191         }
192       }
193
194       ## compute mean SSE between folds
195       res[i] <- mean(tmp, na.rm=TRUE)
196       cat(sprintf('r = %.2f | value = %.7f (NA = %d)\n', r, res[i],
              sum(is.na(tmp))))
197
198     }
199
200     ## select argmin of sse vector
201     range.best <- ranges[which.min(res)]
202     cat(sprintf('chosen r = %.2f \n', range.best))
203
204   }
205
206   return(range.best)
207
208 }
```

A.4 TestPerformances

`TestPerformances()` function encloses the whole performance testing procedure, allowing to perform prediction for increasing amounts of hi-fi data with either Universal Trace-Kriging or Universal Trace-Cokriging. Also, it is possible to select the desired variogram fitting routine between cross-validation and maximum likelihood.

```
TestPerformances <- function(tstep, coord.train, coord.test, curves.l1,
    curves.l2=NULL, curves.test, idx=NULL, drift.frml='~ .', method='CV',
    model.name='Sph', n.lags=NULL, lag.max=NULL, seed=NULL,
    filename=ifelse(is.null(curves.l2, 'UK', 'UCoK'))) {

  ## exit if required packages are missing, otherwise load them
  if(!require(fda)) stop()
  if(!require(fda.usc)) stop()
  if(!require(fdagstat)) stop()
  if(!require(purrr)) stop()
  if(!require(caret)) stop()

  ## check if seed is provided
  if( is.null(seed) ) stop('Sampling strategy requires a seed. Exiting!')

  ## if no list of indexes is provided, create one
  if( is.null(idx) ) {

    idx <- list()
    for( i in 1:10 ){
      if( i < 10 ) {
        ## select a subset of input indexes
        set.seed(seed)
        nsample  <- i*0.1
        if( i == 1 ) {
          ## no previous indexes selected
          past.idx <- NULL
        } else {
          ## store indexes from previous run
          past.idx <- idx[[i-1]]
        }
        ## conditioned LHS for new points, previous ones are included
        idx[[i]] <- clhs::clhs(coord.train, nrow(coord.train)*nsample,
            include=past.idx, progress=FALSE)
        idx[[i]] <- sort(idx[[i]])
      }else{
        ## all indexes are selected
        idx[[i]] <- 1:nrow(coord.train)
      }
    }
  }

  g         <- list()
  pred.all  <- list()
  sse       <- list()
  normconst <- list()
  sse.norm  <- list()

  out.file  <- paste0(output.file, format(Sys.time(), "%y%m%d-%H%M%S"),
      '.txt')
```

```
46
47  for(i in 1:10) {
48
49    if( i == 1) cat(paste0('-- Building', out.file, ' predictor --\n' )
50    cat('\nInitializing block ', i, '/', 10, '\n')
51    cat(file=out.file, '\nInitializing block ', i, '/', 10, ' ',
          append=TRUE)
52
53    ## building gstat object
54    g[[i]] <- fstat(NULL, vName="Fine", Coordinates=coord.train[idx[[i]],
          ], Functions=as.data.frame(curves.l1[ ,idx[[i]]]), scalar=FALSE)
55
56    if(!is.null(curves.l2)) {
57      ## add secondary variable if provided
58      g[[i]] <- fstat(g[[i]], vName="Coarse", Coordinates=coord.train,
            Functions=as.data.frame(curves.l2), scalar=FALSE)
59    }
60
61    ## OLS estimate of drift component
62    frml  <- drift.frml
63    g[[i]] <- estimateDrift(frml, g[[i]], Intercept=TRUE)
64
65    if( method == 'CV' ) {
66
67      ## range optimization with cross-validation
68      range.best <- FitRangeCV(coord.train, idx[[i]], tstep, curves.l1,
            curves.l2, drift.frml, model.name='Sph', seed=seed+1)
69
70      ## add valid model
71      g[[i]] <- fvariogram("~.", g[[i]], Nlags=n.lags, LagMax=lag.max,
            ArgStep=mean(tstep), useResidual=TRUE, comments=FALSE)
72      g[[i]] <- fitVariograms(g[[i]], model=vgm(psill=1, model=model.name,
            range=range.best), fitSills=FALSE, fitRanges=FALSE)
73      g[[i]] <- addCovariance(g[[i]], type='omni')
74
75      ## iterate GLS drift estimate and residuals computation
76      for ( j in 1:5 ) {
77        g[[i]] <- estimateDrift(frml, g[[i]], .type="GLS", Intercept=TRUE)
78        g[[i]] <- fvariogram("~.", g[[i]], Nlags=n.lags, LagMax=lag.max,
              ArgStep=mean(tstep), useResidual=TRUE, comments=FALSE)
79        g[[i]] <- fitVariograms(g[[i]], model=vgm(psill=1,
              model=model.name, range=range.best), fitSills=FALSE,
              fitRanges=FALSE)
80        if( j == 5 ) {
81          ## rescale sill(s)
82          mm <- max(sapply(g[[i]]$model$omni, function(el) el$psill ))
83          g[[i]]$model$omni$Fine$psill <- g[[i]]$model$omni$Fine$psill / mm
84          if( !(is.null(curves.l2)) ) {
85            g[[i]]$model$omni$Coarse$psill      <-
                g[[i]]$model$omni$Coarse$psill / mm
86            g[[i]]$model$omni$Fine.Coarse$psill <-
                g[[i]]$model$omni$Fine.Coarse$psill / mm
87          }
88        }
89        g[[i]] <- addCovariance(g[[i]], type='omni')
90      }
91    } else if (method == 'ML') {
92      ## ML variogram fitting, with default range value
93      range.best <- NA
94      ## add valid model
95      g[[i]] <- fvariogram("~.", g[[i]], Nlags=n.lags, LagMax=lag.max,
```

```
          ArgStep=mean(tstep), useResidual=TRUE, comments=FALSE)
96      g[[i]] <- fitVariograms(g[[i]], model=vgm(psill=1, model=model.name,
          range=range.best), fitSills=FALSE, fitRanges=TRUE)
97      g[[i]] <- addCovariance(g[[i]], type='omni')
98
99      ## iterate GLS drift estimate and residuals computation
100     for ( j in 1:5 ) {
101       g[[i]] <- estimateDrift(frml, g[[i]], .type="GLS", Intercept=TRUE)
102       g[[i]] <- fvariogram("~.", g[[i]], Nlags=n.lags, LagMax=lag.max,
            ArgStep=mean(tstep), useResidual=TRUE, comments=FALSE)
103       g[[i]] <- fitVariograms(g[[i]], model=vgm(psill=1,
            model=model.name, range=range.best), fitSills=FALSE,
            fitRanges=TRUE)
104       if( j == 5 ) {
105         # rescale sill(s)
106         mm <- max(sapply(g[[i]]$model$omni, function(el) el$psill ))
107         g[[i]]$model$omni$Fine$psill <- g[[i]]$model$omni$Fine$psill / mm
108         if( !(is.null(curves.l2)) ) {
109           g[[i]]$model$omni$Coarse$psill      <-
                g[[i]]$model$omni$Coarse$psill / mm
110           g[[i]]$model$omni$Fine.Coarse$psill <-
                g[[i]]$model$omni$Fine.Coarse$psill / mm
111         }
112       }
113       g[[i]] <- addCovariance(g[[i]], type='omni')
114     }
115   }
116
117   ## prediction of test set curves
118   pred.all[[i]] <- predictFstat(g[[i]], .newCoordinates=coord.test,
        .what="Fine", .type=ifelse(is.null(curves.l2, 'UK', 'UcoK')))
119
120   ## build basis representation
121   sse[[i]] <- vector()
122   bspbasis <- create.bspline.basis(range(tstep), nbasis=30)
123   real     <- Data2fd(argval=tstep, y=curves.test, basisobj=bspbasis)
124   pred     <- Data2fd(argval=tstep, y=pred.all[[i]]$Forecast,
        basisobj=bspbasis)
125
126   ## compute prediction error
127   for(j in 1:ncol(curves.test)) {
128     sse[[i]][j] <- fda.usc::norm.fd(real[j] - pred[j])^2
129   }
130
131   ## compute mean function for SSE normalization
132   normconst.tmp <- NULL
133   real.mean     <- Data2fd(argval=tstep, y=rowMeans(curves.test),
        basisobj=bspbasis)
134   for(j in 1:ncol(curves.test)) {
135     normconst.tmp[j] <- fda.usc::norm.fd(obs[j]-obs.mean)^2
136   }
137   cat('.')
138
139   ## normalize SSE
140   normconst[[i]] <- sum(normconst.tmp)/ncol(curves.test)
141   sse.norm[[i]]  <- sse[[i]]/normconst[[i]]
142
143   ## print outputs both to file and console
144   cat(file=out.file, ' done\n', append=TRUE)
145   cat(file=out.file, '\n', append=TRUE)
146   sink(out.file, append=TRUE); print(g[[i]]$model); sink()
```

```
147     cat(file=out.file, '> minimum normalized SSE:', min(sse.norm[[i]]) %>%
            log10(), ' (log10)\n', append=TRUE)
148     cat(file=out.file, '> mean normalized SSE:   ', mean(sse.norm[[i]])
            %>% log10(), ' (log10)\n', append=TRUE)
149     cat(file=out.file, '> median normalized SSE: ', median(sse.norm[[i]])
            %>% log10(), ' (log10)\n', append=TRUE)
150
151     cat(' done\n')
152     cat('> minimum normalized SSE:', min(sse.norm[[i]]) %>% log10(), '
            (log10)\n')
153     cat('> mean normalized SSE:   ', mean(sse.norm[[i]]) %>% log10(), '
            (log10)\n')
154     cat('> median normalized SSE: ', median(sse.norm[[i]]) %>% log10(), '
            (log10)\n')
155   }
156
157   final <- list(model=g, idx=idx, pred=pred.all, norm.const=normconst,
          sse.norm=sse.norm)
158
159   return(final)
160
161 }
```