POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell'Informazione

Laurea Magistrale in Mathematical Engineering

Random Domain Decompositions for object-oriented Kriging over complex domains

Supervisor: Prof. Piercesare SECCHI

Co-supervisor: Dott.ssa Alessandra MENAFOGLIO

Tesi di Laurea Magistrale di:

Giorgia GAETANI

Matr. 836677

Academic year 2015-2016

“When you look, look wide;
and even when you think you are looking wide,
look wider still.”

B. Powell

Contents

List of Figures

List of Tables

Ringraziamenti

Oggi inizia un nuovo capitolo della mia vita. Ci saranno nuove esperienze, nuovi luoghi, nuove responsabilità. Spero che questo nuovo percorso sia ricco di soddisfazioni, ma mi auguro, soprattutto, di poter avere accanto le persone che mi hanno accompagnato fino a questo traguardo.
Un primo ringraziamento è per coloro che mi hanno dato la possibilità di lavorare su questo progetto di Tesi. Grazie al professor Secchi per la passione nella Statistica che è riuscito a trasmettermi con i suoi insegnamenti. Grazie per l'entusiasmo e l'umiltà con le quali mi ha guidato in questi mesi. Grazie Alessandra, per la tua disponibilità, la tua dolcezza e la tua passione. Sei la dimostrazione di come con impegno e dedizione si possano raggiungere i propri obiettivi.
Grazie Mamma, per aver ascoltato le mie parole e per aver, più spesso, interpretato i miei silenzi. Grazie per i consigli, la fiducia, l'amore e la determinazione che solo tu sai trasmettermi. Grazie Papà, perchè credi in me più di ogni altra persona. Grazie per gli abbracci, per le parole dette e non. Grazie per l'emozione che provi, ma che spesso nascondi, di fronte ad ogni mio successo. Grazie Giacomo, per essere un punto di riferimento, una fonte di forza, positività e voglia di mettersi in gioco. Sei il mio opposto di cui avrò sempre bisogno.
Grazie a nonno Checco e nonna Tetta, ai loro abbracci che tolgono il fiato, ai loro occhi lucidi ogni volta che c'è un treno direzione Milano ad aspettarmi, alle loro mani che mi hanno donato le carezze più dolci. Grazie a voi, che più di tutti avete sofferto la mia lontanza. Grazie a nonno Antonio e nonna Rita, che la vita, troppo presto, ha portato via da me. Grazie ai miei zii, per il loro amore e le loro attenzioni. Grazie perchè mi avete sempre trattato come una “principessa”, ma soprattutto, grazie per avermi donato i miei meravigliosi cugini che, anche se non ho potuto vedere crescere, amo come se fossero miei fratelli.
Grazie alle mie amiche Elisa, Nicoletta, Graziana e Alessandra perché mi hanno insegnato che la distanza non è in grado di separare le persone. Grazie alla mia “famiglia svedese” Bea, Luisa, Tifaine. Grazie per i vostri sorrisi che sono stati un punto di riferimento nei mesi trascorsi lontano dal mio mondo.
Infine, grazie a lui, la mia fonte di felicità. Grazie per avermi insegnato ad amare, per aver vissuto insieme a me ogni istante della mia vita, per aver asciugato le mie lacrime e gioito di ogni mio successo. Grazie per le emozioni che mi regali da otto anni, ma soprattutto grazie perché so che lo farai per sempre.

Abstract

This dissertation introduces a new methodology for the analysis of spatial fields of object data (such as scalar, functional or constrained data) possibly distributed over complex domains. We combine the approaches of Object Oriented Data Analysis and classical geostatistic in the framework of Object Oriented Spatial Statistics. The method we propose enables one to jointly handle both data and domain complexities (e.g., very large size or geographical constraints), through a non-parametric approach which allows avoiding strong distributional assumptions. As a key element of innovation, we propose to perform repeated Random Domain Decompositions (RDDs), each defining a set of homogeneous sub-regions where to perform simple, independent weak local analyses, under stationarity assumption. The latter local analyses are then aggregated into a final strong analysis. In this broad framework, the possible complexity of the domain can be taken into account by defining its partitions on the basis of a non-Euclidean metric, which allows to properly represent the adjacency relationships among the data over the domain. To account for the domain complexity, we propose to use a metric based on a graph, which is defined by a triangulation of the domain (e.g., Delaunay triangulation).
As an insightful illustration of the potential of the methodology, we consider the analysis and spatial prediction (Kriging) of data distributed over estuaries. An estuary is a non-convex and very irregularly shaped region where the narrow areas of land between adjacent tributaries act as barriers for many aquatic variables. Here, the method we propose is used for the study of the dissolved oxygen depletion problem in the Chesapeake Bay, an important estuary in the United States.

Keywords: non-stationarity, object-oriented spatial statistics, variogram, Kriging, kernel function, Delaunay triangulation, non-Euclidean metric, estuarine system.

Sommario

Questa tesi introduce una nuova metodologia per l'analisi di campi spaziali di dati-oggetto (e.g., dati funzionali o vincolati) distribuiti su domini complessi. Il presente lavoro combina gli approcci della Analisi di Dati Orientata agli Oggetti (*Object Oriented Data Analysis*) con quelli tipici della Geostatistica, nel contesto della Statistica Spaziale Orientata agli Oggetti (*Object Oriented Spatial Statistics*). Il metodo proposto permette di gestire contemporaneamente eventuali complessità del dato e del dominio di osservazione, attraverso un approccio non parametrico che consente di evitare la formulazione di forti assunzioni distribuzionali. Come elemento chiave di innovazione, è qui proposto un metodo basato su partizioni aleatorie del dominio (*Random Domain Decompositions*, RDDs). Ogni realizzazione di tali partizioni definisce un insieme di regioni omogenee all'interno delle quali è possibile eseguire analisi semplici e indipendenti, fondate sull'ipotesi di stazionarietà locale del fenomeno. Tali analisi locali sono poi aggregate con lo scopo di fornire un risultato globale. In questo contesto generale, le possibili complessità del dominio possono essere incluse nello studio attraverso la definizione delle RDDs tramite metriche non Euclidee, che consentono di rappresentare opportunamente la relazione di adiacenza tra i dati. Per tener conto della complessità del dominio, è qui proposto l'uso di una metrica basata su un grafo, definito attraverso una opportuna triangolazione del dominio (e.g., triangolazione di Delaunay).
Come illustrazione delle potenzialità della metodologia è qui considerata l'analisi e la previsione spaziale (*Kriging*) di dati distribuiti su domini complessi come quelli degli estuari. Un estuario è una regione non convessa dai contorni irregolari dove, spesso, diversi affluenti confluiscono in un corpo centrale. In questi casi, le aree di suolo che separano affluenti vicini rappresentano delle barriere per molte variabili acquatiche. Il metodo proposto è qui applicato allo studio del livello di ossigeno dissolto nelle acque dell'estuario più grande degli Stati Uniti: la Baia di Chesapeake.

Parole chiave: non stazionarietà, Statistica Spaziale Orientata agli Oggetti, variogramma, Kriging, funzione kernel, triangolazione di Delaunay, metrica non Euclidea, sistema estuarino.

Introduction

In recent years, the increasing availability of complex data led to the growth of *Object Oriented Data Analysis* (OODA, Marron and Alonso [2014]) which is a novel frontier of applied statistics focusing on the study of object data, such as high- or infinite-dimensional data or constrained data. The key idea upon which this framework is grounded is that the *atom* of the analysis is the whole complex datum, which is represented as a point within a mathematical space (e.g., a Hilbert space) reflecting all its fundamental properties. The datum is analysed with all its attributes and complexities. In some cases, spatial dependence may exist among these object data: this is the focus of *Object Oriented Spatial Statistics* (O2S2, Menafoglio and Secchi [2016]). O2S2 combines the approaches of OODA with classical geostatistical methods, used to take into account the spatial dependence among the object data. The description of the spatial dependence may be based, e.g., on the definition of the variogram (or covariogram) function, which can then be used to make predictions at unsampled locations. A common assumption regards the stationarity of the process, i.e., the homogeneity of its distributional properties over the whole domain. In many real spatial processes the covariance structure may vary over different areas of the study region or, in other cases, the characteristics of the domain may lead to the actual impossibility of defining a global stationary model for describing the spatial dependence.
Several approaches exist to handle non-stationary spatial fields by using local models. Each local model describes the spatial dependence of the random field within a neighborhood in the domain, where it is possible to assume stationarity. All these methods often require strong assumptions on the distribution which generated the data, which are rarely met in application with object data.
In this dissertation we propose a new family of methodologies for the analysis of object data, within the O2S2 framework, prone to account for the possible complexity associated with the geographical characteristics of the domain. A complex domain may consist in a very large domain or in a region with natural or artificial constraints, such as holes, barriers, irregular boundaries. Note that the complexity of the domain may even hinder the formulation of stationarity. In fact, within a non-convex domain one can hardly assume homogeneity in the structure of spatial dependence. To define the spatial dependence between two observations one needs to know how far two points are from each other, but also in which part of

the domain they are located, as well as if a barrier exists between them. Moreover, the presence of obstacles may require the use of a non-Euclidean metric as measure of the adjacency relationships among observations, which turns into theoretical and operational problems in most classical geostatistics techniques (e.g., variogram modeling).
The basic idea of the method we propose is to develop the spatial analysis of object data by using simple, local and repeated analyses instead of an unique global and complex analysis. Indeed, our computationally intensive yet simple methodology is based on a *divide et impera* idea. During the *divide* step the domain is randomly partitioned into several sub-regions, within which local geostatistical analyses are performed. These local and *weak* analyses are repeated for different realizations of the random domain decomposition and then, during the *impera* step, they are aggregated into a final *strong* analysis.
Although this dissertation focuses on the application of this new idea with the aim of performing linear spatial prediction (i.e., Kriging), an important property of this novel class of methodologies relies in its generality. Indeed, general geostatistical problems (e.g., classification or smoothing problems) may be successfully tackled in this broad framework.
The non-parametric nature of this approach is open to handle complex object data, while the complexity of the domain may be taken into account in the definition of the metric upon which the partitions of the domain are based. For example, within a domain with barriers it is possible to use a graph-based metric: the distance between two points is then computed as the length of shortest paths from one point to the other on a given undirected graph representing the real closeness relation among the data.
The method we propose is entirely general, and suitable to be applied in a wide range of real environmental studies. Here, we illustrate the application of the proposed methodology to the analysis of spatial data observed within an estuary. Indeed, an estuarine system develops on a complex, non-convex and highly irregular domain where the areas of land between adjacent tributaries act as barriers for many aquatic variables. Therefore, the use of the Euclidean distance is somehow inappropriate for describing the adjacency relation between observations. Moreover, the data collected within this particular region may be complex (e.g., functional data). The proposed methodology is able to handle jointly both data and domain complexities, providing more accurate predictions with respect to the existing methods, which are able to treat at most one of these two complexities.

The remaining part of this dissertation is organized as follows.

Chapter 1: *Geostatistics.* This chapter recalls the basic notions on Geostatistics for real-valued random fields based on Cressie [1993] and Chilès and Delfiner [1999]. Here, we introduce the key concepts, i.e., the definition of stationarity, the variogram and covariogram functions, used to define the spatial dependence among data, and the Kriging predictor.

Chapter 2: *Non-stationary Spatial Data.* This chapter overviews current approaches to handle non-stationary spatial processes and highlights the reasons why they would be inappropriate for the analysis of object data.

Chapter 3: *Random Domain Decompositions for O2S2.* This chapter provides a more detailed description of the O2S2 framework and introduces possible applications of the new method for spatial statistics analysis. In particular, the chapter introduces the first original contribution of the dissertation, describing the key idea of the proposed methodology: the *Random Domain Decompositions* (RDDs). The need of performing local analyses, without requiring strong distributional assumptions, motivates us to use random partitions of the domain. Following a *bagging* procedure, the same spatial analysis is performed several times and then the bootstrap replicates are aggregated into a final analysis. The difference between two bootstrap iterations regards the realization of the RDD, which defines a system of homogeneous neighborhoods. Therefore, within each neighborhood, a local and simple analysis can be performed. The use of a geographically weighted statistics viewpoint gives the possibility to consider information from neighbour sub-regions for the estimate of the structure of spatial dependence within each neighborhood.

Chapter 4: *Kriging-RDDs: the case of regular domain.* The chapter provides an in-depth description of the RDDs idea applied in the context of Kriging predictions of spatial random fields distributed over simple domains. Section 4.2 shows the results of the method applied to two simulated examples (i.e., a simple stationary Gaussian process and a non-stationary process, both distributed over a square domain). Moreover, using these simulated examples, we investigate the performance of the method with respect to the model parameters (e.g., the number of sub-regions).

Chapter 5: *Kriging-RDDs applied to irregularly shaped domain.* The chapter illustrates the difficulties which can occur when one has to analyse spatial fields distributed over a complex domain, where a non-Euclidean metric is needed. This chapter describes how the proposed method may account for the complex properties of the domain. In particular, as a key element of innovation, we here propose to map the sampled points into a *neighborhood relational graph* representing their spatial adjacency and to use a graph-based metric (i.e., the length of the shortest path) as distance for the definition of the domain partitions. Finally, the chapter describes the application of the proposed methodology to two simulated datasets distributed over complex regions: a domain with a hole within Section 5.5.1, and a C-shaped domain, Section 5.5.2.

Chapter 6: *Case Study: Chesapeake Bay.* The chapter describes the application of the method for the Kriging prediction of the Dissolved Oxygen values in the Chesapeake Bay, the biggest and most productive estuary in the United States.

Appendix A: `R` *codes.* The appendix includes the codes implemented in `R` [R Development Core Team, 2008] of the main functions defining all the steps of the RDDs methodology employed to perform Kriging prediction of spatially distributed random fields.

Chapter 1

Geostatistics

"Geostatistics is the application of probabilistic methods to regionalized variables"

G.Matheron

The definition of the founder of *Geostatistics*, the French mathematician Georges François Paul Marie Matheron, summarizes the role of this branch of statistics: it focuses on the analysis of spatial or spatiotemporal data, which are observed at different locations within a spatial domain. Geostatistics deviates from the the classical statistical framework due to the need to release the independence assumption among the samples. Indeed, most of the Earth science data often do not satisfy this independence assumption, since they are spatially correlated. A geostatistical analysis is then based on the intuitive idea that data from locations that are closer together, in time or space, are likely to be correlated, therefore they tend to be more similar than data from locations that are far away. If classical statistics aim to examine the statistical distribution of a sample of data, Geostatistics focuses on both the statistical distribution of the sampled data and the structure of spatial correlation underlying them.
The fundamental goal of Geostatistics is often to predict the spatial distribution of a random process at unsampled locations, by using information coming from the observed data within the study region. This estimation is based not only on the sample data, but also on a model (variogram), which represents the spatial correlation of the observed data. The estimate at unsampled locations can be performed by applying the Kriging technique. In the geostatistical setting, the Kriging approach is widely preferred to the classical interpolation method. Indeed, with a simple interpolation approach the predictor is defined as linear combination of the observed data with weights that may depend on some properties (e.g. inverse distance or inverse distance squared), but they do not incorporate information on spatial correlation. On the other hand, the Kriging predictor considers both the distance and the spatial correlation in the definition of the weights for the observed data.

In the geostatistical framework, spatial data can be thought as observations of a random field

$$\{Z_{\mathbf{s}}, \mathbf{s} \in D\} \tag{1.1}$$

where $D \subseteq \mathbb{R}^p$ (usually $p = 2, 3$) represents a fixed domain. The data $\{Z_{\mathbf{s}_1}, \ldots, Z_{\mathbf{s}_n}\}$ are partial observations of this field at n locations $\{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$ in D. The characteristics of the process (1.1) and the properties of the domain D give the possibility to distinguish three prototypes of spatial data [Cressie, 1993]: (a) Geostatistical data, (b) Lattice Data, and (c) Point patterns.

Geostatistical data. In this setting the data are observations of the variable of interest $Z_{\mathbf{s}}$ (or a vector of variables) over a fixed spatial region D. The measurement points $\{\mathbf{s}, \mathbf{s} \in D\}$ vary continuously over the study region. The analysis of these data is aimed to recognize spatial variability at both the large scale and the small scale. This means that a model for both spatial trend and spatial correlation must be defined. The model for the spatial variability is used to perform predictions at unsampled locations.

Lattice data. In this setting, the study region D is a fixed, countable, regular or irregular, collection of spatial sites where observations are made. The domain is described by its partition into sub-regions (such as counties or provinces). The collection of points in D is known as a *lattice*. Also in this case $Z_{\mathbf{s}}$ is a random variable (or a vector of variables) observed at the location $\mathbf{s}$. The analyses of lattice data often involve estimation or classification problems.

Point patterns. Contrarily to the other two cases, here D is a random domain. Point patterns arise when the aim is to analyse locations of *events*. The question of interest is whether the pattern is exhibiting complete spatial randomness, clustering or regularity. The random variable $Z_{\mathbf{s}}$ may be observed at location $\mathbf{s} \in D$ (in this case we call (1.1) *marked process*) or not ((1.1) is called *standard point process*).

Our focus is on the analysis of geostatistical data. In particular, we will mostly assume that (1.1) is a real-valued random field distributed over an Euclidean space. Nevertheless, all the geostatistical methodologies here considered can be extended to more complex frameworks, for example to spatially dependent functional data in Hilbert spaces [Menafoglio et al., 2013].

1.1 Geostatistics for real-valued random fields

In the standard geostatistical framework, the random variable $Z_{\mathbf{s}}$ at location $\mathbf{s}$ is modeled as the combination of a global trend $m_{\mathbf{s}}$ and the element $\delta_{\mathbf{s}}$ of a stochastic process:

$$Z_{\mathbf{s}} = m_{\mathbf{s}} + \delta_{\mathbf{s}} \quad \mathbf{s} \in D. \tag{1.2}$$

Here, the mean function $m_{\mathbf{s}}$, called *drift*, is assumed to be deterministic while the residual $\delta_{\mathbf{s}}$ represents the stochastic term. This residual part defines the random properties of the field and it is characterized by the structure of spatial dependence that we want to investigate. For any set of locations $\{\mathbf{s}_1, \dots, \mathbf{s}_n\}$, under the Kolmogorov homogeneity assumption [Cressie, 1993], the joint distribution of the random vector $\mathbf{Z} = (Z_{\mathbf{s}_1}, \dots, Z_{\mathbf{s}_n})^T$, named *finite-dimensional law*, is defined as

$$F_{\mathbf{s}_1,\dots,\mathbf{s}_n}(z_1, \dots, z_n) = \mathbb{P}\{Z_{\mathbf{s}_1} \leq z_1, \dots, Z_{\mathbf{s}_n} \leq z_n\} \tag{1.3}$$

with $z_1, \dots, z_n \in \mathbb{R}$. The family $\{F_{\mathbf{s}_1,\dots,\mathbf{s}_n}, \mathbf{s}_1, \dots, \mathbf{s}_n \in D\}$ completely characterizes the distribution of process (1.1).

1.1.1 Stationary processes

In principle, a Geostatistical analysis could be performed independently of any hypothesis on the nature of the random field (1.1). However, in order to achieve acceptable estimating results without requiring an amount of data much larger than what is usually available, some restrictions on the nature of the random field are essential. Simplifying assumptions are often made on the spatial dependence structure. The most important is the *stationary* assumption. There are several definitions of stationarity:

Definition 1.1. The random process $\{Z_{\mathbf{s}}, \mathbf{s} \in D\}$ is said *strongly stationary* if $(Z_{\mathbf{s}_1}, \dots, Z_{\mathbf{s}_n})^T$ and $(Z_{\mathbf{s}_1+\mathbf{h}}, \dots, Z_{\mathbf{s}_n+\mathbf{h}})^T$ have the same joint distribution for all $\mathbf{h} \in \mathbb{R}^p$, $n \geq 1$ and for any collection $\{\mathbf{s}_i\}_{i=1}^n$ in D.

This property can be summarized by saying that the finite dimensional distributions of a stationary field are translation invariant.
Since the strong stationarity assumption is difficult to be verified in practise, and it is an uncommon property for random fields commonly observed in real applications, a weakened definition is required:

Definition 1.2. The random process $\{Z_{\mathbf{s}}, \mathbf{s} \in D\}$ is defined to be *second-order* (or *weak*) *stationary* if the following conditions are satisfied:

(i) $\mathbb{E}[Z_{\mathbf{s}}] = m$, for all $\mathbf{s} \in D$;

(ii) $\mathrm{Cov}(Z_{\mathbf{s}_i}, Z_{\mathbf{s}_j}) = \mathbb{E}[(Z_{\mathbf{s}_i} - m)(Z_{\mathbf{s}_j} - m)] = C(\mathbf{h})$, for all $\mathbf{s}_i, \mathbf{s}_j \in D$, such that $\mathbf{h} = \mathbf{s}_i - \mathbf{s}_j$.

In general, the class of second-order stationary random fields is much larger than that of strong stationary random fields. Furthermore, a strong stationary field is also second-order stationary, whenever each element of the random process

satisfies the finite second moment assumption (i.e. $\mathbb{E}[Z_{\mathbf{s}}^2] < \infty$ for all $\mathbf{s}$ in D). The reverse implication generally does not hold, except for the case when the process is Gaussian (in this case the two definitions coincide).
The function $C(\cdot)$ appearing in the previous definition is called *covariogram* and it is characterized by the following properties:

(a) Positive definiteness:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j C(\mathbf{s}_i - \mathbf{s}_j) \geq 0 \quad \forall \lambda_i, \lambda_j \in \mathbb{R}, \mathbf{s}_i, \mathbf{s}_j \in D;$$

(b) Symmetry: $C(\mathbf{h}) = C(-\mathbf{h})$;

(c) Boundedness: $|C(\mathbf{h})| \leq C(\mathbf{0})$.

The second-order stationarity property is a common assumption for random fields, but in some cases a further weaker definition is needed.

Definition 1.3. The random process $\{Z_{\mathbf{s}}, \mathbf{s} \in D\}$ is said *intrinsically stationary* if:

(i) $\mathbb{E}[Z_{\mathbf{s}}] = m$, for all $\mathbf{s} \in D$;

(ii) $\mathrm{Var}(Z_{\mathbf{s}_i} - Z_{\mathbf{s}_j}) = \mathbb{E}[(Z_{\mathbf{s}_i} - Z_{\mathbf{s}_j})^2] = 2\gamma(\mathbf{h})$, for all $\mathbf{s}_i, \mathbf{s}_j$ in D, $\mathbf{h} = \mathbf{s}_i - \mathbf{s}_j$.

The function $\gamma(\cdot)$ is called *semivariogram*, while $2\gamma(\cdot)$ indicates the *variogram* function.
Under finite second moment assumption, we can observe that a second-order stationary process is also intrinsically stationary; in this case we can relate the covariogram and the semivariogram functions according to the following relation:

$$\gamma(\mathbf{h}) = C(\mathbf{0}) - C(\mathbf{h}), \qquad \mathbf{h} \in \mathbb{R}^d. \tag{1.4}$$

1.1.2 Variogram properties

The variogram is the function describing the second-order properties of a second-order (or intrinsic) stationary process. It quantifies the spatial correlation among the observations, giving a measures of dissimilarity among them. In order to analyse a spatially distributed dataset, the first step which must be performed is the estimate of the variogram. According to Cressie [1993] and Chilès and Delfiner [1999], a variogram must satisfy the following properties:

(a) Conditional negative definiteness:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j \gamma(\mathbf{s}_i - \mathbf{s}_j) \leq 0 \quad \forall \mathbf{s}_i, \mathbf{s}_j \in D;\ \lambda_i, \lambda_j \in \mathbb{R} \text{ s.t. } \sum_{i=1}^{n}\lambda_i = 0;$$

(b) Symmetry: $\gamma(\mathbf{h}) = \gamma(-\mathbf{h})$;

(c) Non-negativity: $\gamma(\mathbf{h}) \geq 0$;

(d) Zero at the origin: $\gamma(\mathbf{0}) = 0$;

(e) Sub-quadratic growth: $\lim_{\|\mathbf{h}\|\to\infty} \frac{2\gamma(\mathbf{h})}{\|\mathbf{h}\|^2} = 0$.

A semivariogram fulfilling the properties (a)-(e) is said to be a *valid* semivariogram.

Definition 1.4. An intrinsic stationary process $\{Z_\mathbf{s}, \mathbf{s} \in D\}$ is said *isotropic* if its variogram is isotropic:

$$\mathrm{Var}(Z_{\mathbf{s}_i} - Z_{\mathbf{s}_j}) = 2\gamma(h), \quad h = \|\mathbf{h}\| = \|\mathbf{s}_i - \mathbf{s}_j\|$$

for all $\mathbf{s}_i, \mathbf{s}_j$ in D.

Therefore an isotropic variogram only depends on the separation distance between two points. This distance is computed according to the Euclidean metric. The isotropic property characterizes a process with a covariance structure which is homogeneous over all the directions of $\mathbb{R}^p$. If this property is not satisfied the process is said *anisotropic.*
Assume that the variogram is isotropic. The structural properties of a valid semivariogram reflect the physical properties of the field that we want to analyse. Even if the semivariogram is null at the origin ((d)-property), here it may present a discontinuity, associated with a non-zero limit when h goes to zero:

$$\lim_{h\to 0} \gamma(h) = \tau^2 \neq 0.$$

In this case τ^2 is called *nugget.* Since an L^2-continuous process cannot have a discontinuous variogram, the nugget effect is associated with either a measurement error in the data or a discontinuity in the process. Measurement error is in general assumed to be spatially uncorrelated and should not affect the structure of the variogram for values of h different from zero. Another important property of the variogram is the *sill*, which indicates the value of the semivariogram asymptote. It is defined as:

$$\lim_{h\to\infty} \gamma(h) = \tau^2 + \sigma^2$$

where τ^2 is the nugget and σ^2 is called *partial sill.* The existence of this finite limit indicates that the process is second-order stationary and the variance of each random variable $Z_\mathbf{s}$ is defined as $C(0) = \tau^2 + \sigma^2$.
As last property of a valid semivariogram, we mention the *range* R:

$$\gamma(R) = \tau^2 + \sigma^2.$$

The range of a valid semivariogram is the point where it reaches the sill. It describes how fast the autocovariance decays with the distance, quantifying the range of influence of the process: if two points of the domain have a distance greater than R, then they are uncorrelated. However, the range of a variogram can be infinite. It happens when the sill does not exist (in this case the process is not stationary) or when the sill is reached asymptotically. In the latter case it is possible to define a *practical range* $\tilde{R}$, such that $\gamma(\tilde{R}) = 0.95(\tau^2 + \sigma^2)$.

1.1.3 Variogram estimation

Most methods for variogram estimate consist of three stages: (a) compute a raw estimate from the data, which may lead to a non valid model; (b) based on graphical tools, choose a theoretical model among the family of valid variograms; finally (c) fitting of a parametric valid model via least square (LS) or maximum likelihood (ML) methods.

Empirical variogram

Given n observations $\mathbf{Z} = (Z_{\mathbf{s}_1}, \dots, Z_{\mathbf{s}_n})^T$ of an intrinsic stationary and isotropic random process, an empirical estimate of the semivariogram can be obtained via method of moments, according to:

$$\hat{\gamma}(h) = \frac{1}{2|N(h)|} \sum_{(i,j) \in N(h)} [Z_{\mathbf{s}_i} - Z_{\mathbf{s}_j}]^2 \tag{1.5}$$

where $N(h) = \{(i,j) : \|\mathbf{s}_i - \mathbf{s}_j\| = h\}$ and $|N(h)|$ indicates its cardinality. Since the estimate is based on the data, we observe only a finite number of distances h (the distances between each pair of sampled locations). Therefore a discretized version of (1.5) is generally used for the estimate. Firstly, the semivariogram cloud is computed: its plot displays the individual point-pair $\frac{1}{2}[Z_{\mathbf{s}_i} - Z_{\mathbf{s}_j}]^2$ contributions to the final variogram. This cloud of points gives important information regarding the number of couples available for the variogram estimation and the spread of the squared differences (the cloud's shape may help in getting a visual impression of the appropriateness of the stationarity assumption and it helps the selection of a valid variogram).
Then, a discretized version of (1.5) is defined

$$\hat{\gamma}(\mathbf{h}) = (\hat{\gamma}(h_1), \dots, \hat{\gamma}(h_M))^T.$$

Here, M classes of distances are identified and $\hat{\gamma}(h_m)$ indicates the average of the estimator (1.5) within the m-th class, for every $m = 1, \dots, M$. Under the assumption of intrinsic stationarity the estimator defined by (1.5), or its discretized version, is an unbiased estimate of $\gamma(\cdot)$. If the mean function is not constant over the whole domain, estimator (1.5) defines an unbiased estimate of

$$\mathbb{E}[(Z_{\mathbf{s}} - Z_{\mathbf{s}+\mathbf{h}})^2] = 2\gamma(\|\mathbf{h}\|) + (m_{\mathbf{s}} - m_{\mathbf{s}+\mathbf{h}})^2$$

which does not represent the process variogram and, in general, it does not satisfy the sub-quadratic growth property of a valid variogram model.
Once the empirical variogram has been estimated, we need to fit a valid model. Indeed, the empirical variogram typically does not produce a valid model, since some of the properties described before are not fulfilled. To guarantee that in all the cases the statistical analysis of a spatially distributed dataset is based on a valid variogram, a number of families of parametric valid models was defined.

Valid variogram models

Within the family of valid variograms, a function is selected. This function must accurately represent the underlying spatial dependence of the available data. The final estimated variogram will be obtained by fitting this selected theoretical model to the empirical estimation obtained from the data.
Among the valid models, the most employed ones are the following:

- *Pure Nugget*
$$\gamma(h) = \begin{cases} \tau^2 & h > 0 \\ 0 & h = 0. \end{cases} \tag{1.6}$$
The random process with a pure nugget variogram is a white noise with variance τ^2.

- *Linear model*
$$\gamma(h) = \begin{cases} a^2 h & h > 0 \\ 0 & h = 0 \end{cases} \tag{1.7}$$
with $a \in \mathbb{R}$. In this case range and sill are infinite. The associated process is intrinsically stationary, but non-stationary.

- *Exponential model*
$$\gamma(h) = \begin{cases} \sigma^2(1 - e^{-h/a}) & h > 0 \\ 0 & h = 0. \end{cases} \tag{1.8}$$
In this case σ^2 is the sill, the range is infinite, but we can compute the practical range $\tilde{R} = 3a$.

- *Spherical model*
$$\gamma(h) = \begin{cases} \sigma^2[\frac{3}{2}\frac{h}{a} - \frac{1}{2}(\frac{h}{a})^3] & h \leq a \\ \sigma^2 & h > a \\ 0 & h = 0. \end{cases} \tag{1.9}$$
In this case a is the range and σ^2 is the sill.

There are no universal algorithms in order to define the best variogram model. The most common methods used to choose a valid family are based on graphical tools. From the graphical representation of the empirical variogram we can detect some properties of the most suitable theoretical model. These properties (such as the behaviour close to the origin, the existence of a finite or infinite range) may be used to restrict the set of valid variogram models useful for our analysis. One of the most important structural properties, that can be detected from the empirical variogram, is the behaviour of the variogram close to the origin. The discontinuity at the origin is associated with a highly irregular and L^2- discontinuous random process and it is due to measurement errors. The linear behaviour at the origin (linear, exponential and spherical models) is associated with continuous but non-differentiable random processes. A quadratic behaviour at the origin is associated with regular random fields with smooth realizations.

Fitting step

A valid variogram model is characterized by a set of unknown parameters which must be estimated. For example, the valid models previously described (i.e., pure nugget, linear, exponential and spherical models) are completely identified by the three parameters $\boldsymbol{\vartheta} = (\tau^2, \sigma^2, R)$. The fitting step concerns the estimate of these parameters. We can follow a parametric approach by using the maximum likelihood method, or a more popular non-parametric approach like a least square criterion. We can choose among three LS criteria: ordinary least square (OLS), generalized least square (GLS) and weighted least square (WLS). The OLS variogram fitting consists of looking for the parameters $\boldsymbol{\vartheta}$ which minimize:

$$\sum_{m=1}^{M} (\hat{\gamma}(h_m) - \gamma(h_m; \boldsymbol{\vartheta}))^2$$

where $\gamma(h_m; \boldsymbol{\vartheta})$ indicates a valid model parameterized by $\boldsymbol{\vartheta}$ and evaluated in the center h_m of the m-th class of distances.
To take into account also the variability of the binned estimator, a GLS criterion can be used. In this case we look for the parameters which minimize:

$$(\hat{\gamma}(\mathbf{h}) - \gamma(\mathbf{h}; \boldsymbol{\vartheta}))^T [\mathbb{C}\text{ov}(\hat{\gamma}(\mathbf{h}))]^{-1} (\hat{\gamma}(\mathbf{h}) - \gamma(\mathbf{h}; \boldsymbol{\vartheta}))$$

where $\gamma(\mathbf{h}; \boldsymbol{\vartheta}) = (\gamma(h_1; \boldsymbol{\vartheta}), \ldots, \gamma(h_M; \boldsymbol{\vartheta}))^T$. The limit of this approach is that the covariance of the variogram estimator is, in general, unknown.
A good compromise between OLS and GLS comes from the WLS method, based on the minimization of:

$$\sum_{m=1}^{M} \frac{1}{\omega_m} (\hat{\gamma}(h_m) - \gamma(h_m; \boldsymbol{\vartheta}))^2$$

where ω_m can be set, for example, to $\mathbb{V}\text{ar}(\hat{\gamma}(h_m))$ or $|N(h_m)|$, for $m = 1, \ldots, M$.

1.1.4 Kriging prediction

In most cases, the geostatistical analysis based on a set of n locally distributed observations $\{Z_{\mathbf{s}_1}, \dots, Z_{\mathbf{s}_n}\}$ of the random field (1.1) is aimed to predict the value for $Z_{\mathbf{s}_0}$ at an unsampled location $\mathbf{s}_0$ in D. The geostatistical approach consists of three steps: (1) an experimental variogram analysis based on the observed data aimed to define the empirical variogram; (2) the selection of a valid variogram and the following parameters estimate by a LS criterion; (3) a final Kriging interpolation based on the estimated variogram.
The Kriging predictor at an unsampled locations $\mathbf{s}_0$ is defined as a weighted linear combinations of the observed data according to the following identity:

$$Z^*_{\mathbf{s}_0}(\mathbf{Z}) = \lambda_0 + \sum_{i=1}^{n} \lambda_i Z_{\mathbf{s}_i}. \tag{1.10}$$

The Kriging predictor $Z^*_{\mathbf{s}_0}(\mathbf{Z})$ is the Best Linear Unbiased Predictor (BLUP) (1.10), whose weights $\lambda_0, \lambda_1, \dots, \lambda_n$ are solutions of the following constrained minimization problem:

$$\begin{array}{rl} \min & \mathbb{E}[(Z_{\mathbf{s}_0} - Z^*_{\mathbf{s}_0}(\mathbf{Z}))^2] \\ \text{s.t.} & \mathbb{E}[Z^*_{\mathbf{s}_0}(\mathbf{Z})] = \mathbb{E}[Z_{\mathbf{s}_0}]. \end{array}$$

According to the prior knowledge on the process, we distinguish three Kriging approaches: (a) *Simple Kriging* (SK), (b) *Ordinary Kriging* (OK) and (c) *Universal Kriging* (UK).

(a) *Simple Kriging*
It is employed when the stationarity assumption holds true and the drift term m is known. In this case the unbiasedness constraint implies

$$\lambda_0 = m\left(1 - \sum_{i=1}^{n} \lambda_i\right).$$

The optimal weights vector $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_n)^T$ is solution of the following system

$$\Sigma\boldsymbol{\lambda} = \boldsymbol{\sigma}_0$$

where the ij-th element of the covariance matrix Σ is defined as $[\Sigma]_{ij} = \mathrm{Cov}(Z_{\mathbf{s}_i}, Z_{\mathbf{s}_j})$, with $i, j = 1, \dots, n$. While the i-th element of the $\boldsymbol{\sigma}_0$ vector is defined as $[\boldsymbol{\sigma}_0]_i = \mathrm{Cov}(Z_{\mathbf{s}_i}, Z_{\mathbf{s}_0})$, with $i = 1, \dots, n$.

(a) *Ordinary Kriging*
It is employed when the stationarity assumption holds true, but the drift

term is unknown. In this case the uniform unbiasedness constraint can be expressed as $m = \lambda_0 + \sum_{i=1}^n \lambda_i m$, for any m, which implies:

$$\begin{cases} \lambda_0 = 0 \\ \sum_{i=1}^n \lambda_i = 1. \end{cases}$$

The optimal weights are solutions of the following system

$$\begin{pmatrix} \Sigma & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda} \\ \xi \end{pmatrix} = \begin{pmatrix} \boldsymbol{\sigma}_0 \\ 1 \end{pmatrix} \tag{1.11}$$

where ξ is the Lagrangian multiplier used to solve the constrained optimization problem, while the other elements of the system are defined as before.

(c) *Universal Kriging*
It is applied in non-stationary framework. In this case the (unknown) drift term is modeled as

$$m_{\mathbf{s}} = \sum_{l=0}^{L} a_l f_l(\mathbf{s}) \tag{1.12}$$

where $\{a_l\}_{l=0}^L$ are coefficients in $\mathbb{R}$ independent from the spatial locations, while $\{f_l(\mathbf{s})\}_{l=0}^L$ are known functions, with the same role of the regressors in the linear model case. If we define the vector $\mathbf{f}_0$ such that the l-th element is $[\mathbf{f}_0]_l = f_l(\mathbf{s_0})$ and the design matrix $\mathbb{F}$ such that $[\mathbb{F}]_{il} = f_l(\mathbf{s}_i)$, with $l = 0, 1, \ldots, L$ and $i = 1, \ldots, n$, then the optimal weights defining the UK predictor are solutions of the following linear system

$$\begin{pmatrix} \Sigma & \mathbb{F} \\ \mathbb{F}^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\xi} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\sigma}_0 \\ \mathbf{f}_0 \end{pmatrix}$$

where $\boldsymbol{\xi} = (\xi_0, \ldots, \xi_L)$ is the vector of Lagrangian multipliers coming from the constraints:

$$\sum_{i=1}^n \lambda_i f_l(\mathbf{s}_i) = f_l(\mathbf{s}_0) \qquad l = 0, \ldots, L.$$

In all the three Kriging approaches, we can evaluate the uncertainty in the estimated field through the *Kriging variance*. It is the variance associated with the prediction error $\epsilon^*_{\mathbf{s}_0}(\mathbf{Z}) = Z^*_{\mathbf{s}_0}(\mathbf{Z}) - Z_{\mathbf{s}_0}$ and it is computed (according to the type of performed Kriging) in the following way

$$\sigma^2_{SK}(\mathbf{s}_0) = C(0) - \sum_{i=1}^n \lambda_i C(\mathbf{s}_0 - \mathbf{s}_i); \tag{1.13}$$

$$\sigma^2_{OK}(\mathbf{s}_0) = C(0) - \sum_{i=1}^n \lambda_i C(\mathbf{s}_0 - \mathbf{s}_i) - \xi; \tag{1.14}$$

$$\sigma^2_{UK}(\mathbf{s}_0) = C(0) - \sum_{i=1}^n \lambda_i C(\mathbf{s}_0 - \mathbf{s}_i) - \sum_{l=0}^L \xi_l f_l(\mathbf{s}_0). \tag{1.15}$$

We can notice that in stationary framework, if the variogram is known, the Kriging method gives the possibility to compute prediction at unsampled locations without having estimated the mean function in advance. This cannot be applied in non-stationary framework, since the variogram cannot be estimated according to the (1.5) estimator, because it defines a biased estimator. In this case the drift term, defined by the equation (1.12), must be estimated.
Given the n observations $\mathbf{Z} = (Z_{\mathbf{s}_1}, \dots, Z_{\mathbf{s}_n})^T$, from the definition of the drift term given by equation (1.12), we can write

$$\mathbf{Z} = \mathbb{F}\mathbf{a} + \boldsymbol{\delta}$$

with $\mathbf{m} = \mathbb{F}\mathbf{a} = (m_{\mathbf{s}_1}, \dots, m_{\mathbf{s}_n})^T$ and the residuals $\boldsymbol{\delta} = (\delta_{\mathbf{s}_1}, \dots, \delta_{\mathbf{s}_n})^T$, whose covariance structure is defined by Σ.
If Σ is known, the Generalized Least Square method (GLS) can be used to estimate the coefficients $\mathbf{a}$, by minimizing

$$(\mathbf{Z} - \mathbb{F}\mathbf{a})^T \Sigma^{-1} (\mathbf{Z} - \mathbb{F}\mathbf{a}).$$

The GLS estimator of $\mathbf{a}$ is defined as

$$\hat{\mathbf{a}}^{GLS} = (\mathbb{F}^T \Sigma^{-1} \mathbb{F})^{-1} \mathbb{F}^T \Sigma^{-1} \mathbf{Z}.$$

Since in general the matrix Σ is unknown, we need to estimate it. If the residuals were observed, we can estimate the variogram according to

$$\hat{\gamma}(h) = \frac{1}{2|N(h)|} \sum_{(i,j) \in N(h)} [\delta_{\mathbf{s}_i} - \delta_{\mathbf{s}_j}]^2 \tag{1.16}$$

but also the residuals are unknown informations. We can only use estimated residuals obtained as $\hat{\boldsymbol{\delta}} = \mathbf{Z} - \mathbb{F}\hat{\mathbf{a}}$.
Since $\hat{\mathbf{a}}$ depends on Σ, an iterative algorithm must be defined to jointly estimate $\gamma(\cdot)$ and $\mathbf{a}$. The algorithm can be initialized with the OLS estimate $\hat{\mathbf{a}}^{OLS} = (\mathbb{F}^T \mathbb{F})^{-1} \mathbb{F}^T \mathbf{Z}$ and then the estimated residuals $\hat{\boldsymbol{\delta}}$ are used to compute an estimate of the covariance matrix: $\hat{\Sigma}$. Finally, $\hat{\Sigma}$ will be used to define the GLS estimator of $\mathbf{a}$

$$\hat{\mathbf{a}}^{GLS} = (\mathbb{F}^T \hat{\Sigma}^{-1} \mathbb{F})^{-1} \mathbb{F}^T \hat{\Sigma}^{-1} \mathbf{Z}.$$

The estimate $\hat{\mathbf{a}}^{GLS}$ is used to update the residuals estimate and these steps are repeated until convergence is reached.

Chapter 2

Non-stationary Spatial Data

The most common assumption in geostatistical analysis is stationarity, which gives the possibility to define a global variogram (or covariogram) function describing the spatial relationships among data over the entire study domain. This function describes the second-order properties of the phenomenon and under (intrinsically) stationary assumption it only depends on the separation vector between locations. But, stationarity is more an exception than a general condition. Indeed, local factors (such as environmental or human-based activities, natural or artificial constraints, as well as irregular boundaries) may influence the geographical dependence differently at different locations of the domain, introducing non-stationary effects, reflecting the fact that the characteristics of the phenomenon may vary across the study region. In recent years, various modeling approaches have been proposed to take into account the possible non-stationary structure of spatial distributed phenomena. A brief review of these methods is reported in this chapter. For a complete and in-deep description of the existing methods for analysing non-stationary and spatially distributed random processes we refer the reader to the work of Fouedjio [2016].

2.1 Approaches for modeling non-stationary spatial processes

2.1.1 Moving-Window

Haas [1990] (and Haas [1995]) proposed a moving-window approach for handling the non-stationarity property of spatial processes. The method consists in defining a local window around each location $\mathbf{s}_0$ to be predicted. Within the window the stationarity is assumed, therefore the Ordinary Kriging can be performed. The Kriging predictor is, as usual, a weighted linear combination of the data, but it only takes into account the data within the defined window.
The size of the window is chosen by cross-validation and it controls the trade-off

between the need to have a sufficient number of pairs to estimate the local stationary spatial dependence structure and the need to use a window as small as possible in order to make the stationarity within the window a viable hypothesis. Since the window is often symmetrically designed around the location $\mathbf{s}_0$, the method is not able to capture sharp changes in covariance structure.
Later, Harris et al. [2010] proposed a generalized moving window Kriging (GMWK) approach where the classical variogram estimator is replaced with a geographically weighted variogram estimator, constructed by using kernel smoothing. This kernel weighting function is used to smooth the individual semivariances of each lag interval according to the distance of these paired values from the target location where the estimate of the variogram is computed. Therefore, contrarily to the simple moving window approach, the GMWK also uses information coming from sampled data outside the moving window, with weights depending on the kernel function.
The moving window-based approaches handle the non-stationarity without providing a global non-stationary spatial dependence structure model. Indeed, they provide a collection of local stationary models, each of them is related to a neighborhood of the locations where prediction must be computed. Moreover, since data locations are suddenly included and excluded from the window, this approach may result in discontinuities on the Kriging map.

2.1.2 Basis functions

The basis functions based approach for handling non-stationarity was proposed for the first time by Cohen and Jones [1969]. It is based on the Karhunen-Loève decomposition of a random field $\{Z_{\mathbf{s}}, \mathbf{s} \in D\}$, which defines the random variable $Z_{\mathbf{s}}$ as an infinite linear combination of orthogonal functions

$$Z_{\mathbf{s}} = \sum_{k=1}^{\infty} \omega_k \phi_k(\mathbf{s}) \quad \forall \mathbf{s} \in D$$

where $\{\omega_k\}_{k=1}^{\infty}$ are i.i.d variables and $\{\phi_K(\cdot)\}_{k=1}^{\infty}$ are orthogonal basis functions. In real applications, the random process $\{Z_{\mathbf{s}}, \mathbf{s} \in D\}$ is often approximated by a finite linear combination of orthogonal basis functions

$$\hat{Z}_{\mathbf{s}} = \sum_{k=1}^{M} \omega_k \phi_k(\mathbf{s}) \quad \forall \mathbf{s} \in D.$$

In this framework replicated observations through the time of the random field $\{Z_{\mathbf{s}}, \mathbf{s} \in D\}$, under the temporal stationarity assumption, make easier the estimate of the non-stationary covariance function defined as

$$C(\mathbf{s}_i, \mathbf{s}_j) = \sum_{k=1}^{M} \lambda_k \phi_k(\mathbf{s}_i) \phi_k(\mathbf{s}_j)$$

where $\phi_k(\cdot)$ indicates the k-th eigenfunction of the covariance function and λ_k is the corresponding eigenvalue. Therefore the estimate of the non-stationary covariance function $C(\cdot,\cdot)$ corresponds to the estimate of its eigenfunctions and eigenvalues. The numerical approximations of the eigenfunctions are performed by using principal component analysis on the empirical covariance matrix estimated from the data.
Nychka et al. [2002], replaced the orthogonal basis functions with wavelets. In particular the focus was on computational problems related to the analysis of large data sets. In this case, the data have to be mapped to a regular grid before applying the Kriging analysis. The correlation among the random variables ω_k allows the possibility to define a flexible model which can be used to estimate non-stationary covariance structures.
Even if the approaches based on the use of basis functions are general and flexible, they are not popular for handling non-stationarity, due to the lack of connection with traditional approaches based on the Kriging technique and variograms. Furthermore, these approaches require replicated observations and, even if they provide a global non-stationary spatial dependence model, it is not guaranteed that this resulting global covariance has closed-form.

2.1.3 Space deformation

The space deformation based approach for handling non-stationary spatial data was introduced for the first time by Sampson and Guttorp [1992] and Guttorp and Sampson [1994]. The method consists in transforming the original non-stationary space $D \subseteq \mathbb{R}^p$ into a space of given dimension where stationarity and isotropy assumptions hold true. The geostatistical analysis is not performed on the original random field $\{Z(\mathbf{s}), \mathbf{s} \in D \subseteq \mathbb{R}^p,\, p \geq 1\}$ but on the transformed field $\{Y(\mathbf{u}), \mathbf{u} \in G \subseteq \mathbb{R}^q,\, q \geq p\}$ such that

$$Z(\mathbf{s}) = Y(f(\mathbf{s})) \qquad \forall \mathbf{s} \in D.$$

Function $f : D \subseteq \mathbb{R}^p \to G \subseteq \mathbb{R}^q$ is deterministic, non-linear, smooth and bijective. Since stationarity can be assumed in G, here standard geostatistical techniques can be applied. The variogram $\gamma(\cdot,\cdot)$ of the random field $Z(\cdot)$ is modeled as

$$2\gamma(\mathbf{s}_i, \mathbf{s}_j) = \mathrm{Var}(Z(\mathbf{s}_i) - Z(\mathbf{s}_j)) = \gamma_0(\|f(\mathbf{s}_i) - f(\mathbf{s}_j)\|)$$

for all $\mathbf{s}_i, \mathbf{s}_j$ in D, where $\gamma_0(\cdot)$ represents the isotropic and stationary variogram of the random field $Y(\cdot)$ and $\|\cdot\|$ is the Euclidean norm in the space $\mathbb{R}^q$. The results obtained on $Y(\cdot)$ are then transposed to $Z(\cdot)$ by the inverse deformation function f^{-1}. In general the functions $f(\cdot)$ and $\gamma_0(\cdot)$ are unknown and therefore they need to be estimated.
The space deformation-based approaches had great success as methods for handling data distributed over domains where the Euclidean distance is not the most

appropriate measure of *closeness* among locations. In this case, one must be careful about using a non-Euclidean distance with the traditional Kriging techniques since, as discussed by Curriero [2006], covariance functions that are known to be valid in an Euclidean space are not necessarily valid in a non-Euclidean framework. In these cases the multidimensional scaling method (MDS) can be used to embed locations of the domain of interest in an Euclidean space where the distances between points well approximate the original non-Euclidean distances and where the validity of the described variogram models is guaranteed.
The drawbacks of these space deformation methods concern the necessity of having multiple realizations of the field. Moreover, a non-stationary spatial dependence structure is not always reducible to a stationary model by using a transformation of the space.

2.1.4 Convolution approach

The convolution based approach introduced by Higdon [1998] is based on the possibility of defining a non-stationary process $\{Z(\mathbf{s}), \mathbf{s} \in D\}$ as a convolution of a deterministic function $K(\mathbf{s})$ with a white noise process $\mathcal{X}(\mathbf{s})$ according to

$$Z(\mathbf{s}) = \int_D K(\mathbf{s}-\mathbf{u})\mathcal{X}(\mathbf{u})d\mathbf{u}.$$

An improvement to this method came from Higdon et al. [1999], who proposed the idea of a spatially evolving kernel $K_{\mathbf{s}}(\cdot)$. They defined the random process as a convolution of a deterministic kernel $K_{\mathbf{s}}(\mathbf{u})$, whose parameters vary spatially, with a white noise process $\mathcal{X}(\mathbf{s})$ according to

$$Z(\mathbf{s}) = \int_D K_{\mathbf{s}}(\mathbf{u})\mathcal{X}(\mathbf{u})d\mathbf{u}.$$

According to this definition, the non-stationary covariance of the random field $Z(\cdot)$ can be expressed as

$$C(\mathbf{s}_i, \mathbf{s}_j) = \int_D K_{\mathbf{s}_i}(\mathbf{u})K_{\mathbf{s}_j}(\mathbf{u})d\mathbf{u}.$$

The function $K_{\mathbf{s}}(\cdot)$ is treated as un unknown smooth function which should be estimated in a Bayesian hierarchical framework, together with the model parameters. The choice of the kernel function is crucial for a convolution based approach and, in general, one tries to find a kernel such that the global non-stationary covariance function has closed-form.

2.1.5 Kernel smoothing

The kernel-based approach was proposed for the first time by Fuentes [2001]. He defined a non-stationary random field as a linear and weighted combination of stationary random fields. Each stationary random field describes the phenomenon within a sub-region of the study domain D. This domain is partitioned into K disjoint sub-regions D_k such that

$$Z(\mathbf{s}) = \sum_{k=1}^{K} w_k(\mathbf{s}) Z_k(\mathbf{s}) \qquad \forall \mathbf{s} \in D$$

where the collection $\{Z_k(\cdot)\}_{k=1}^{K}$ are unobserved and orthogonal random functions defined on D and each of them is characterized by a stationary covariance function $C_{\theta_k}(\cdot)$. The linear combination is weighted by using the weights $\{w_k(\mathbf{s})\}_{k=1}^{K}$ that are functions of the distance between the sub-region D_k and the location $\mathbf{s}$ (e.g., the distance between the centroid of D_k and the points $\mathbf{s}$). The weight $w_k(\mathbf{s})$ gives more importance to the random function $Z_k(\mathbf{s})$ at locations $\mathbf{s} \in D_k$ and less importance to random functions at locations outside D_k. The global valid non-stationary covariance can be define as follows

$$C(\mathbf{s}_i, \mathbf{s}_j) = \sum_{k=1}^{K} w_k(\mathbf{s}_i) w_k(\mathbf{s}_j) C_{\theta_k}(\mathbf{s}_i - \mathbf{s}_j).$$

In Fuentes [2001], the shape of the sub-regions was considered as *a priori* known and the number of sub-regions was chosen by using an Akaike or Bayesian information criterion.
Later, Fuentes [2002] introduced an extension to this approach. He defined the random field $\{Z(\mathbf{s}), \mathbf{s} \in D\}$ as a continuous convolution of locally stationary and independent processes $Z_{\boldsymbol{\vartheta}_{(\mathbf{u})}}(\cdot)$ with a kernel weighting function $K(\cdot)$

$$Z(\mathbf{s}) = \int_D K(\mathbf{s} - \mathbf{u}) Z_{\boldsymbol{\vartheta}_{(\mathbf{u})}}(\mathbf{s}) d\mathbf{u}.$$

The index $\boldsymbol{\vartheta}(\cdot)$ is the parameter associated with the local stationary covariance function $C_{\boldsymbol{\vartheta}(\mathbf{u})}(\cdot)$ of the random field $Z_{\boldsymbol{\vartheta}_{(\mathbf{u})}}(\cdot)$ and it is allowed to vary across the domain to reflect the lack of stationarity of the process. The global non-stationary covariance function is then defined according to

$$C(\mathbf{s}_i, \mathbf{s}_j) = \int_D K(\mathbf{s}_i - \mathbf{u}) K(\mathbf{s}_j - \mathbf{u}) C_{\boldsymbol{\vartheta}(\mathbf{u})}(\mathbf{s}_i - \mathbf{s}_j) d\mathbf{u}.$$

Unlike the moving window approach, this method has the advantage that a global model is simultaneously defined everywhere. In addition, for suitable choices of the kernel function, it is possible to obtain a closed-form non-stationary covariance function with locally varying geometric anisotropy.

2.1.6 Partitioning

The approaches based on the partition of the domain assume that the global non-stationary spatial dependence structure can be assumed stationary within some sub-regions of the domain. The stationarity assumption holds true within each sub-region D_k but, contrarily to the partition $\{D_k\}_{k=1}^K$ proposed by Fuentes, here observations are considered independent across different sub-regions. Fuentes assumed that the shapes of the homogeneous regions were known, although this is not a common situation in real applications. Later, Kim et al. [2005] introduced uncertainty about the properties of the homogeneous sub-regions of the domain, by defining a prior distribution on their shape and number. They proposed a random partition approach with the aim to define a method able to model sharp transitions in the covariance function. In Kim et al. [2005], the study region was partitioned into several sub-regions such that the data were assumed to be homogeneous within each region and independent across sub-regions. The partition of the domain was computed according to an Euclidean Voronoi tessellation with K tiles. The number K of sub-regions and the coordinates of their centers were treated as unknown information and they were estimated in a Bayesian framework together with the model parameters. The process within each region was assumed to be a Gaussian process, therefore the overall field was defined as a piecewise stationary Gaussian process. In particular, given a tessellation $\mathbf{t} = (K, \mathbf{c})$ of the domain D, where $\mathbf{c} = (\mathbf{c}_1, \dots, \mathbf{c}_K)$ are the locations of the centers, they assumed a Gaussian distribution within each k-th sub-region

$$\mathbf{Z}_k|\mathbf{s}_k, \boldsymbol{\beta}_k, \mathbf{t} \sim \mathcal{N}_{n_k}(\mathbf{G}_k\boldsymbol{\beta}_k, \Sigma_k)$$

where n_k is the number of data points within the k-th sub-region such that $\sum_{k=1}^K n_k = n$ and $\mathbf{Z}_k = (Z_{\mathbf{s}_{k.1}}, \dots, Z_{\mathbf{s}_{k.n_k}})^T$ is the vector of observations at locations $\mathbf{s}_k = (\mathbf{s}_{k.1}, \dots, \mathbf{s}_{k.n_k})$ within the k-th region. The matrix $\mathbf{G}_k$ is a known design matrix used to capture the trend within each region, and the Σ_k is a covariance matrix that describes the local spatial dependence. Having assigned the priors (uniform for $\mathbf{t}$ and Gaussian for $\boldsymbol{\beta}_k$, with $k = 1, \dots, K$), to draw sample from the posterior model space, they used a modified version of the MCMC approach, due to the need of designing a Markov chain which is able to move between models of different dimensions.

Heaton et al. [2015] defined a partition of the spatial domain into disjoint sub-regions by using hierarchical clustering of observations and finite differences as a measure of dissimilarity. Also in this case stationarity was assumed within each sub-region and independence among different sub-regions.

The approaches based on the partition of the study domain show the problem of defining the local spatial dependence structure at the border of two disjoint sub-regions. This could imply, as in the case on moving window approaches, some discontinuities on the Kriging map. Moreover, the assumption of independence between sub-regions ignores the spatial correlation between neighboring points located in different sub-regions.

2.2 Local Stationarity

From this brief review on the approaches that allow the analysis of second-order non-stationary geostatistical data, we can divided them into two groups: (a) approaches which handle non-stationarity without providing a global non-stationary model for the dependence structure underlying the data (e.g., partitioning and moving window) and (b) approaches which define a global non-stationary model for the covariance structure (e.g., basis functions, space deformation, convolution, kernel smoothing). Moreover, the review shows the importance of the stationarity assumption in geostatistical settings since, in most of the described methods, the stationarity is always considered. Precisely, almost all the methods assume that a non-stationary process can be at least considered locally stationary. The definition of local stationary comes from Matheron [1971].

Definition 2.1. (Matheron, 1971.)
The random field $\{Z_{\mathbf{s}}, \mathbf{s} \in D\}$ is *local stationary* (or *quasi-stationary*) if its mean function $m_{\mathbf{s}}$ and its non-stationary covariance function $C(\cdot, \cdot)$ fulfill the following properties:

(i) $m_{\mathbf{s}}$ is a very regular function varying slowly in space (at the scale of the grid). More precisely, $m_{\mathbf{s}}$ can be considered constant in a neighborhood $V_{\mathbf{s}}$ of $\mathbf{s}$. That is, if we define at any location $\mathbf{s} \in D$ the neighborhood $V_{\mathbf{s}} = \{\mathbf{x} \in D : \|\mathbf{s} - \mathbf{x}\| \leq b\}$ with $b > 0$, then the drift term of a quasi-stationary process satisfies the following relation

$$m_{\mathbf{x}} \approx m_{\mathbf{y}} \approx m_{\mathbf{s}} \qquad \forall (\mathbf{x}, \mathbf{y}) \in V_{\mathbf{s}} \times V_{\mathbf{s}}.$$

(ii) There exists a function of three arguments $C^S(\mathbf{h}; \mathbf{x}, \mathbf{y})$ such that

$$C(\mathbf{x}, \mathbf{y}) = C^S(\mathbf{x} - \mathbf{y}; \mathbf{x}, \mathbf{y})$$

and $C^S(\mathbf{h}; \mathbf{x}, \mathbf{y})$ is a very regular slowly varying function of the two arguments $\mathbf{x}$ and $\mathbf{y}$, for a given $\mathbf{h}$. Therefore, for locations $\mathbf{x}$ and $\mathbf{y}$ in D not too far from each other, the function $C^S(\mathbf{h}; \mathbf{x}, \mathbf{y})$ only depends on $\mathbf{h}$, hence the covariance function is locally stationary.

From this definition, we can conclude that if a random field (1.1) is *local stationary*, then at any location $\mathbf{s}_0 \in D$ there exists a neighborhood $V_{\mathbf{s}_0}$ where the random field can be considered as stationary. However, note that one needs to define this neighborhood in each location of the domain and handles the trade-off between the range of homogeneity of the studied phenomenon and the number of observations available around the considered point.

The method that we propose for handling possibly non-stationary geostatistical data is based on the local stationarity assumption. Although some of the ideas are inspired by the approaches described in this chapter, the family of methodologies here proposed is entirely general and prone to the final aim of analysing complex data with spatial dependence in the framework of *Object Oriented Spatial Statistics* (O2S2) [Menafoglio and Secchi, 2016].

Chapter 3

Random Domain Decompositions for O2S2

3.1 Object Oriented Spatial Statistics

Object Oriented Spatial Statistics (O2S2) is a new branch of statistics focused on the analysis of data sets of complex objects with spatial dependence. O2S2 combines the ideas of Object Oriented Data Analysis (OODA) with the approaches used in Spatial Statistics in order to take into account the spatial dependence among object data. OODA was introduced by Wang and Marron [2007], with the aim to define some new statistical approaches for analysing populations of complex object, such as curves, images, shape objects, tree-structured objects or elements of a manifold.
In univariate statistics the atoms of the analysis are numbers, in multivariate analysis the atoms are vectors, while in O2S2 the *atom* is the whole complex spatially distributed datum. Further, in O2S2 one may need to take into account also the fact that the object-datum may be distributed over a complex domain. Here, we face the need to jointly handle both the object and the domain complexities.
In O2S2 the atom is represented in the so-called *feature space*. This space should represent, through its geometry, all the salient data features. Within this space we need to formalize appropriate definitions of distance and similarity between object data, such that the fundamental idea of geostatistics, i.e., points close together are likely to be correlated, may still hold true. For these reasons, in the O2S2 framework, the object data are often represented within a separable Hilbert space, where we are able to define the fundamental operations $(+, \cdot)$, the inner product and the induced norm.
The complexity related to a datum may concern its dimensionality or its constraints (or both). An example of complex spatial data treated within the O2S2 is represented by the particle-size densities (PSDs) dataset collected in a German aquifer, analysed by Menafoglio et al. [2016]. In this case the datum is a probabil-

ity density function associated with the local distribution of the particle size measured at a given location of the aquifer. It is a curve, thus an infinite-dimensional datum, but, in addition, it is a distributional datum, hence constrained to be positive and integrate to unity. These complexities cannot be ignored for the final aim of spatial prediction of PSDs at unsampled locations, since O2S2 aim to predict the whole complex object datum, not only some of its attributes. In this case a Bayes Hilbert space was used as feature space, that is a space of positive functions with constant integral, hence it precisely represents, with its geometric properties, the attributes of the complex data.
The complexity of the domain may be associated with its size or its geometrical constraints. When data are observed over a very large domain, the definition of an unique global model describing the spatial dependence may be not an appropriate approach, while it might be better to perform several local analyses. Furthermore, also a *small* domain may be considered as a complex domain, e.g., for the presence of holes, highly irregular boundaries or barriers. In this case we are not able to define the meaning of global stationarity, regardless of the observed phenomenon. An example can be the dataset of population densities distributed over the Island of Montréal [Sangalli et al., 2013]. Hence, data are simple scalar observations, while the complexity of the analysis is associated with the domain. Indeed, the Island of Montréal is located between two rivers, which represent natural geographical constraints. Within the island there are some areas (the harbour and the public parks) where people cannot live. These regions represent the holes of the domain and must be taken into account if the goal of the analysis is the reconstruction of the population densities over the whole domain. In Sangalli et al. [2013] a semi-parametric approach was used for the analysis, which takes into accounts the presence of holes and the possible presence of boundary conditions.
The complexity of a domain with holes or irregular boundaries is not only associated with the possible necessity of defining boundary conditions, but also with the more difficult problem of finding an appropriate measure of distance between points. Most applications of Geostatistics have defined the separation between sample points simply by using the Euclidean distance. However, in many environmental settings some features (natural or artificial) may act as barriers. The presence of such barriers is not taken into account if the analysis is based on the Euclidean distance between locations.

Barriers are common features in coastal or estuarine environments. The study of estuarine ecosystems is becoming increasingly important, in particular for analysing the impact of the human activities on these environments, that are among the most productive resources of the Earth. Estuaries are irregularly shaped non-convex regions. Therefore the classical geostatistical methods based on the Euclidean distance are not suitable for their analyses. Previous methods based the variogram modeling and the Kriging analysis on the use of non-Euclidean distance metrics, such as the *water distance* defined in Rathbun [1998]

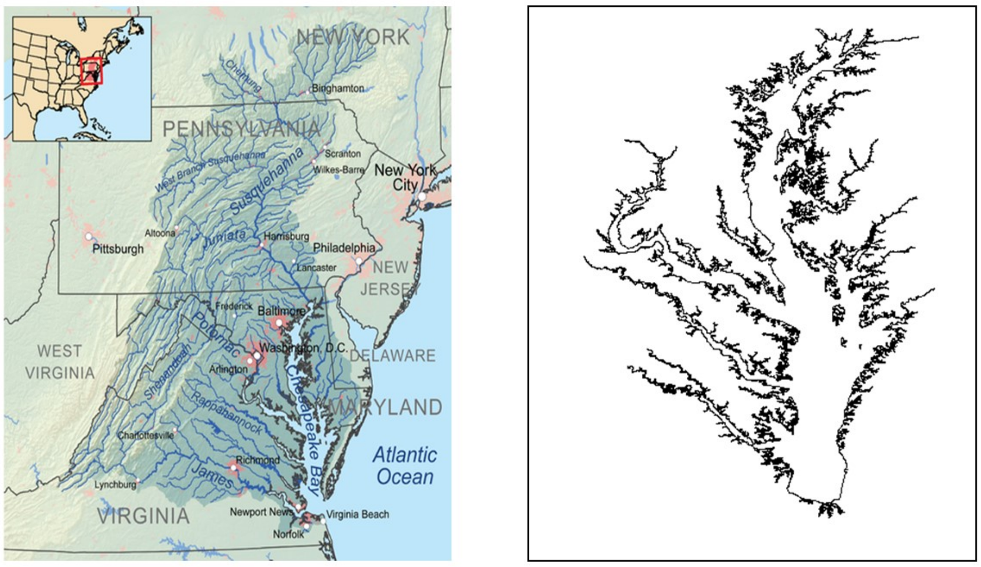

Figure 3.1: *Chesapeake Bay. Left panel: satellite image from the https://commons.wikimedia.org/wiki/File:Chesapeakewatershedmap.png website. Right panel: shape file used for the analysis.*

or the *landscape-based distance* defined in Jensen et al. [2006]. In these cases, significant problems may arise, due to the fact that the valid models defined in the Euclidean framework (see, e.g., those recalled in Chapter 1, Section 1.1.3) may be no longer valid when non-Euclidean metrics are used.
The problem that motivated our work is related to the depletion of dissolved oxygen (DO) in the Chesapeake Bay. The Chesapeake Bay is the largest, most productive and biologically diverse estuary in North America (Figure 3.1). The oxygen depletion problem was addressed for the first time by Newcombe and Horne [1938]. These scientists pointed out the contribution of physical and biological factors to the seasonal reduction of DO in the bottom water of the Bay. During the following years, the excess of nutrient pollution made the DO depletion and its consequences for ecosystem dynamics a central theme of research in the Chesapeake Bay. Indeed, the low oxygen levels are insufficient to support most marine life and habitats. For this reason, these low-oxygen areas are called *Dead zones*.
As we can see from Figure 3.1 the boundaries of the Chesapeake Bay are highly irregular and very jagged. Likewise many estuaries, the Chesapeake Bay has several tributaries arising from the central unit. These tributaries are separated by long and narrow areas of land that represent barriers for the distributions of many aquatic variables. Moreover, the different uses of the watersheds imply different chemical and biological characteristics of points located in adjacent tributaries. Despite adjacent tributaries can show different characteristics, they result to be

very close and, consequently, presumed to be *similar* if the Euclidean distance is used for the geostatistical analysis. Domains like this estuary require appropriate statistical methods to be analysed in order to take into account the complexity of the region. Furthermore, some environmental variables may take deterministic values at the estuarine boundaries or they can be functional data, therefore the complexity of the domain must be associated with the complexity of the data.
This dissertation aims to define a novel methodology allowing for the analysis of complex data distributed over complex domains. The complexity of the datum addresses our attention on a non-parametric and computationally intensive approach, that allows to avoid strong assumptions on the distribution generating the random field. On the other hand, the complexity of the study region and the possible non-stationarity motivate us to develop an approach based on local object oriented geostatistical analyses, instead of an unique global analysis. In the local framework, the method is based on the well-documented geostatistical techniques (such as variogram estimator, Kriging predictors). These local analyses are here properly developed and combined to handle data distributed over complex domains where the adjacency relationships among locations cannot be described by the Euclidean distance, still avoiding to work with non valid variograms.

3.2 Random Domain Decompositions (RDDs)

Performing local analyses suggests the idea of partitioning the study domain into disjoint sub-regions, such that within each sub-region stationarity is a viable assumption. Partitioning the domain leads overall to a model simplification: here, a collection of locally stationary models are considered instead of a global (and complex) non-stationary model.
The first issue arising within this approach then regards how to perform the partition. In most real applications we do not have any information regarding the optimal partition underlying the data. Therefore, we cannot generally consider it as *a priori* information, as done by Fuentes [2001]. A possible solution may be to estimate an optimal partition by using the information coming from the observed data. This was the approach proposed by Kim et al. [2005], recalled in Chapter 2. The application of the latter method to the analysis of object data would become challenging since we can hardly make similar distributional assumptions on these complex data.
The complexity associated with the nature of object data motivates us to move away from the idea of estimating an optimal partition of the study region. Indeed, we are not interested in defining an optimal partition, we rather want to build our analysis on a collection of clever, but simple, partitions of the domain.

3.2.1 Voronoi RDDs for complex domains

Aiming to avoid strong distributional assumptions, we will randomly perform the domain decomposition through simple methods, regardless and independently of the data structure and complexity. It is clear that a single random partition rarely approximates the real existing subdivision of the domain. For this reason we do not perform a single simple random partition, but we use several random partitions of the domain as starting points for the local analyses of the same dataset. Each analysis is based on a different system of neighborhoods.
There exist many ways to perform a random decomposition of the study domain. Although in principle the methodology we propose is completely general, for ease of exposition we focus in this work on a particular decomposition: the Voronoi tessellation. This partition is defined by a set of sites (called *nuclei*) and a metric function $d(\cdot,\cdot)$. Given a spatial domain $D \subset \mathbb{R}^p$, assume that we want to randomly divide the domain into K disjoint sub-regions. The set of nuclei of the Voronoi tessellation is defined by sampling K points in D from an appropriate distribution F defined on D or by randomly selecting K points among the n sampled locations. In our work, the centers were uniformly extracted among the sample sites. In case of non-homogeneous distribution of the measurement locations, a non-homogeneous sampling scheme can be used (e.g., by considering a tessellation generated by non-homogeneous Poisson processes or more complex point processes).
Let $\Phi_K = \{\mathbf{c}_1, \dots, \mathbf{c}_K\}$ be the locations set of the K nuclei in D. The k-th Voronoi cell is defined as follows

$$V(\mathbf{c}_k|\Phi_K) = \{\mathbf{s} \in D : d(\mathbf{s}, \mathbf{c}_k) \leq d(\mathbf{s}, \mathbf{c}_j),\ \text{for all } \mathbf{c}_j \in \Phi_K,\ j \neq k\}. \tag{3.1}$$

In other words, the Voronoi cell $V(\mathbf{c}_k|\Phi_K)$ consists of all the points of D closer to the nucleous $\mathbf{c}_k$ than to any other nucleous, according to the metric $d(\cdot,\cdot)$.
There are no restrictions on the type of metric; obviously different metrics define different partitions of the same domain, even if the set of nuclei is the same. Typically, the Euclidean metric is used for the construction of Voronoi tessellations. In this case, if the domain is a convex region, each Voronoi cell is a convex polytope with straight segments as boundaries. Voronoi cells can be defined for metrics other than Euclidean, such as the Mahalanobis distance, Manhattan distance or the metric induced by local kernels. However, in these cases the boundaries of the Voronoi cells may be more irregular than in the Euclidean case. Furthermore, the properties of the domain, such as irregular shapes or the presence of barriers, can be used for choosing a more suitable metric $d(\cdot,\cdot)$ for the definition of each domain decomposition. As we shall see later in detail, we here use an appropriate spatial metric for defining the relation of *closeness* among the observations within a possibly complex domain. The partition based on such a metric allows the definition of sub-regions which are more likely to be homogeneous. Here, the key idea is to map the sampled locations on a graph representing the spatial adjacency and

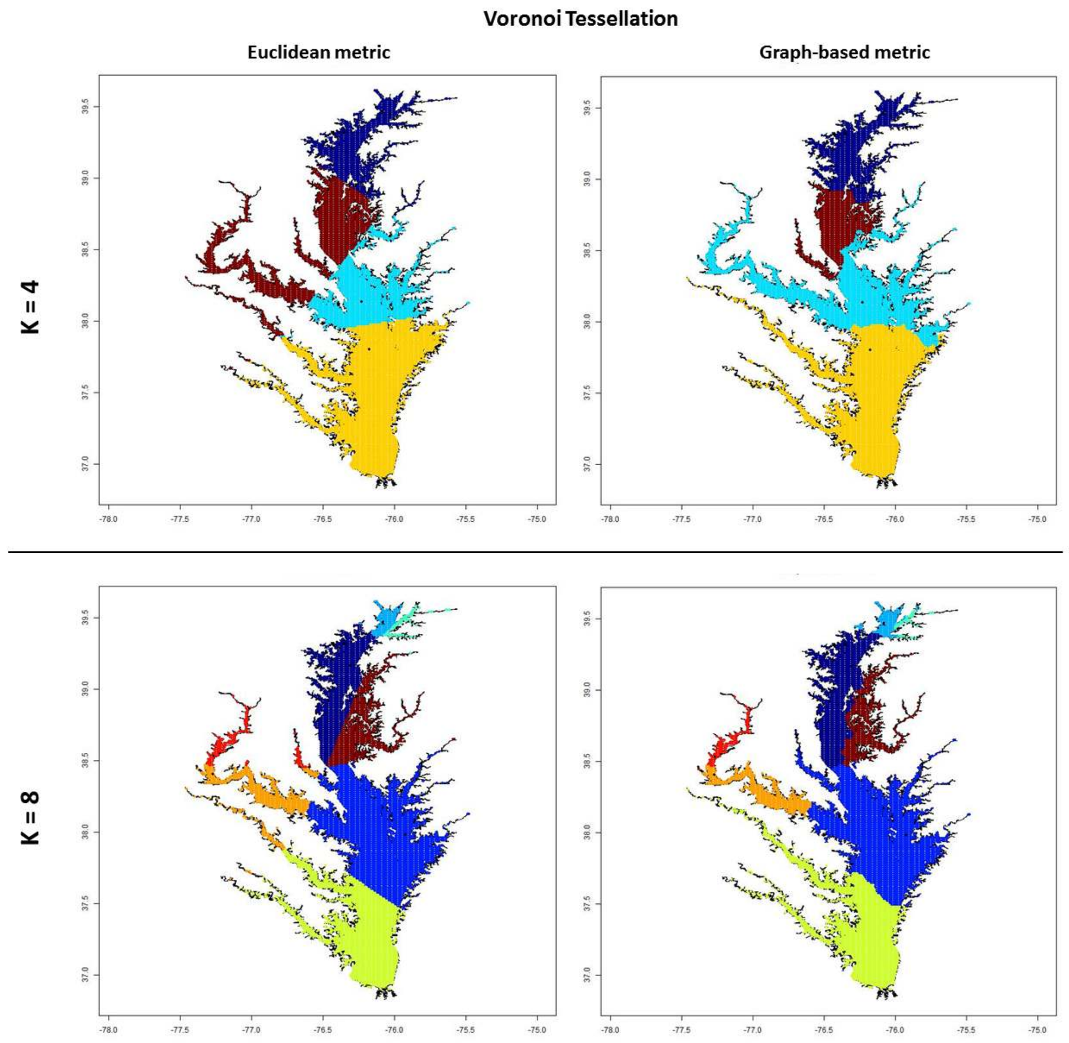

Figure 3.2: *Voronoi tessellations of the Chesapeake Bay. Top panels: $K = 4$; bottom panels: $K = 8$. Right panels: Euclidean metric; left panels: graph-based metric.*

neighborhood relation of the observed objects. This gives the possibility to use a graph-based metric for defining homogeneous regions within a complex domain, which takes into account the geometrical properties of the domain and it better describes the closeness among points, with respect to the Euclidean metric. The construction of such graph is based on the Delaunay triangulation of the domain, as we will discuss later. Figure 3.2 shows the differences between tessellations of the Chesapeake Bay based on the Euclidean metric (on the right side) and tessellations, with the same centers, obtained by using a graph-based metric (on the left side). In the top panels the tessellation divides the estuarine domain into $K = 4$

sub-regions, while in the bottom panels the sub-regions are $K = 8$. The differences are, in particular, in the partitions of tributaries areas: with an Euclidean approach points located on different tributaries are assigned to the same Voronoi tile. This situation rarely occurs with a graph-based metric tessellation, where locations belonging to different tributaries are likely to be assigned to different Voronoi neighborhoods.

3.2.2 Voronoi tessellation and coverage property

The choice of using a Voronoi tessellation approach is due to its simplicity, as well as to the numerous interesting properties of this type of domain partition. The most interesting properties for our purpose are the *coverage* property and the duality between Voronoi tessellation and Delaunay triangulation (as we will describe in Chapter 5).
The *coverage property* was proved by Penrose [2007]. Let A be a Lebesgue measurable set, with $A \subset D$ and set $V_k := V(\mathbf{c}_k|\Phi_K)$, we define

$$A^K = \bigcup_{\mathbf{c}_k \in A} V_k.$$

A^K is an approximation of A, given by the Voronoi cells whose nuclei belong to A. The coverage property states that the set A^K is a consistent estimator for the unknown set A, since:

$$|A^K \Delta A| \to 0 \qquad \text{as } K \to \infty$$

where Δ denotes the symmetric difference between two sets and $|\cdot|$ is the Lebesgue measure. The proof of this property is based on the assumption that the support of F includes D and it represents a strong law of large numbers in the Voronoi tessellation framework.
Therefore an unknown set A can be always consistently estimated by the corresponding union of Voronoi cells, whose nuclei belong to A. This coverage property provides a strong basis upon which building RDDs-based methods. Indeed, if $\{A_j\}_{j=1}^M$ represents the (unknown) collection of sets that partition the domain D into homogeneous sub-regions, the coverage property guarantees that, when the tessellation becomes more refined, the homogeneous sub-regions of D are well approximated by appropriate sets of Voronoi elements.

3.3 Spatial statistics based on RDDs

We here propose RDDs-based methods for O2S2, aiming to develop a flexible class of methodologies able to deal with general types of data (e.g., object data) and

general domains, possibly complex, convoluted or with holes.
Following a *divide et impera* approach, given a system of neighborhoods (i.e., a domain partition) randomly generated, we shall propose to perform K *weak* local analyses, K being the number of sub-regions, each analysis restricted to a sub-region of the domain. To filter out the dependence of the analysis on the single realization of the partition generating the result, these local analyses shall be repeated for several realizations of the random domain decomposition, and then aggregated into an unique final *strong* analysis. The method we propose is based on a **bagging** approach, acronym used by Breiman [1996] to define a two steps procedure which includes: (1) a first **b**ootstrap stage, where the same analysis is performed several times on different learning samples, and (2) a final **agg**regat**ing** stage where the repeated analyses are aggregated into a final analysis.

3.3.1 Bootstrap step

During the bootstrap step, the domain is repeatedly and randomly partitioned into K sub-regions (e.g., by using Voronoi tessellations). Once a K dimensional partition has been defined, K local analyses are performed, one for each sub-region. For ease of exposition, we shall focus on spatial prediction, which will be the core of the present study. Nevertheless, the method is entirely general, and apt to be applied in varied contexts.
If the final goal is to predict the value of the random variable $Z_{\mathbf{s}_0}$ at an unsampled location $\mathbf{s}_0 \in D$, at each bootstrap iteration the BLUP of $Z_{\mathbf{s}_0}$ can be defined according to an Ordinary Kriging approach restricted within the sub-region where $\mathbf{s}_0$ is located. Even though, in this case, the Kriging predictor would be thus defined as a linear combination only of the data within the sub-region of $\mathbf{s}_0$, one could borrow strength from the information coming from neighbour regions, e.g., by employing geographically weighted statistics to estimate the spatial dependence. For example, a weighted estimate of the local variogram could be performed, where the weights are defined according to a kernel function $K_\epsilon(\cdot,\cdot)$. The kernel downweights the contribution of data "far apart" from the center of the neighborhood, where the range of influence of neighbour locations is controlled by the bandwidth parameter ϵ. This weighted version of the variogram would be in contrast with the assumption of independence among disjoint sub-regions of the previously mentioned approaches, but it is a relevant source of flexibility and robustness for the methodology, as we shall discuss below.
As in the standard bagging approach, during the bootstrap step multiple versions of the same target (e.g., predictor for $Z_{\mathbf{s}_0}$) are generated. Contrarily to the classical definition, the difference between two bootstrap iterations does not concern the dataset used for the analysis, but the partition of the domain. Indeed, in the classical definition of the bagging meta-algorithm [Breiman, 1996], given an n-dimensional dataset $\mathcal{L} = (Z_{\mathbf{s}_1}, \dots, Z_{\mathbf{s}_n})$, at each iteration a new n-dimensional

dataset $\mathcal{L}^b$ was generated from $\mathcal{L}$ by using combinations with repetitions. In our approach we use the same original dataset $\mathcal{L}$, but the analyses are based on different partitions of the domain. In the case of Kriging prediction via RDDs, at the b-th iteration the predictor of $Z_{\mathbf{s}_0}$ is a linear combination of a set observations which is not necessarily the same set of data that will define the predictor at the $(b+1)$-th iteration.
These simple random partitions give the possibility to perform simple, independent weak local analyses: although computationally intensive, this class of methodologies is suitable for introducing parallel approaches.
After B replicates of the K local analyses aimed to define a predictor for $Z_{\mathbf{s}_0}$, the output of the bootstrap step consists of:

- a collection of Kriging predictors $\{Z^{*b}_{\mathbf{s}_0}(\mathbf{Z})\}_{b=1}^{B}$;

- a collection of estimated variograms $\{\hat{\gamma}^b(\mathbf{h}; \mathbf{s}_0)\}_{b=1}^{B}$
  (each of them describes the second-order property of the study random field limited to the sub-region where $\mathbf{s}_0$ was located at the b-th bootstrap iteration).

3.3.2 Aggregating step

The multiple weak local analyses of the same analysis have to be aggregated into a final strong analysis. In this work the focus will be on the aggregation of the predictors, while we will not consider the aggregation of the local measures of spatial dependence, which will be the scope of future work.
The multiple realizations of the target predictor, each of them conditional to the partition and to the data, will be then used to define a final aggregated predictor, which is expected to have a smaller variance with respect to the predictor obtained through a single partition of the domain. If the variable that we want to predict is a real-valued random variable, the aggregate predictor can be obtained as an average (or a weighted average) of the predictors obtained along the bootstrap repetitions. If we wanted to perform classification, the final predictor would be the label assigned to the target point $\mathbf{s}_0$ and, in this case, the aggregation could be based on the plurality vote criterion.
We remark that the choice of defining multiple versions of the same predictor, based on different partitions of the domain, allows to overcome the main drawback of the previous partitioning and moving window approaches, that is the presence of discontinuities in the Kriging map. Indeed, since we do not define a single partition (or a single window around each grid point to be predicted) we do not expect discontinuities on the predictions of points located along the boundaries defining two different sub-regions, because the final estimate at these points is an average over multiple values obtained during the bootstrap replicates.

3.4 Parameters setting

Contrarily to the partitioning approaches defined in Chapter 2 (Section 2.1.6) the proposed methodology allows avoiding any strong distributional assumption for the definition of the partition, but only requires to initialize at most three parameters: the number of auxiliary analyses B, the number K of sub-regions defining the partition of the domain and, possibly, the bandwidth parameter ϵ defining the kernel function, if this is used. Furthermore, unlike the method of Kim et al. [2005], here we do not need to estimate an "optimal" position of the centers of the K sub-regions, since every time they are randomly and independently redefined. The K centers, which define the Voronoi nuclei, may be randomly selected among the n observations or as points within the domain D. The parameter B should be chosen large enough to ensure that the algorithm reaches a desired accuracy. It controls the robustness of the final result: the higher the number of B *weak* analyses performed, the *stronger* the basis upon which the final result is obtained.

3.4.1 Number of sub-regions: bias-variance trade-off

While the parameter B can be initialized to a sufficiently large value which guarantees good results and a reasonable computational time, the parameter K should be carefully evaluated since it has great influence on the algorithm performances. Indeed, the value of K affects the Kriging bias-variance trade-off: if K is small, the predictor for an unsampled location will be based on sub-samples that are large, on average, in terms of numerosity. This ensures a minimal variance, but on the other hand the partition of the domain will be coarse, with the consequence that the local stationarity assumption within each sub-region may not be verified (maximal bias). The limiting case is $K = 1$: here we would assume stationarity over the whole domain, although we might not even be able to formulate a definition of stationarity due to, e.g., domain complexities, such as holes or irregular boundaries. On the other hand, if K increases, the partition of the domain will become more and more refined, being able to accurately define the boundaries of different homogeneous sub-regions. This ensures a minimal bias, but at the same time the size of each sub-sample will decrease: a predictor based on few observations will eventually show maximal variance. The limiting case is $K = n$, when the prediction of $Z_{\mathbf{s}_0}$ is based on a single observation, that is the closest datum to the location $\mathbf{s}_0$. It is then apparent that setting the parameter K is key for the analysis. Methods to do this will be discussed in Chapter 5.

3.4.2 Introducing a kernel

As we mentioned before, we can include within the general RDDs methodology, a geographically weighted statistics viewpoint by using a kernel function. This is

an optional choice which contributes to the flexibility and robustness of our proposal. However, the use of this kernel function can be unimportant in some cases, e.g, if many observations of the phenomenon are available, while it may become necessary if we have very few observations, since each local analysis would be then performed on a subset of the data of very small size.
The first choice regards the type of kernel function. In this work, we will mostly focus on isotropic Gaussian kernels, defined as $K_\epsilon(\mathbf{x}, \mathbf{y}) = e^{-\frac{1}{2\epsilon^2}d(\mathbf{x},\mathbf{y})^2}$ where $d(\cdot, \cdot)$ indicates the distance function on the spatial domain D. Nevertheless, the choice may concern other types of kernel function, possibly driven by prior knowledge on the phenomenon. As we can see from Figure 3.3, if we fix a point $\mathbf{x} \in D$, the Gaussian kernel assigns the same value to each point $\mathbf{y} \in D$ at the same distance from $\mathbf{x}$.
As regards the bandwidth parameter ϵ, it controls the range of influence of the observations on the estimate of a local variogram. It allows to ground the estimate of the variogram within the k-th neighborhood not only on the local information, but also on that coming from distant locations. A too small value of ϵ may imply a variogram estimate based on too few observations, which can be reflected in a biased estimate, while a too large value of ϵ can assign high weights to observations which are very far away from the considered neighborhood, with the risk to make the domain partition useless.

An example of the differences which can occur in the kernel weights if different values of the parameter ϵ are used is showed in Figure 3.3. If we want to estimate the variogram within a neighborhood whose center is the black point of the Po-

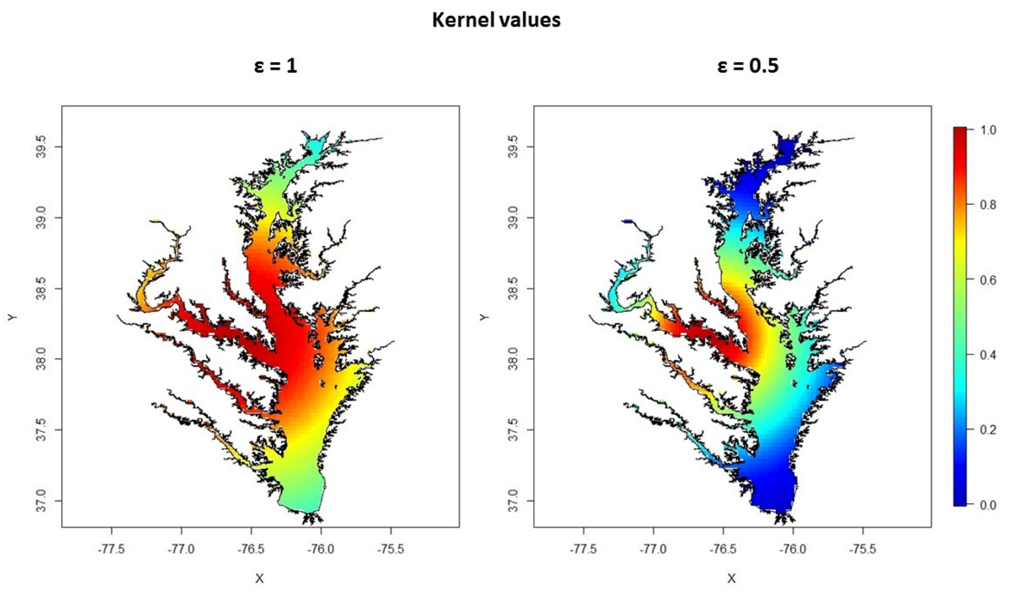

Figure 3.3: *Isotropic Gaussian kernel. Left panel:* $\epsilon = 1$, *right panel:* $\epsilon = 0.5$

tomac river (the main tributary of the estuary) and set $\epsilon = 1$ we will assign high weights to information provided by data at locations very far away. For example observations from points on tributaries on the other side of the central unit, such as points of the Choptank river, contribute to the estimate with weights very close to 1, even if the distance is around 115 km. Instead, with $\epsilon = 0.5$ the highest weights are assigned only to points very close to the considered center: the red points, which are associated with the highest kernel values, are distant from the black points at most 35 km.
In these two examples the kernel weights were based on the Euclidean distance, but also another type of distance can be used. In particular when the analysis concerns data distributed over complex domains, like this estuary, the kernel could represent the topology of the study region by using a non-Euclidean metric which should reflect the geographical characteristics of the domain, as we will examine in depth in Chapter 5.

Chapter 4

Kriging-RDDs: the case of regular domain

We here propose *Random Domain Decompositions* (RDDs) for Kriging prediction, a method based on random partitions of the domain and aimed to predict the value of a random field at unsampled locations. The method is detailed in Section 4.1 and a sketch of the pseudocode is showed in Figure 4.3. The algorithm introduced for the analysis of Section 4.2 has been implemented in `R` [R Development Core Team, 2008].
Even if the description of the method will be based on simple real-valued random fields, the approach is totally general. As we can see from the pseudocode in Figure 4.3, the initialization step does not require any assumption on the nature of the distribution which generated the data. Therefore, the method may be used to handle more complex object data in the O2S2 framework.

4.1 Kriging prediction via RDDs method

This section is aimed to detail each step of the described Random Domain Decompositions method for Kriging prediction over a regular domain.

4.1.1 Bootstrap stage

Assume that n observations are available over the domain D and we want to use this dataset to estimate the values of the random field (1.1) at an unsampled location in D or a grid of unsampled locations. At each bootstrap replicate a *weak* predictor for each $Z_{\mathbf{s}_0}$ is defined, conditional to the random Voronoi tessellation generated at that iteration. The following steps are repeated B times.

Step 1. Voronoi tessellation

Given the parameter K (which is the same for all the iterations), the centers of the Voronoi tiles are uniformly sampled among the sampled locations $\{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$. Let $\Phi_K^b = \{\mathbf{c}_1^b, \ldots, \mathbf{c}_K^b\}$ be the set of selected nuclei, the collection $\{V(\mathbf{c}_k^b|\Phi_K^b)\}_{k=1}^K$ defines the random Voronoi tessellation obtained at the b-th bootstrap iteration, where each Voronoi tile $V(\mathbf{c}_k^b|\Phi_K^b)$ is defined according to (3.1). From Figure 4.1 it is possible to see some examples of the Voronoi tessellation of a simple square domain. Each column of images refers to a fixed value for the parameter K. For each value of K, according to the RDDs method, B random domain partitions are performed. The squared domain is a very simple region, therefore the Voronoi neighborhoods can be defined by using an Euclidean metric.

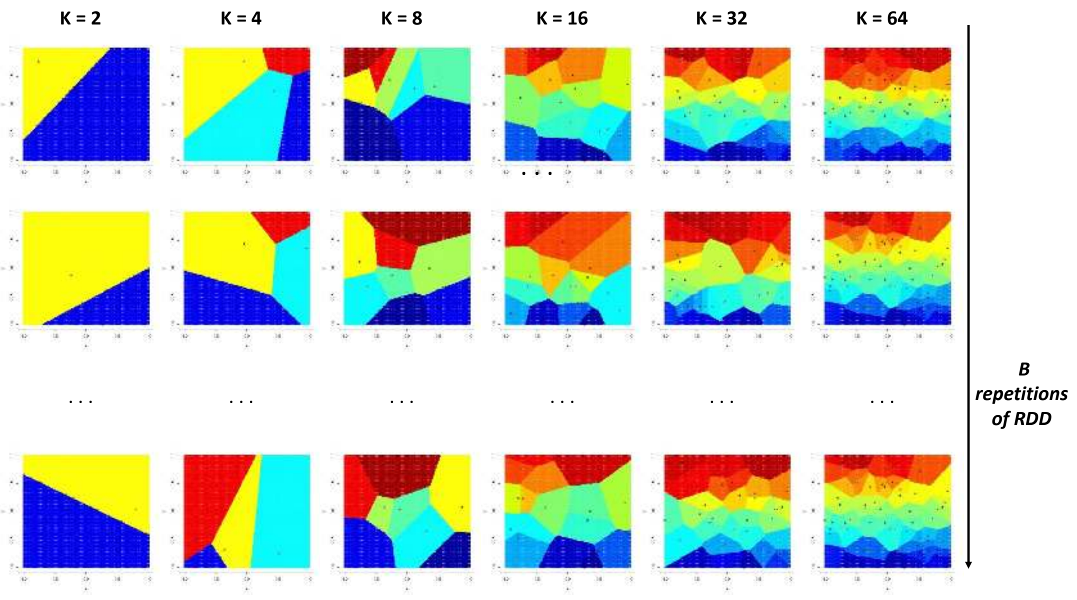

Figure 4.1: *B repetitions of the Voronoi Tessellation for a simple square domain.*

Step 2. Variogram Estimate

Under the local stationary framework, a non-parametric estimator of the local variogram can be obtained via method of moments by using kernel-based weights [Fouedjio et al., 2016]. At each b-th bootstrap iteration we need to estimate K local variograms. For ease of notation we omit the index b. Nevertheless, one needs to remember that the following variogram estimate is to be interpreted as conditional to the realization of the random Voronoi tessellation of the domain obtained at that bootstrap iteration. The estimate of the k-th local variogram is computed at the center $\mathbf{c}_k$ (the Voronoi nucleous) of the corresponding tile. Let $\gamma(\mathbf{h}; k)$ be the variogram related to the k-th Voronoi neighborhood evaluated for a spatial lag $\mathbf{h} \in \mathbb{R}^p$. Its empirical estimate $\hat{\gamma}_\epsilon(\mathbf{h}; k)$ is defined as:

$$\hat{\gamma}_\epsilon(\mathbf{h}; k) = \frac{\sum_{N(\mathbf{h})} K_\epsilon(\mathbf{c}_k, \mathbf{s}_i) K_\epsilon(\mathbf{c}_k, \mathbf{s}_j) \|Z_{\mathbf{s}_i} - Z_{\mathbf{s}_j}\|^2}{2 \sum_{N(\mathbf{h})} K_\epsilon(\mathbf{c}_k, \mathbf{s}_i) K_\epsilon(\mathbf{c}_k, \mathbf{s}_j)} \tag{4.1}$$

where $N(\mathbf{h}) = \{(\mathbf{s}_i, \mathbf{s}_j) \in D \times D : \mathbf{s}_i - \mathbf{s}_j = \mathbf{h}\}$. The norm $\| \cdot \|$ is in general the norm on the feature space; in case of real-valued random field it simply indicates the absolute value: $\|Z_{\mathbf{s}_i} - Z_{\mathbf{s}_j}\| = |Z_{\mathbf{s}_i} - Z_{\mathbf{s}_j}|$.
The kernel $K_\epsilon : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}^+$ is a non-negative symmetric function which depends on the bandwidth parameter $\epsilon > 0$. We use a Gaussian kernel, hence a kernel with exponential decay

$$K_\epsilon(\mathbf{x}, \mathbf{y}) = e^{-\frac{1}{2\epsilon^2} d^2(\mathbf{x}, \mathbf{y})} \tag{4.2}$$

where the distance $d(\mathbf{x}, \mathbf{y})$ must be the same distance used for defining the Voronoi tiles in (3.1). If we use the Euclidean distance, the Gaussian kernel will be an isotropic kernel: all the points at the same (Euclidean) distance from $\mathbf{c}_k$ have the same kernel weights.
This choice for the kernel is coherent with the Voronoi tessellation. Indeed a point $\mathbf{s} \in D$ is assigned to the nearest Voronoi nucleous in the Φ_K^b set, i.e., the center at minimum distance according to the metric $d(\cdot, \cdot)$. In fact, this center is also associated with the maximum value of the kernel function $\{K_\epsilon(\mathbf{s}, \mathbf{c}_k)\}_{k=1}^K$, as we can see from Figure 4.2. Hence, the following relation holds true

$$\mathbf{s} \in V(\mathbf{c}_k|\Phi_K) \iff K_\epsilon(\mathbf{s}, \mathbf{c}_k) \geq K_\epsilon(\mathbf{s}, \mathbf{c}_j) \quad \forall j = 1, \ldots, K, \quad j \neq k.$$

The kernel-based estimator defined by equation (4.1) is a weighted local average of squared differences between all pairs of observations separated by a distance $\mathbf{h}$. The kernel function is used to smoothly down-weight the square differences according to the distance of these paired values from the center $\mathbf{c}_k$. To each data pair is assigned a weight proportional to the product of the individual weights. Therefore, observation pairs close to the center $\mathbf{c}_k$ have more influence on the local variogram estimator for the k-th neighborhood, than those which are far away. Moreover, the use of a kernel allows to make more robust the estimation of variograms in sub-regions with low sampling density. Indeed, without the use of a kernel function, the estimated variogram within sub-regions with very few observations may prove to be very uncertain. We remark that, the choice of employing a weighted variogram estimate is in contrast with the assumption of independence and strong discontinuities among disjoint sub-regions, since the variogram estimate within a Voronoi tile is based also on observations coming from other neighborhoods. Hence, unlike the previous partitioning approaches, we allow taking into account both the spatial correlation between points within the same sub-region and the correlation between neighboring points located in different sub-regions.

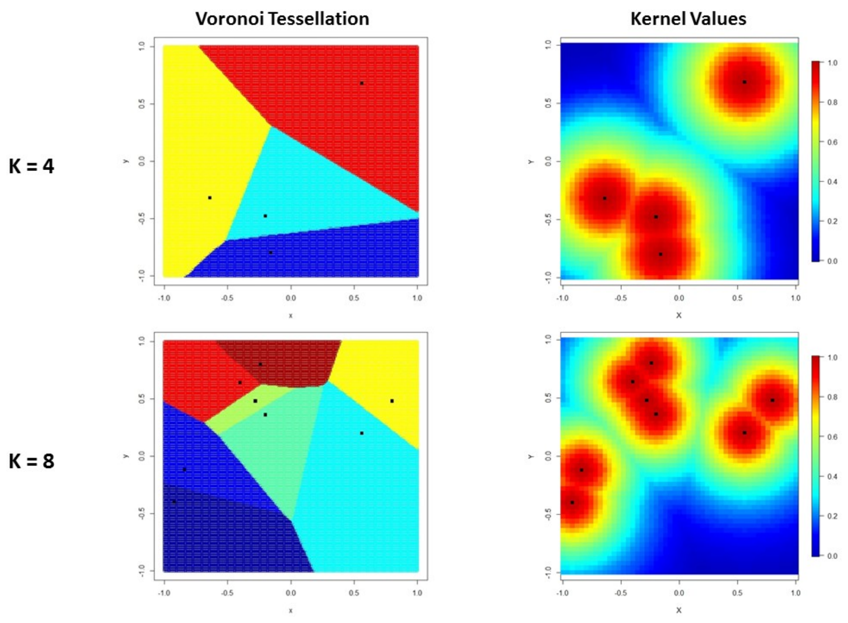

Figure 4.2: *Voronoi tessellation and the corresponding values of the kernel weights.*

The kernel function $K_\epsilon(\cdot,\cdot)$ depends on the bandwidth parameter ϵ, which is a key parameter for the analysis. In Fouedjio et al. [2016], a similar bandwidth parameter was used to set the size of the local stationary neighborhoods. The parameter ϵ has to be initialized before performing the Kriging analysis; a possible data driven method for the selection of ϵ may involve the minimization of the cross-validation mean square error [Zhang and Wang, 2010]:

$$MSE(\epsilon) = \frac{1}{n}\sum_{i=1}^{n}(Z_{\mathbf{s}_i} - \hat{Z}_{-i}(\mathbf{s}_i;\epsilon))^2$$

where $\hat{Z}_{-i}(\mathbf{s}_i;\epsilon)$ is the spatial predictor computed at location $\mathbf{s}_i$ using all the observations except $Z_{\mathbf{s}_i}$. The value of the parameter K may influence the initialization of the parameter ϵ. If K has a small value, we should use a large value for ϵ, otherwise we will assign very small weights even to observation pairs within the same Voronoi tile, where we assumed homogeneity. Instead, if K is large, the size of each Voronoi tile is smaller, therefore we should use a smaller value for ϵ - still compatible with the need to provide robust variogram estimates - otherwise we would contrast the partition of the domain. In our studies we did not use data driven methods for the initialization of ϵ, but we tested the robustness of the method with respect to this parameter by investigating through graphical tools the effect of varying the value of ϵ.
Once the empirical variogram has been estimated for each Voronoi neighborhood, we fit a valid model by using a least square approach, as described in Chapter 1. In the following, we denote by $\hat{\gamma}^b(\mathbf{h};k,\epsilon)$ the fitted variogram within the k-th

Voronoi neighborhood at the b-th iteration.

Step 3. Kriging on the grid

We now aim to predict $Z_{\mathbf{s}_0}$ at an unsampled location $\mathbf{s}_0 \in D$. Let $V^b_{\mathbf{s}_0} \subset D$ be the Voronoi tile $\mathbf{s}_0$ belongs to at the b-th bootstrap iteration and let $\hat{\gamma}^b(\mathbf{h}; k, \epsilon)$ be the corresponding estimated variogram . The analysis is now restricted to $V^b_{\mathbf{s}_0}$ and results in the *weak* predictor $Z^{*b}_{\mathbf{s}_0}(\mathbf{Z})$. We use a Kriging procedure, and thus look for the Best Linear Unbiased Predictor (BLUP) which is defined as linear combination of the data located within $V^b_{\mathbf{s}_0}$, according to

$$Z^{*b}_{\mathbf{s}_0}(\mathbf{Z}) = \sum_{i=1}^{n} \lambda^b_i Z_{\mathbf{s}_i} \mathbb{1}\{\mathbf{s}_i \in V^b_{\mathbf{s}_0}\} \tag{4.3}$$

where the vector of weights $\boldsymbol{\lambda}^b = (\lambda^b_1, \dots, \lambda^b_n)$ is solution of the linear problem (1.11) (or its extension to object data, discussed in Menafoglio et al. [2013]). Within each Voronoi neighborhood we assume stationarity, therefore we apply an Ordinary Kriging approach. The term $\mathbb{1}\{\mathbf{s}_i \in V^b_{\mathbf{s}_0}\}$ is an indicator function which is 1 if $\mathbf{s}_0$ and $\mathbf{s}_i$ are in the same Voronoi tile at the b-th iteration, 0 otherwise. If the $\mathbb{1}\{\mathbf{s}_i \in V^b_{\mathbf{s}_0}\} = 0$, then the weight λ^b_i has 0 value. This means that, at this specific bootstrap iteration, the observation $Z_{\mathbf{s}_i}$ is not used for the estimate of $Z_{\mathbf{s}_0}$, since the two locations $\mathbf{s}_0$ and $\mathbf{s}_i$ are not in the same Voronoi neighborhood. Even if the predictor $Z^{*b}_{\mathbf{s}_0}(\mathbf{Z})$ is defined as function of the data, we must highlight the fact that it can be defined only if a partition $P = P_b$ of the domain has been previously defined: $Z^{*b}_{\mathbf{s}_0}(\mathbf{Z}) := Z^{*}_{\mathbf{s}_0}(\mathbf{Z}; P = P_b)$.

The step 1.-3. are repeated B times. Every time, a bootstrap Kriging predictor $Z^{*b}_{\mathbf{s}_0}(\mathbf{Z})$ is computed, conditionally on the defined random domain partition $P_b = \{V(\mathbf{c}^b_k | \Phi^b_K)\}^K_{k=1}$. At each iteration the set of K local maps together defines a *weak Kriging map.* In the following final step the B *weak Kriging maps* will be aggregated to obtain a final *strong Kriging map.*

4.1.2 Aggregation stage

The aggregation step consists in aggregating the B Kriging predictors $Z^{*b}_{\mathbf{s}_0}(\mathbf{Z})$, with $b = 1, \dots, B$, into an unique final predictor $Z^{*}_{\mathbf{s}_0}(\mathbf{Z})$. Since each bootstrap predictor can be defined only if a Voronoi tessellation of the domain is given, the final predictor will depend on the B tessellations: $Z^{*}_{\mathbf{s}_0}(\mathbf{Z}) := Z^{*}_{\mathbf{s}_0}(\mathbf{Z}; P)$ where $P = \{P_b\}^B_{b=1}$ indicates the collection of the realizations of the random domain partitions performed along the B replicates. The replicates of the same predictor have the final aim to define a predictor $Z^{*}_{\mathbf{s}_0}(\mathbf{Z})$ which is the discretized version of

$$\mathbb{E}_P[Z^{*}_{\mathbf{s}_0}(\mathbf{Z}; P)] = \int_{\mathcal{P}} Z^{*}_{\mathbf{s}_0}(\mathbf{Z}; p) dP(p)$$

where $\mathcal{P}$ is the set of all possible partitions of the domain.
The aggregation can be performed in different ways. We here investigate (a) a simple average aggregation; (b) a kernel-weighted average aggregation; and (c) a variance-weighted average aggregation of the B weak analyses.

Equal aggregation

This is the simplest way of aggregating the results coming from the bootstrap replicates. Indeed, the final predictor for $Z_{\mathbf{s}_0}$ is a simple average of all the predictors obtained throughout the B iterations

$$Z^*_{\mathbf{s}_0}(\mathbf{Z}) = \frac{1}{B}\sum_{b=1}^{B} Z^{*b}_{\mathbf{s}_0}(\mathbf{Z}). \tag{4.4}$$

With this type of aggregation, all the predictors computed during the boostrap stage have the same weights in the definition of the final predictor.

Kernel aggregation

For each point of the grid $\mathbf{s}_0$ and at each bootstrap iteration we keep the additional information regarding the value of the kernel function

$$K^b_\epsilon(\mathbf{s}_0) := K_\epsilon(\mathbf{s}_0, \mathbf{c}^b)$$

where $\mathbf{c}^b$ indicates the coordinates vector of the Voronoi nucleous to which $\mathbf{s}_0$ is assigned at the b-th bootstrap iteration, and $K_\epsilon(\cdot,\cdot)$ is the kernel function defined in (4.2). This value gives an indication of how far the target point $\mathbf{s}_0$ is from the center of the corresponding neighborhood. This center is also the location where the local variogram estimate is computed. The weighted average is defined according to

$$Z^*_{\mathbf{s}_0}(\mathbf{Z}) = \frac{\sum_{b=1}^{B} K^b_\epsilon(\mathbf{s}_0)\cdot Z^{*b}_{\mathbf{s}_0}(\mathbf{Z})}{\sum_{b=1}^{B} K^b_\epsilon(\mathbf{s}_0)}. \tag{4.5}$$

With a kernel aggregation, the final predictor is more influenced by the predictors of those repetitions in which the target location is near the center of the corresponding Voronoi tile.

Variance aggregation

An additional information kept for each bootstrap iteration and for each grid point $\mathbf{s}_0$ regards the Kriging variance. Given the Kriging predictor $Z^{*b}_{\mathbf{s}_0}(\mathbf{Z})$ the b-th Kriging prediction error is defined as

$$\epsilon^{*b}_{\mathbf{s}_0}(\mathbf{Z}) = Z_{\mathbf{s}_0} - Z^{*b}_{\mathbf{s}_0}(\mathbf{Z}).$$

Its variance $\sigma^2_{OK}(\mathbf{s}_0)^b$ is called *Kriging variance* and, since an Ordinary Kriging has been performed within each neighborhood, it is defined by the equation (1.14). The Kriging variance in a point $\mathbf{s}_0$ gives an indication of how far the point is from the nearest data: a small variance value indicates a more reliable prediction. Therefore, the aggregation weight may be set to be proportional to the inverse of the Kriging variance

$$Z^*_{\mathbf{s}_0}(\mathbf{Z}) = \frac{\sum_{b=1}^{B}[\sigma^2_{OK}(\mathbf{s}_0)^b]^{-1} \cdot Z^{*b}_{\mathbf{s}_0}(\mathbf{Z})}{\sum_{b=1}^{B}[\sigma^2_{OK}(\mathbf{s}_0)^b]^{-1}}. \tag{4.6}$$

Using the variance aggregation, one gives more importance to more precise predictors obtained throughout the bootstrap repetitions.

It is important to observe that, in the case of weighted average, each target point is weighted according to its own weight, which is not necessarily the same as those of the neighbors. This weighting strategy could be useful to account (and smooth down) the discontinuities generated by the domain partitions.

```
Algorithm. RDDs for Kriging prediction.

Initialization.
    Set parameters: $K$, $B$, a valid variogram model, the kernel function $K_\epsilon(\cdot,\cdot)$ with its bandwidth $\epsilon$, a metric $d(\cdot,\cdot)$ for the spatial domain $D$, and a grid of prediction over the spatial domain.
    Choose an aggregation method among: (a) simple average; (b) a weighted average with weights proportional to the kernel value; (c) weighted average with weights inversely proportional to the Kriging variance.

Bootstrap step.
    for $b := 1$ to $B$ do

    Step 1. Randomly generate a set of nuclei $\Phi_K^b = \{\mathbf{c}_1^b, \dots, \mathbf{c}_K^b\}$ among the observed sites in $D$: for $k = 1, \dots, K$, $\mathbf{c}_k^b \sim \mathcal{U}(\mathbf{s})$ where $\mathcal{U}$ is the uniform distribution on the set of the $n$ sampled locations $\mathbf{s} = (\mathbf{s}_1, \dots, \mathbf{s}_n)$. Obtain a random Voronoi tessellation of $D$ defined by the collection of Voronoi tiles $\{V(\mathbf{c}_k^b|\Phi_K^b)\}_{k=1}^K$ by assigning each site $\mathbf{s} \in D$ to the nearest nucleus $\mathbf{c}_k^b$ according to the specified metric $d(\cdot,\cdot)$;
    Step 2. For each neighborhood: perform a non-parametric geographically weighted estimate of the empirical variogram $\hat{\gamma}_\epsilon(\mathbf{h};k)^b$ by using a kernel function $K_\epsilon(\cdot,\cdot)$. Fit the parametric valid model to the empirical estimate.
    Step 3. For each neighborhood: perform Kriging on the portion of the grid belonging to the considered neighborhood, according to the defined Voronoi tessellation, based on the variogram structure associated with the neighborhood.
    Results of the b-th iteration: $K$ local maps, which jointly define a global map, called weak Kriging map.

    end for.

Aggregation step.
    Aggregate the $B$ weak Kriging maps according to the chosen criterion and obtain a final strong Kriging map.
```

Figure 4.3: *Pseudocode scheme of the Random Domain Decompositions (RDDs) algorithm for Kriging prediction over the grid domain.*

4.2 Simulated example

The proposed RDDs procedure for Kriging prediction has been tested on some simulated examples to explore its performances.
Following a similar approach of Kim et al. [2005], we firstly tested the method on a simple domain where a stationary random process has been generated, and then we considered a more complex simulation by considering a non stationary random process.

4.2.1 Stationary process

In the first simulation example we tested the method by using a single smooth Gaussian process (mean 0 and standard deviation 1). We used an exponential covariance function with sill 1, nugget 0 and range 2 (i.e, it is the same setting as the simulation example used in Kim et al. [2005]). The data are distributed over a simple square domain. The x- and y-coordinates of the sampled locations are in the interval $[-1, 1]$. Figure 4.4 shows the reference realization of the generated random process.
We performed Kriging on the domain by using the RDDs method with $B = 100$, a Gaussian kernel with $\epsilon = 1$ and an "Equal" aggregation. Due to the convexity of the domain, we used an Euclidean metric for the definition of the Voronoi tessellations and the kernel weights. The domain has been partitioned into $K = \{1, 2, 4, 8, 16, 32, 64\}$ sub-regions. The case $K = 1$ is the simple case when stationarity assumption is assumed on the whole domain, therefore no partition is computed and the Ordinary Kriging technique is used to obtain the desired

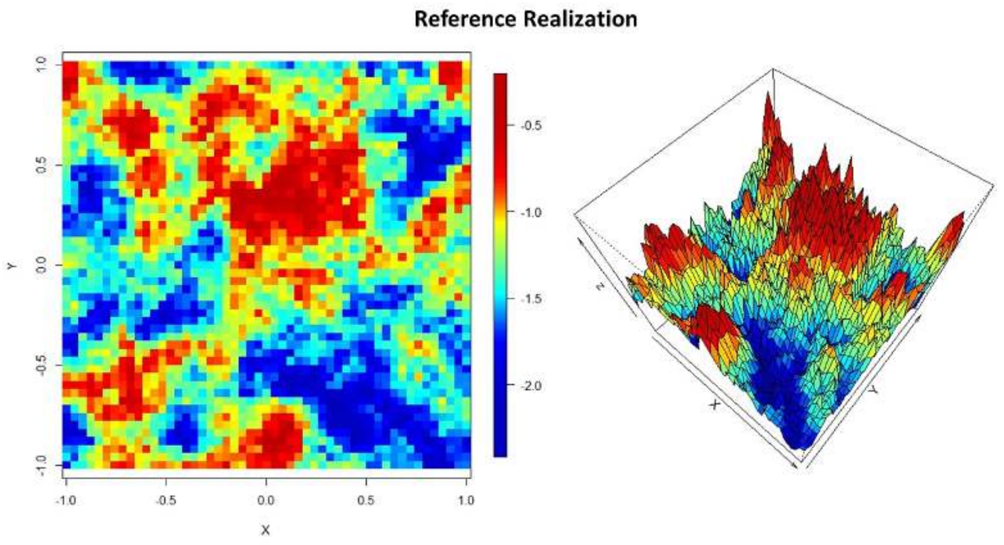

Figure 4.4: *Stationary process: reference realization (example from Kim et al. [2005]).*

predictions. Moreover, the case of $K = 1$ does not require the use of any kernel function. Therefore, the empirical variogram is simply estimated via method of moments according to equation (1.5).
Since we are in simulation framework, we can compute the mean square prediction error (MSPE) and use it to understand how the values of the parameters influence the results of the method. Given all the parameters, the (normalized) MSPE is computed as

$$\text{MSPE} = \frac{\sum_{\mathbf{s}_0 \in \mathcal{G}} (Z^*_{\mathbf{s}_0}(\mathbf{Z}) - Z_{\mathbf{s}_0})^2}{\sum_{\mathbf{s}_0 \in \mathcal{G}} Z^2_{\mathbf{s}_0}} \tag{4.7}$$

where $\mathcal{G}$ is the set of grid points where we want to perform Kriging prediction. In this simple stationary case we are interested in understanding if the Ordinary Kriging and the Kriging via RDDs provide equivalent results. We repeated the whole analysis $M = 50$ times. At each repetition we used a different set of observations randomly selected among the grid points within the domain. Figure 4.5 shows the boxplots of the MSPEs obtained through the M repetitions for each value of K and for each dimension of the used dataset (in particular we set $nsub = 100, 250, 500$). The results of Ordinary Kriging and Kriging via RDDs are very similar in terms od MSPEs, in particular when $nsub = 250$ or $nsub = 500$. When $nsub = 100$, if K assumes high values, then the MSPEs increase on average. This is a consequence of the fact that when K assumes values close to the number of available observations, there may be some neighborhoods wherein the local Kriging analysis is based on a single datum (i.e., the Voronoi center).

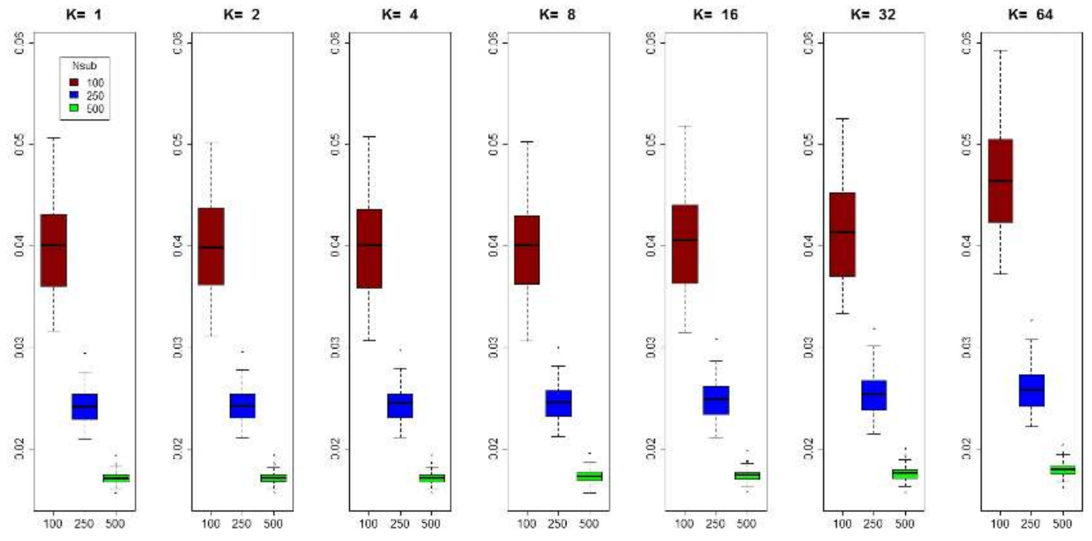

Figure 4.5: *Stationary simulation example. MSPEs boxplots: different $nsub$ values.*

nsub	**K**						
	1	2	4	8	16	32	64
100	0.0402	0.0401	**0.0400**	0.0400	0.0405	0.0418	0.0467
250	**0.0243**	0.0244	0.0245	0.0249	0.0254	0.0254	0.0259
500	**0.0171**	0.0171	0.0172	0.0173	0.0175	0.0177	0.0180

Table 4.1: *Stationary simulation example. Different $nsub$: Average of the MSPEs over the $M = 50$ repetitions.*

4.2.2 Non-stationary process

The second simulation example comes from the literature, in particular it follows the same model used by Kim et al. [2005] to test their partitioning method applied to the analysis of a non-stationary process. In this simulation two Gaussian processes, with a large difference in means and different covariance structures, have been generated over a simple square domain. From Figure 4.6 we can see that the domain is divided into two homogeneous sub-regions. The process occupying the bottom region of the domain (with negative ordinates) has been generated by using an exponential covariance with range 4 and a Gaussian process with mean 0 and standard deviation 1. The second process, in the top region of the domain (positive ordinates) has been generated by using an exponential covariance with range 1.5 and a Gaussian process with mean 10 and standard deviation 1. For both processes the sill and the nugget values are 1 and 0, respectively.

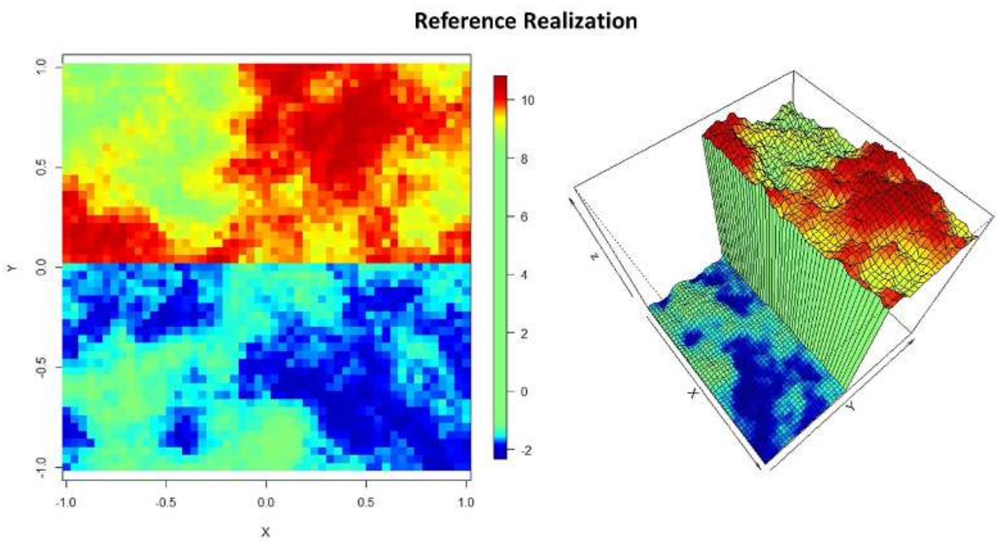

Figure 4.6: *Non-stationary process: reference realization (example from Kim et al. [2005])*

This simple example does not show any complexity associated with the domain or to object data, but the nature of the process does not allow us to assume stationarity over the whole domain. We know that there exists an optimal partition of the domain which divides the region into two disjoint sub-regions separated by the axis of ordinates. Indeed, using a Bayesian approach, the partitioning method proposed in Kim et al. [2005] finds these two regions more that 95% of the times in the posterior samples. Nevertheless, we here aim to test the performances of our general method on a non-stationary case for which the local stationarity assumption is not verified everywhere (i.e., it is not valid at the boundary between the sub-regions).
We tested our method on this simple domain and, since there are no holes or barriers, its random partition, as well as the kernel weights, can be computed by simply using the Euclidean metric. The same analysis has been performed by using different values of $K = \{1, 2, 4, 8, 16, 32, 64\}$ to explore how this parameter influences the results of the method. In all the cases, for this analysis we chose an exponential valid model for fitting the empirical variogram and a Gaussian kernel and we performed $B = 100$ bootstrap iterations. We are interested in seeing how the Kriging prediction computed via RDDs at unsampled locations is influenced by the parameter K, the number of available observations, the kernel bandwidth ϵ and the type of aggregation.
In Figure 4.7 we can observe the result of the analysis performed with fixed values of $K = 8$ on a dataset of 250 observations, by setting the bandwidth parameter $\epsilon = 1$ and using a simple equal aggregation. We can observe that the points where

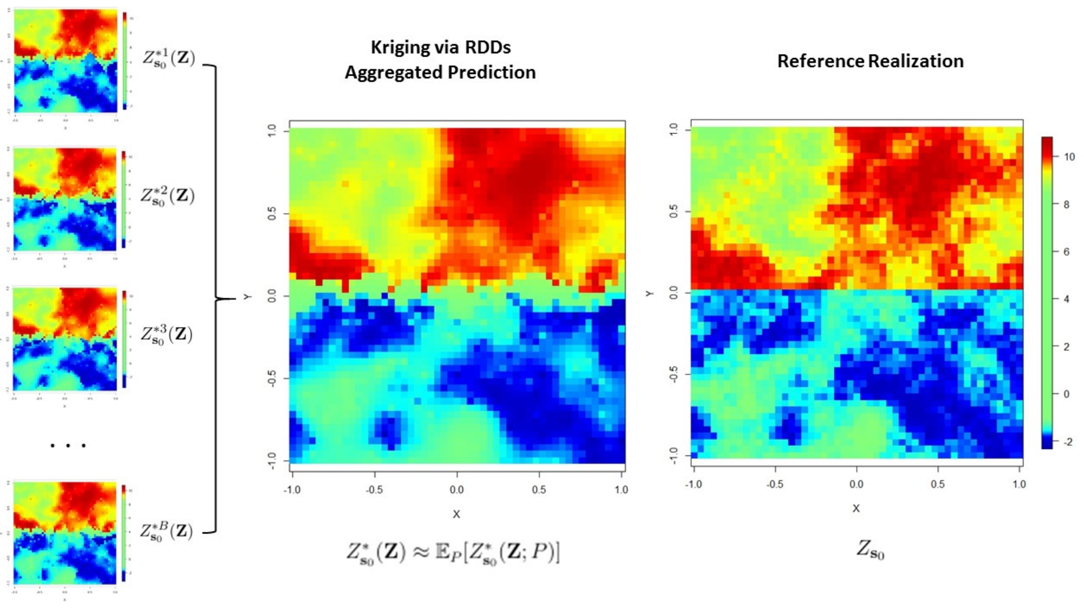

Figure 4.7: *Kriging prediction via Random Domain Decompositions.*

the algorithm gives less precise results are those close to the axis $Y = 0$, which represents the separation line of the two sub-region with different processes.

4.2.3 Number of observations

Before performing the analysis, we need to consider the number of observations which are available for our study. This is an important information for the initialization of the parameter K. If we have a small number of observations we cannot compute a dense partition of the domain, because we can meet the problem of having very few observations within each Voronoi neighborhood, with the consequence of bad predictions. In Figure 4.8 we can see three examples of datasets with different number of sampled locations. The left panel of the figure shows an example of dataset of size 100, the central panel shows an example of dataset with 250 observations, the right one is an example of 500 observations.
We repeated the whole Kriging-RDDs analysis $M = 50$ times. For this simulation we set $\epsilon = 1$ and we aggregated the bootstrap predictors according to an "Equal Aggregation". Every time a different $nsub$-dimensional set of points is used as dataset. These data points were randomly chosen among all the available grid points. Such simulation enables us to plot the boxplots of the MSPEs obtained for each subsample. In Figure 4.9 the red boxplots refer to datasets with $nsub = 100$ observations, the blue ones to datasets with $nsub = 250$ and the green ones to dataset with $nsub = 500$. As we can expect, the MSPEs related to small datasets have higher values with respect to the others. Furthermore, as in the stationary case, the MSPEs values related to $nsub = 100$ increase on average when the parameter K assumes high values. The analyses performed on datasets of dimensions 250 and 500 show smaller values, on average, of MSPEs for each $K > 1$ with respect to the MSPEs obtained with simple Ordinary Kriging on the

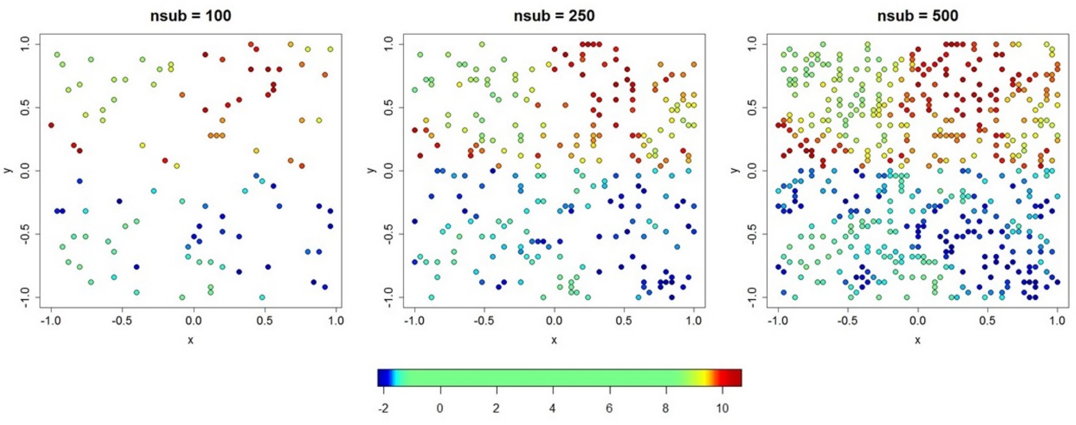

Figure 4.8: *Examples of sampled points:* $nsub = 100, 250, 500$, *respectively.*

whole domain (i.e., the case $K = 1$). Another important observation concerns the fact that we do not see a minimum for the MSPEs obtained when $K = 2$, which would correspond to the optimal partition of the domain. In fact, we are no looking for such an optimal partition, but for a partition which makes viable the stationary assumption within each neighborhood, ensuring that each sub-region contains a sufficient number of observations upon which it is possible to build the Kriging local analysis.

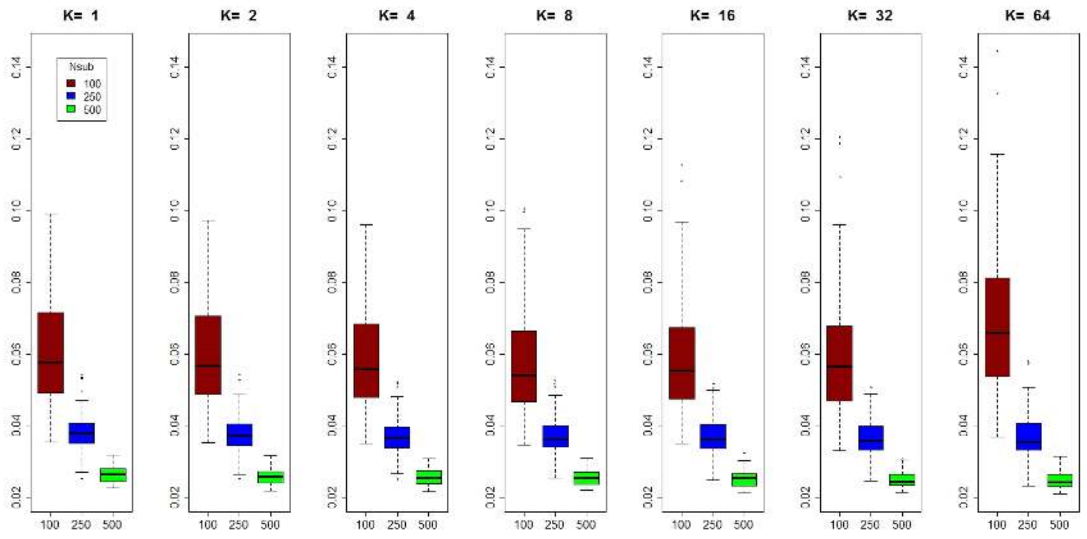

Figure 4.9: *Non-stationary simulation example. MSPEs boxplots: different $nsub$ values.*

nsub	**K**						
	1	2	4	8	16	32	64
100	0.0607	0.0595	0.0585	**0.0581**	0.0596	0.0614	0.0707
250	0.0385	0.0379	0.0375	0.0373	0.0371	**0.0368**	0.0375
500	0.0264	0.0259	0.0256	0.0255	0.0254	0.0251	**0.0249**

Table 4.2: *Non-stationary simulation example. Different $nsub$: Average of the MSPEs over the $M = 50$ repetitions.*

4.2.4 Kernel width

Another important parameter is the bandwidth ϵ defining the kernel function. For this case we performed the analysis for $\epsilon = \{0.5, 1, 2\}$. Figure 4.10 shows the differences among the kernel values when different values of the bandwidth

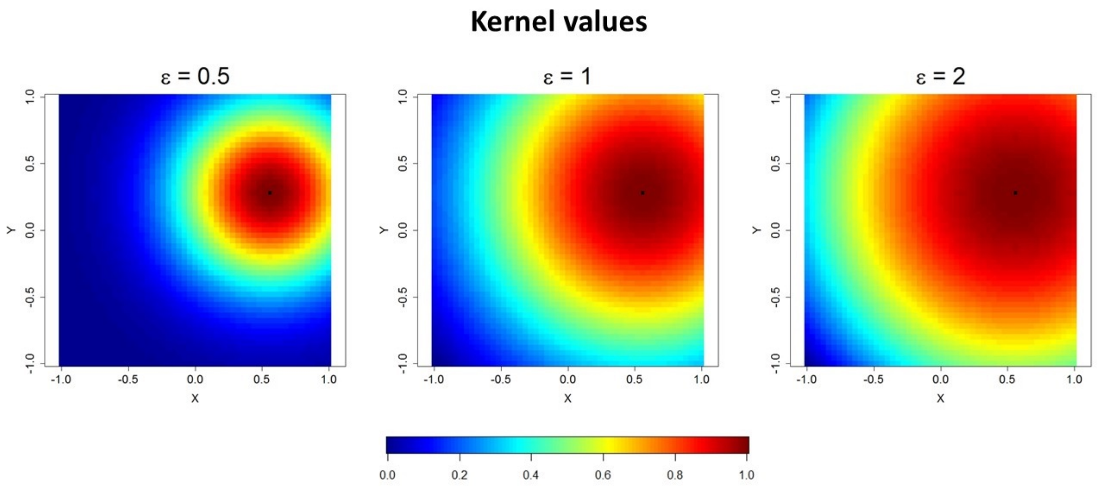

Figure 4.10: *Kernel values for different values of ϵ.*

parameter are defined. For each grid point $\mathbf{s}_0 \in D$ the value $K_\epsilon(\mathbf{s}_0, \mathbf{c})$ has been plotted. The kernel was defined by using the Euclidean metric, therefore the value $K_\epsilon(\mathbf{s}_0, \mathbf{c})$ is the same for points at the same distance from $\mathbf{c}$. The right, the central and the left panels of Figure 4.10 refer to a kernel function with $\epsilon = 0.5$, $\epsilon = 1$ and $\epsilon = 2$, respectively. As we can observe, small values of ϵ give high weights only to points very close to the centers (i.e., the red zone, that is the area of points associated with the highest kernel values, are small neighborhoods around the center), while when ϵ increases also points far away from the center have high kernel values.

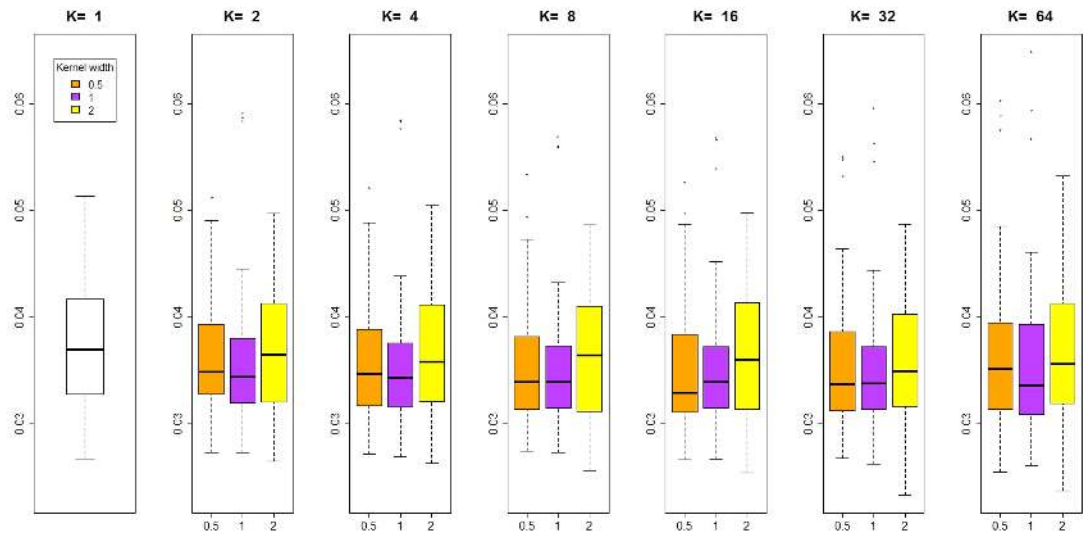

Figure 4.11: *Non-stationary simulation example. MSPEs boxplots: different ϵ values.*

ϵ	K						
	1	2	4	8	16	32	64
0.5	0.0371	0.0362	0.0357	0.0354	**0.0351**	0.0355	0.0364
1	0.0371	0.0361	0.0358	0.0355	0.0354	**0.0354**	0.0362
2	0.0371	0.0366	0.0364	0.0362	0.0362	**0.0359**	0.0368

Table 4.3: *Non-stationary simulation example. Different ϵ: Average of the MSPEs over the $M = 50$ repetitions.*

As before, we repeated the whole Kriging-RDDs analysis $M = 50$ times. Figure 4.11 shows the boxplots of the MSPEs obtained throughout the M repetitions, for each fixed value of K and ϵ. Every time the analysis was based on a different dataset of dimension $nsub = 250$. The final predictor was defined according to a simple average aggregation. We can observe that, on average, the cases with $K > 1$ show smaller MSPEs values than the case $K = 1$. In this latter case, since no partition has been computed, we did not use any kernel function, but the unique variogram estimate has been defined according to equation (1.5) and the Ordinary Kriging has been computed. Figure 4.11 does not show obvious differences in the MSPEs values associated to different values of the parameter ϵ. We can only observe a smaller dimension of boxplots associated with the cases when $\epsilon = 1$. For this reason we opt for using this value for the following analyses, without performing an in-depth study for its initialization.

4.2.5 Aggregation type

The final analysis regards the different types of aggregation which can be used for the definition of the final predictor. In particular we used an "Equal" aggregation according to the equation (4.4), a "Kernel" aggregation defined by the equation (4.5) and a "Variance" aggregation defined by the equation (4.6). Also in this case we repeated the whole Kriging-RDDs analysis $M = 50$ times. The boxplots of the MSPEs are described in Figure 4.12. Also in this case we do not observe an aggregation type which prevails on the others, they give similar results in term of MSPE.

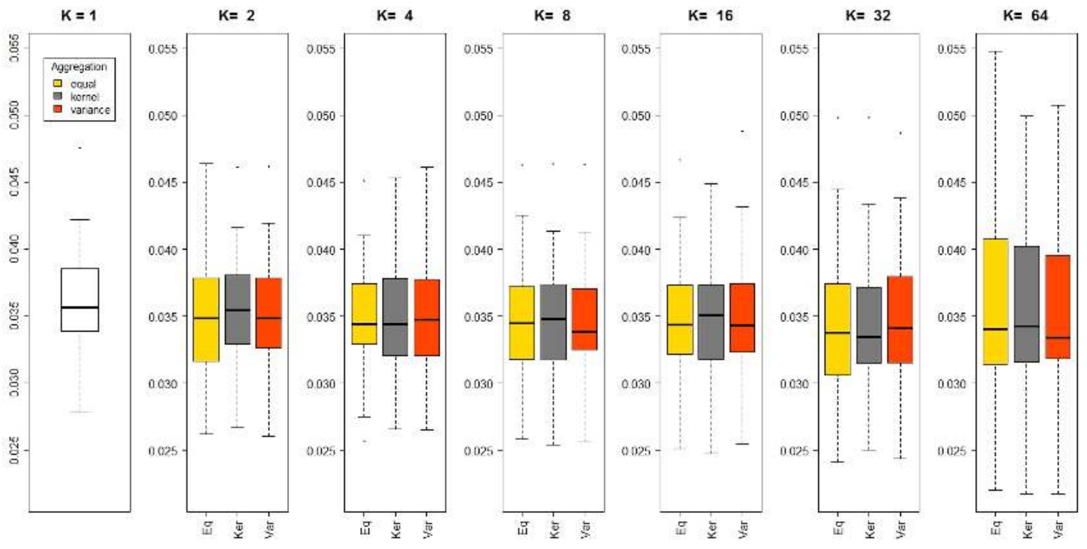

Figure 4.12: *Non-stationary simulation example. MSPEs boxplots: different aggregation types.*

Aggregation	K						
	1	2	4	8	16	32	64
Equal	0.0360	0.0352	0.0349	0.0348	0.0348	**0.0347**	0.0357
Kernel	0.0360	0.0355	0.0350	0.0346	0.0348	**0.0344**	0.0355
Variance	0.0360	0.0354	0.0350	**0.0346**	0.0350	0.0347	0.0354

Table 4.4: *Non-stationary simulation example. Different aggregation types: Average of the MSPEs over the $M = 50$ repetitions.*

Chapter 5

Kriging-RDDs applied to irregularly shaped domain

Geostatistics is based on the intuitive idea that locations which are close together provide more similar information than locations that are far apart. The Kriging predictor uses weights that are calculated according to the value of the variogram, which is usually defined (in the stationary and isotropic case) as a function of the Euclidean distance. Methods for spatial data analysis have been typically applied to convex subsets of $\mathbb{R}^2$ or $\mathbb{R}^3$, where the Euclidean distance is the natural argument for defining correlation functions. Nevertheless, there are some situations where the study domain shows complex characteristics, which make the Euclidean distance a non appropriate metric for describing the proximity relationships. In particular, our focus is on situations where data are distributed over irregularly shaped domains with complex boundaries, barriers, holes or strong concavities. The first idea may be the use of a non-Euclidean metric for the variogram estimate and the consequent Kriging analysis. However, the main problem is that, in general, there is no guarantee that the set of valid models commonly used in Geostatistics (i.e., the models such as (1.6)-(1.9) described in Chapter 1, Section 1.1.3) will remain valid when they are used with distance measures other than the Euclidean distance.

5.1 Isometric embedding

Recall that given a random field $\{Z_{\mathbf{s}}, \mathbf{s} \in D\}$, the functions covariogram $C(\cdot)$ and variogram $\gamma(\cdot)$ must fulfill specific properties. In particular the covariogram must be a positive definite function, while the variogram must be a conditionally negative definite function. Even if, in general, we do not have proofs about the validity of the models used in Euclidean framework when they are based on non-Euclidean metrics, Curriero [2006] proved that there are some metrics *equivalent* to the Euclidean one such that, when used with existing covariance and variogram functions, positive or conditionally negative definiteness are guaranteed. These metrics are said to be *isometrically embeddable.*

Definition 5.1. Let $d_{ij} = d(\mathbf{s}_i, \mathbf{s}_j)$ represent the distance between points $\mathbf{s}_i$ and $\mathbf{s}_j$ of some metric space (D, d). The metric space is said to be *isometrically embedded* in a Euclidean space of dimension n^* if there exist points $\mathbf{s}_i^*$ and $\mathbf{s}_j^*$ and a function $\phi : \mathbb{R}^n \to \mathbb{R}^{n^*}$ such that

$$d_{ij} = d(\mathbf{s}_i, \mathbf{s}_j) = \|\mathbf{s}_i^* - \mathbf{s}_j^*\|$$

for all $\mathbf{s}_i, \mathbf{s}_j \in D$ and $\phi(\mathbf{s}) = \mathbf{s}^*$, where $\|\cdot\|$ is the Euclidean norm.

If a non-Euclidean distance function, satisfying the *metric* properties, is embeddable, then this distance function used with existing covariance and variogram functions will retain the positive and conditionally negative properties, hence these functions are valid in the embedding dimension. Appendix 5.A describes the theorems used to check the isometric embeddable property of a generic metric d. In particular, Wells and Williams [1975] proved that a metric d is embeddable if and only if d^2 is conditionally definite. Moreover, Appendix 5.A shows a simple theorem used to check the isometric embeddable property for the class of norm distances (Theorem 5.3).
Nevertheless, in the more general case, if d is not a norm function, establishing the validity of $C(d)$ or $\gamma(d)$ may be mathematically challenging and methods for dealing with such situations has not received much attention.

5.2 Existing approaches for spatial processes over complex domains

The need of finding a non-Euclidean distance for the description of proximity relationships among spatial data is an essential problem for understanding the spatial distributions of variables over domains like estuaries.
Little et al. [1997] suggested the idea of measuring distances between sites in estuarine streams “as the fish swims” (i.e., the length of the shortest in-water path between two sites) instead of “as the crow flies” (i.e., the Euclidean distance). They provided several comparisons between the results obtained with kriging analyses based on the Euclidean distance and those based on the in-water distance, without dealing with the validity problem.
Rathbun [1998] proposed the use of a *water distance* (i.e., the shortest distance that may be traversed entirely over water) as measure of the proximity relationships among locations within an estuary. Furthermore, he dealt with the additional complexity related to the fact that some environmental variables may take deterministic values at estuarine boundaries. For these reasons, he

based the Kriging analysis on the estimate of a function called *spatial dispersion* $2\delta(\mathbf{s}_i, \mathbf{s}_j) = \mathrm{Var}(Z_{\mathbf{s}_i} - Z_{\mathbf{s}_j})$. It is defined as the variogram function but, under intrinsically stationarity assumption, it does not only depend on $\mathbf{h} = \mathbf{s}_i - \mathbf{s}_j$, but it also depends on the distance between the points $\mathbf{s}_i$, $\mathbf{s}_j$ and the estuarine boundaries where the random variable $Z_{(\cdot)}$ assumes fixed values.
Krivoruchko and Gribov [2002] suggested a moving window kernel approach to estimate the covariance model that is not subject to the same criterion of non-negative definiteness. The modeling of the process is based on a cost weighted distance, that computes the cost of travelling from one cell of a grid to the next one. The covariance estimate was based on an asymmetric kernel, which is able to takes into account the presence of barriers within the domain.
Finally, Løland and Høst [2003] used multidimensional scaling to define an Euclidean approximation of the water distance. This method consists in defining a triangular grid covering the complex domain of interest that is used for computing the water distances as length of the shortest paths on the graph defined by the triangulation. These water distances are then used to remap the sample locations into a new Euclidean space where the (Euclidean) distances approximate the obtained water distances and the validity of the variogram models is guaranteed. One drawback of this approach is that it is dependent on the sample design, in the sense that adding and/or removing a location can change the distance approximation elsewhere.
The basic idea of our approach is inspired by this latter described method, since we use graph-based metrics to better approximate the spatial relationships among points. To avoid the need to check for the validity of the variogram models computed by using non-Euclidean metrics (which could require ad hoc situations in case of non-validity), we opt to use a graph-based metric only for defining the random domain decompositions and the kernel weights. The variogram estimate will be grounded on the Euclidean distance instead, that is used to define the adjacency relations among points located within the same Voronoi neighborhood.

5.3 Triangulation of planar point sets

Before performing the Kriging analysis on a complex domain, we want to well define the spatial adjacency and neighbourhood relation among the data. For this reason, we firstly construct a neighborhood relational graph representing the spatial adjacency of objects distributed over a domain with constraints (e.g., holes, irregular boundaries or barriers).
Given a set of points, a possible way to define a graph is to perform a triangulation of these points. In general, this can be done in many different ways and it does not necessarily exist a "most appropriate" triangulation. Nevertheless, some triangulations look more natural than others. In particular, triangulations containing triangles with small angles can badly approximate the spatial adjacency

among data. Therefore we focused on triangulations that maximize the smallest angle.
Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of points in the plane. The definition of a triangulation of P is based on the *maximal planar subdivision* concept [de Berg et al., 2000a]. A maximal planar subdivision is a subdivision $\mathcal{S}$ such that no edge connecting two vertices can be added to $\mathcal{S}$ without destroying the planarity property of the graph (i.e., a graph is planar if it can be drawn in a plane without graph edges crossing). A *triangulation* $\mathcal{T}$ of P is defined as the maximal planar subdivision whose vertex set is P. Each face of this triangulation, except the unbounded one, is a triangle. This is a consequence of the fact that a bounded face is a polygon and any polygon can be triangulated [de Berg et al., 2000b]. Moreover, we can observe that any segment connecting two consecutive points on the boundary of the convex hull of P is an edge in any triangulation $\mathcal{T}$. This implies that the union of the bounded face of $\mathcal{T}$ is always the convex hull of P and the unbounded face is always the complement of the convex hull.
The number of triangles is the same for every triangulation of P and this number depends on the number of points in P that are on the boundary of its convex hull. Let $\mathcal{T}$ be a triangulation of P, and suppose it has m triangles. The set of $3m$ angles of the triangles of $\mathcal{T}$, sorted by increasing values, is called *angle-vector*: $A(\mathcal{T}) = (\alpha_1, \alpha_2, \ldots, \alpha_{3m})$. Let $\mathcal{T}'$ another triangulation of P, with its angle-vector $A(\mathcal{T}') = (\alpha'_1, \alpha'_2, \ldots, \alpha'_{3m})$. The angle-vector of $\mathcal{T}$ is said to be larger than the angle-vector of $\mathcal{T}'$ if there exists an index i, with $1 \leq i \leq 3m$, such that

$$\alpha_j = \alpha'_j \quad \text{for all } j < i \quad \text{and} \quad \alpha_i > \alpha'_i.$$

If this condition is satisfied $A(\mathcal{T})$ is *lexicographically* larger than $A(\mathcal{T}')$ and we denote it as $A(\mathcal{T}) > A(\mathcal{T}')$.
A triangulation $\mathcal{T}$ is called *angle-optimal* if $A(\mathcal{T}) \geq A(\mathcal{T}')$ for all triangulation $\mathcal{T}'$ of P. A good triangulation of a set of points has the property of having an angle-vector as large as possible. For this reason, among the several ways which can be adopted for the triangulation of a set of points P, our focus is on the Delaunay triangulation.

5.3.1 Delaunay Triangulation

Let P be the set of n points in the plane. Let $\mathrm{Vor}(P)$ be the Voronoi diagram of P, that is the Voronoi tessellation of the plane whose nuclei are the n points in P. The dual graph $\mathcal{G}$ of the Voronoi diagram is a graph with a node for every site $p \in P$ and it has an arc between two nodes if the corresponding Voronoi cells share an edge. The straight-line embedding of the graph $\mathcal{G}$ replaces the arc connecting p and q with the segment $\overline{pq}$. This embedding is called *Delaunay graph* of P and it is indicates as $\mathcal{DG}(P)$. An example of a Voronoi diagram of P and its Delaunay graph is showed in figure 5.1.

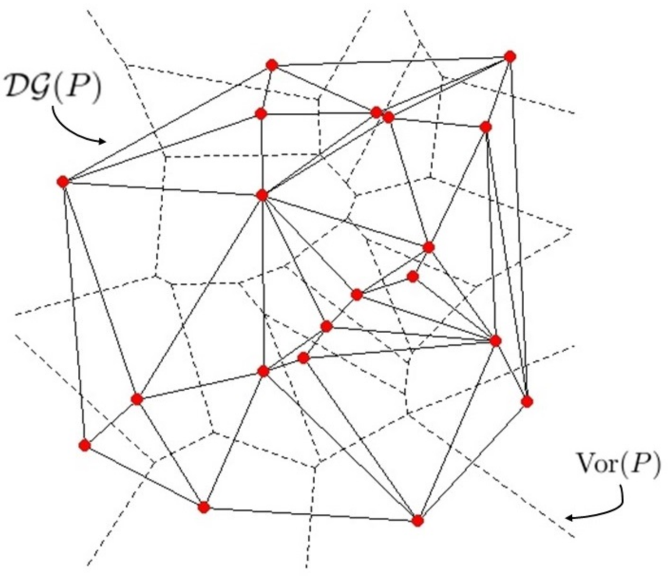

Figure 5.1: *Example: Voronoi diagram and Delaunay graph.*

The name Delaunay comes from the Russian mathematician Boris Nikolaevich Delone (*Delaunay* is the French version of his surname), who proposed this type of triangulation in Delaunay [1934]. The Delaunay graph $\mathcal{DG}(P)$ is often called *Delaunay triangulation* since, if there are not four points in P which are co-circular (i.e., four points which lie on the same circle), then all the vertices of the Voronoi diagram have degree three and, hence, all bounded faces are triangles.
The success of this kind of triangulation and its recurring use in many applications, is due to its several interesting properties. The most important property is the so called *empty circumcircle* [de Berg et al., 2000a]:

Theorem 5.1. Let P be a set of points in the plane, and let $\mathcal{T}$ be a triangulation of P. Then $\mathcal{T}$ is a Delaunay triangulation of P if and only if the circumcircle of any triangle of $\mathcal{T}$ does not contain a point of P in its interior.

The most important consequence of the empty circumcircle property is that, given a set of points P, any angle-optimal triangulation of P is a Delaunay triangulation. Furthermore, the Delaunay triangulation of P maximizes the minimum angle over all the triangulations of P. The need of defining a triangulation without very skinny triangles comes from the goal that we want to reach, i.e., using the Delaunay graph to compute a graph-based metric which must describe the spatial adjacency relations among data. Indeed, the presence of very stretched triangles may not properly be used to describe the obstacles located within the domain and, consequently, this may badly describe the spatial adjacency relations among the data.

In our application, the set P that we use for the Delaunay triangulation of the domain is the set of n observed locations $\{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_n\}$ in D. Without any other information regarding the domain, the Delaunay triangulation of the set of sample locations does not take into account the presence of constraints. For example, if a hole exists within the domain, it must be bounded by segments and its *empty* space should not be triangulated. The addition of information regarding holes, boundaries and obstacles, poses our attention on the definition of *Constrained Delaunay triangulation* (CD-T) [Lin et al., 2013].
Given a set of spatial objects in the plane together with a set of spatial obstacles and facilitators, the constrained Delaunay triangulation is an undirected graph representing the adjacency and neighborhood relations among the objects. It is an approximated triangulation of vertices, since the obstacles are included in the graph as some barrier edges that obstruct the connection of the objects, and the facilitators are included in the graph as edges that connect the objects that are broken by the obstacles. Even if it is an approximation of the Delaunay triangulation, the CD-T is as close as possible to the unconstrained Delaunay triangulation, in the sense that the main properties, such as the empty circumcircle property, are still satisfied.
In our study we do not consider the presence of facilitators, but only barriers. In this cases, the segments defining holes borders or irregular boundaries of the domain must be specified and they will be in the set of the edges defining the Delaunay triangulation. The triangulation of the domain has been developed in `R` [R Development Core Team, 2008] by using the function `triangulate`, implemented in the `RTriangle` package [Shewchuk et al., 2016].

5.4 Graph-based metric

As we mentioned before, the aim of the Delaunay tessellation of the sampled locations is to define a more suitable metric than the Euclidean one, that is able to describe the spatial relation between the data. The RDDs method described in Chapter 4 is based on random domain decompositions, in particular Voronoi tessellations, repeated several times. The Voronoi tessellation is aimed to define homogeneous sub-regions of the domain wherein it is possible to assume stationarity. A Voronoi tessellation is defined by a set of nuclei and a metric $d(\cdot, \cdot)$. So far, we used the Euclidean metric, which generates tessellations of the domain that do not take into account the presence of obstacles.
Let $P = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_n\}$ be the set of sampled locations in the considered domain D and $\mathcal{DG}(P)$ the Delaunay graph obtained with a (constrained) Delaunay triangulation of P. This is an undirected graph $\mathcal{DG}(P) = G(P, E)$ and it is defined by the set of vertices P and the set of edges E (which includes also the segments defining the obstacles). The distance between two vertices $\mathbf{s}_i, \mathbf{s}_j \in P$ can be computed as the length of the shortest path from $\mathbf{s}_i$ to $\mathbf{s}_j$ carried on the graph $\mathcal{DG}(P)$.

A positive weight equals to the length of the segment $\overline{\mathbf{s}_i\mathbf{s}_j}$ is assigned to each edge $e := \{\mathbf{s}_i, \mathbf{s}_j\} \in E$

$$\omega_e = d_E(\mathbf{s}_i, \mathbf{s}_j)$$

where $d_E(\cdot, \cdot)$ indicates the Euclidean distance. Given an undirected graph with non-negative edges weights and with cycles, a possible algorithm for computing the shortest path between two vertices is the Dijkstra's Algorithm [Dijkstra, 1959]. Since we want to compute the distance between each pair of sampled locations, the Dijkstra's algorithm is applied $n(n-1)/2$ times and at each iteration the distance between two sample points is computed. The algorithm is implemented in `R` by using the `distances` functions of the `igraph` package [Csardi and Nepusz, 2006].
Let $d_{SP}(\mathbf{s}_i, \mathbf{s}_j)$ be the graph-distance between the two points, that is the length of the shortest path (SP) from $\mathbf{s}_i$ to $\mathbf{s}_j$ on the Delaunay graph. This distance is defined for each pair of observed locations, therefore a symmetric $(n \times n)$-dimensional distance matrix can be defined.
Since the final aim of this work is to define a Kriging method to perform prediction of unsampled locations $\mathbf{s}_0 \in D$, we should be able to define a distance also between unsampled and sampled locations. One possibility can be to perform the Delaunay triangulation of the domain based on the set of both the observed points and the locations where we want to perform prediction. In this case the distance between an unsampled and a sampled location could be computed as shortest path on the defined Delaunay graph. However, we remark that this approach can be computationally expensive if the final aim of the analysis is to use the observed data to make prediction on the whole grid of points within the domain of interest. In this case the Delaunay triangulation will be composed by very small and unnoticeable triangles. For this reason we opted for an approximation of the distances between unsampled and sampled locations, in the following way.
Let $\mathbf{s}_0$ be a location in D, before performing the distance between this point and a sampled location $\mathbf{s}_i$, we need to find the triangle of the Delaunay triangulation where it is located. There may be three cases

(a) the point is a vertex of the Delaunay triangulation;

(b) the point is in the interior of one triangle only;

(c) the point is on a common edge of two triangles.

Let $d_G(\mathbf{s}_0, \mathbf{s}_i)$ be the approximated distance between $\mathbf{s}_0$ and $\mathbf{s}_i$. The case (a) means that the point is a sampled location, since the set of vertices of the Delaunay triangulation coincides with the set of sampled location. In this case $d_G(\mathbf{s}_0, \mathbf{s}_i) = d_{SP}(\mathbf{s}_0, \mathbf{s}_j)$. The case (b) describes the situation when the point $\mathbf{s}_0$ is located within a triangle, which can be identified by its three vertices, denoted as $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3 \in P$.

The distance $d_G(\mathbf{s}_0, \mathbf{s}_i)$ is defined as:

$$\begin{aligned} d_G(\mathbf{s}_0, \mathbf{s}_i) = \min\{ & d_E(\mathbf{s}_0, \mathbf{p}_1) + d_{SP}(\mathbf{p}_1, \mathbf{s}_i); \\ & d_E(\mathbf{s}_0, \mathbf{p}_2) + d_{SP}(\mathbf{p}_2, \mathbf{s}_i); \\ & d_E(\mathbf{s}_0, \mathbf{p}_3) + d_{SP}(\mathbf{p}_3, \mathbf{s}_i)\} \end{aligned} \tag{5.1}$$

where $d_E(\cdot, \cdot)$ is the Euclidean distance and $d_{SP}(\cdot, \cdot)$ is the length of the SP between the two sampled points. In the last case (c), the point is located on an common edge shared by two triangles. Let $\mathbf{p}_1$ and $\mathbf{p}_2$ be the end points of this common edge. These points are also two of the three vertices of the two triangles. Let $\mathbf{p}_3$ be the third vertex identifying one ot the two triangles and $\mathbf{p}_4$ the third vertex identifying the other one. In this case the distance is defined as follows:

$$\begin{aligned} d_G(\mathbf{s}_0, \mathbf{s}_i) = \min\{ & d_E(\mathbf{s}_0, \mathbf{p}_1) + d_{SP}(\mathbf{p}_1, \mathbf{s}_i); \\ & d_E(\mathbf{s}_0, \mathbf{p}_2) + d_{SP}(\mathbf{p}_2, \mathbf{s}_i); \\ & d_E(\mathbf{s}_0, \mathbf{p}_3) + d_{SP}(\mathbf{p}_3, \mathbf{s}_i); \\ & d_E(\mathbf{s}_0, \mathbf{p}_4) + d_{SP}(\mathbf{p}_4, \mathbf{s}_i)\} \end{aligned} \tag{5.2}$$

Once this graph-based metric has been defined, the Voronoi tessellation of the domain could be computed by considering $d(\cdot, \cdot) = d_G(\cdot, \cdot)$.
Within each Voronoi cell, the estimation of the local stationary variogram is still computed by using a kernel-weighted method of moments, according to the following formula, as described in Chapter 4

$$\hat{\gamma}_\epsilon(\mathbf{h}; k) = \frac{\sum_{N(\mathbf{h})} K_\epsilon(\mathbf{c}_k, \mathbf{s}_i) K_\epsilon(\mathbf{c}_k, \mathbf{s}_j) \| Z_{\mathbf{s}_i} - Z_{\mathbf{s}_j} \|^2}{2 \sum_{N(\mathbf{h})} K_\epsilon(\mathbf{c}_k, \mathbf{s}_i) K_\epsilon(\mathbf{c}_k, \mathbf{s}_j)}$$

The norm $\| \cdot \|$ appearing in the formula is, in general, the norm on the feature space where the object data are represented. The metric used to define the $N(\mathbf{h})$ sets is the Euclidean one, since we are assuming that locally (i.e., within each Voronoi neighborhood) the *closeness* between points is well described by the Euclidean distance.
Since the kernel weights must be coherent with the Voronoi tessellation, the distance $d(\cdot, \cdot)$ appearing in the kernel definition (4.2) will be the graph-based distance $d_G(\cdot, \cdot)$ previously described.
Hence, in presence of complex domain with holes, irregular shapes or barriers, the Kriging-RDDs approach described in the previous chapter will be applied in the same way, with the only difference that the Voronoi tessellation, as well as the kernel values, are computed according to a graph-based metric instead of the standard Euclidean metric.
This modified method has been tested on two simulated datasets distributed over complex domains. In the first case the region of interest is a simple square domain as before, but with a hole in the central part of the domain. In the second case, the data are simulated on a C-shaped domain.

5.5 Simulated examples

5.5.1 Domain with hole

Figure 5.2 describes the domain related to the first example. It is the same domain that we studied in the Chapter 4, with two different Gaussian processes distributed over the two sub-regions separated by the ordinates axis, as presented by Kim et al. [2005]. We deleted the points within an ellipse located in the center of the domain. This ellipse represents our hole. If we look at the hole as an obstacle which cannot be crossed, the two black points of Figure 5.2 will result to be very far from each other: to go from one point to the other one needs to travel around the hole. Instead, if we simply consider the Euclidean distance, they will be very close to each other.

For this simulation example, we defined the exterior boundary of the domain, as

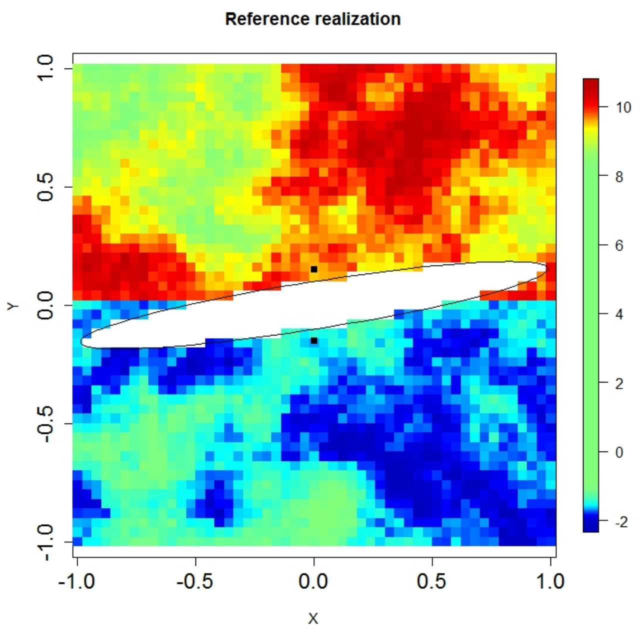

Figure 5.2: *Domain with hole: reference realization.*

well as the border of the hole, starting from the sampled locations. Precisely, the exterior border has been defined as the convex hull of the observed data, instead the hole's border has been defined with such points whose coordinates (x, y) fulfill the following condition

$$1 \leq \frac{(x\cos\alpha - y\sin\alpha - x_0)^2}{a^2} + \frac{(x\sin\alpha + y\cos\alpha - y_0)^2}{b^2} < 1 + \delta \qquad (5.3)$$

where $(x_0, y_0) = (0, 0)$ indicates the center of the ellipse, $\alpha = 9°$ is the angle of rotation, $a = 1$ and $b = 0.1$ are the semimajor and semiminor axis, respectively.

The value of $\delta \in (0, 0.5)$ was set according to the number of available observations (i.e., $\delta \approx 0$ if many observations are available, while δ increases if there are few observations). We can do it since we know the ellipse equation describing the hole. In real applications, we must be able to define the boundary of the domain, as well as the boundaries of the constraints within the domain. This can be computed by using the observed locations and the *a priori* information about the spatial properties of the domain or by adding some points defining the boundaries. These points will not be used for the Kriging analysis (i.e., they are NA data).

In the left panel of Figure 5.3 we can see an example of a dataset with 500 obser-

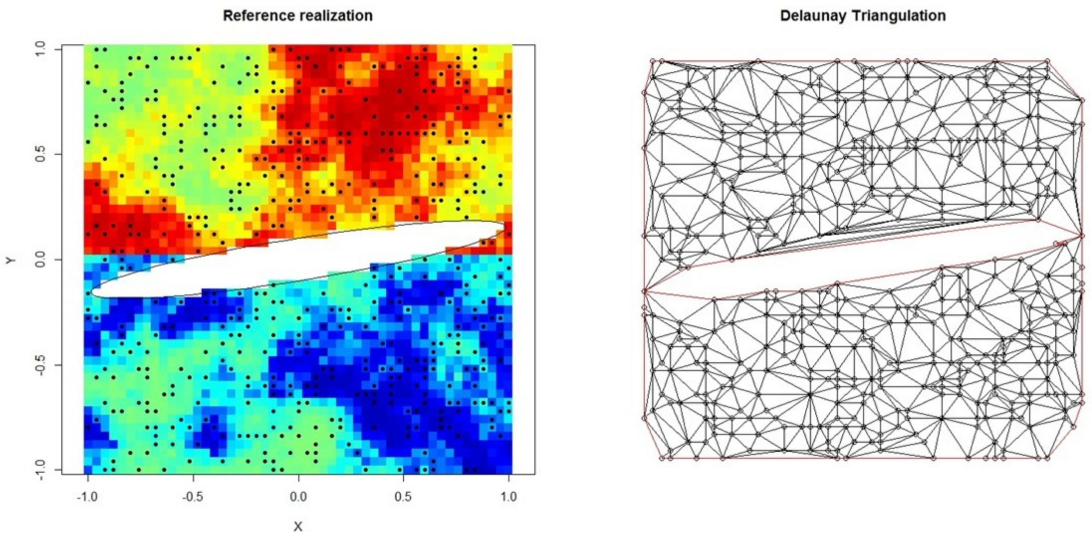

Figure 5.3: *Domain with hole: observations and Delaunay triangulation.*

vations distributed over the domain (the observations are indicated with the black points). In the right panel of the same figure there is the plot of the corresponding Delaunay triangulation of the domain. The red segments are the segments that we used to describe the boundaries of the domain. They must be used in the triangulation, even if they do not define properly *Delaunay triangles* according the previous definitions. Indeed, due to the presence of the hole, the triangles close to it seem to show skinny shapes, which is not a characteristic of triangles in a Delaunay mesh. A possible way for avoiding the presence of elongated triangles may be the addition of points and, consequently, of segments which give a more precise description of the ellipse's boundary. Another observation regards the exterior boundary: it coincides with the convex hull of the set of observed locations. For this reason some of the grid points of the square domain, in particular those close to the corners, are kept out from the Kriging analysis. These points are not located within any Delaunay triangle, therefore at this locations the Kriging predictor will not be computed. However, this is only an exemplifying case, it can be extended if one needs to make predictions for the points on the corners of the

domain.
According to the Kriging-RDDs method, B weak analyses are performed. Each of them is based on a realization of the random Voronoi tessellation of the domain. To understand the differences between the Kriging-RDDs approach based on the Euclidean distance or on the graph-based metric, we repeated the whole analysis $M = 50$ times. Each analysis has been computed on a dataset with $nsub = 250$ observations, firstly by using an Euclidean approach and then the same dataset has been analysed by using the graph-based metric approach.

Figure 5.4 show the boxplots of the MSPEs computed throughout the M repetitions. The case $K = 1$ shows a single boxplot, since the same analysis based on the whole domain without partition is computed and the estimate of the (global) variogram does not require the use of kernel weights. We can observe that, for small values of K, the graph-based metric approach shows better results. When K increases the differences among the two approaches are less obvious. Indeed, for high values of K the tessellations computed with the two metrics are similar, but the boundaries of the tiles obtained with the graph-based metric are more jagged with respect to the straight lines of the Euclidean tiles. The irregularity

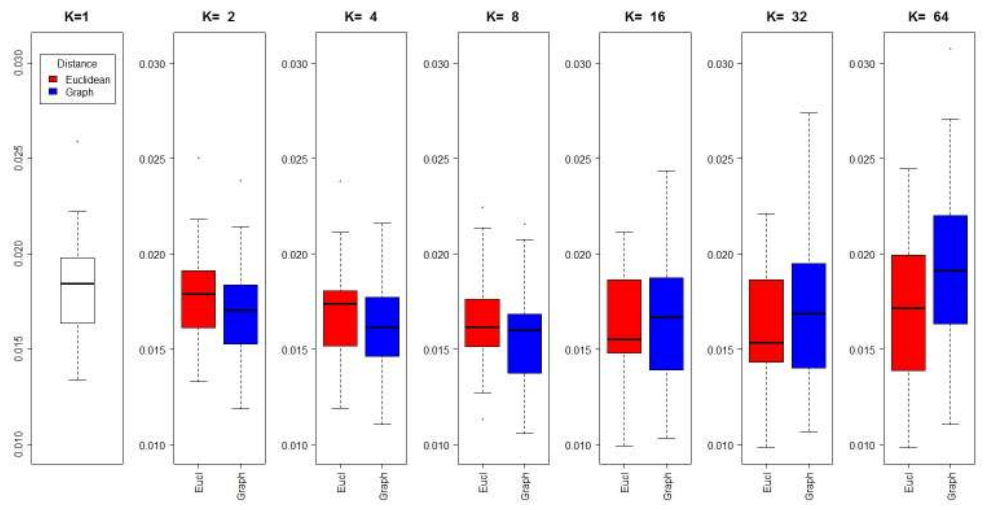

Figure 5.4: *Domain with hole. MSPE boxplots: different distances.*

Distance	K						
	1	2	4	8	16	32	64
Euclidean	0.0183	0.0179	0.0172	0.0165	**0.0161**	0.0163	0.0172
Graph	0.0183	0.0171	0.0164	**0.0159**	0.0164	0.0172	0.0192

Table 5.1: *Domain with hole. Kriging-RDDs method with different metrics: Average of the MSPEs over the $M = 50$ repetitions.*

of the Voronoi tiles boundaries is due to the fact that the Delaunay triangulation is computed by considering only the sampled locations as vertices of the graph. A denser triangulation, with the addition of other vertices, may better describe the adjacency relations among the locations, which may be reflected into a more regularity in the boundaries of the Voronoi tiles.
A more detailed analysis on the differences between the two approaches is described by using the following simulation example.

5.5.2 C-shaped domain

This example of irregularly shaped region comes from the literature: it is the C-shaped domain studied by Ramsay [2002], Wood et al. [2008] and Sangalli et al. [2013]. In this simulation study a test function varies smoothly from a value of about -4 at the boundary of the lower branch of the C-shape to a value of about 4 at the boundary of the upper branch, as we can see from Figure 5.5. The particular shape of the region does not allow us to assume stationarity on the whole domain, due to the presence of the thin space which separates the two branches of the C-shape, but the local stationarity assumption is verified for almost all the points within the domain. Therefore, given a set of observations distributed over this domain, we performed Kriging analysis via RDDs.

Contrarily to the previous example, here the Delaunay triangulation of the do-

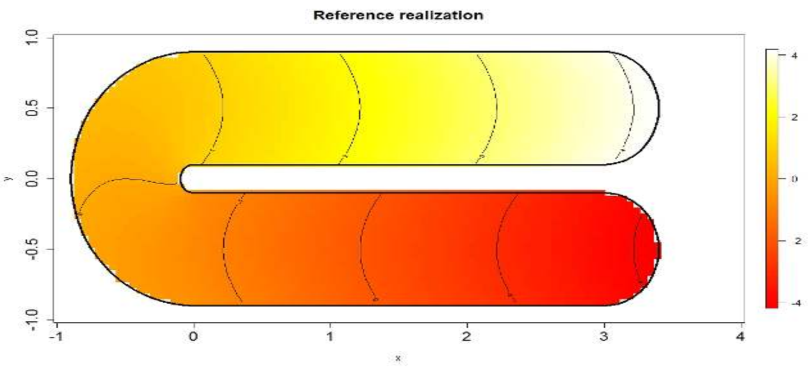

Figure 5.5: *C-shaped domain, smooth test function.*

main cannot be computed by using the convex hull of the observed points, but the boundary of the domain must be carefully defined. For this reason, we defined the boundary through a set of locations which will not be used for the kriging prediction. In fact, we use only their positions, but not the value of the test function at these points (i.e., their value is set to NA for the Kriging analysis). In Figure 5.6 we can see an example of dataset with $nsub = 250$ observations. The black points in the top panel of the figure are the points used for defining the domain boundary. The bottom panel of the same figure shows the corresponding Delaunay triangulation based on this set of sampled points. The introduction of such points for the definition of the domain boundary ensures a more regular triangulation with respect to the one which can be obtained when the boundaries are estimated by using the observed locations. We remark that a good triangulation is essential for the performance of the algorithm.

It is interesting to see the differences between random domain partitions based on the Euclidean metric and those defined on a graph-based metric. Figure 5.7 show some examples which allow us to make this comparison: the panels on the right side refer to Euclidean Voronoi tessellations, while those on the left side refer to Voronoi tessellations defined by a graph-based metric. For small values of K, tessellations based on the two metrics are dramatically different. If points located on the terminal part of different branches of the C-shape are often assigned the same sub-region with an Euclidean approach, this does not happen if the graph-based metric is used. Therefore, in the neighborhoods defined with a graph-based metric the local stationarity is a viable assumption, while it often does not hold

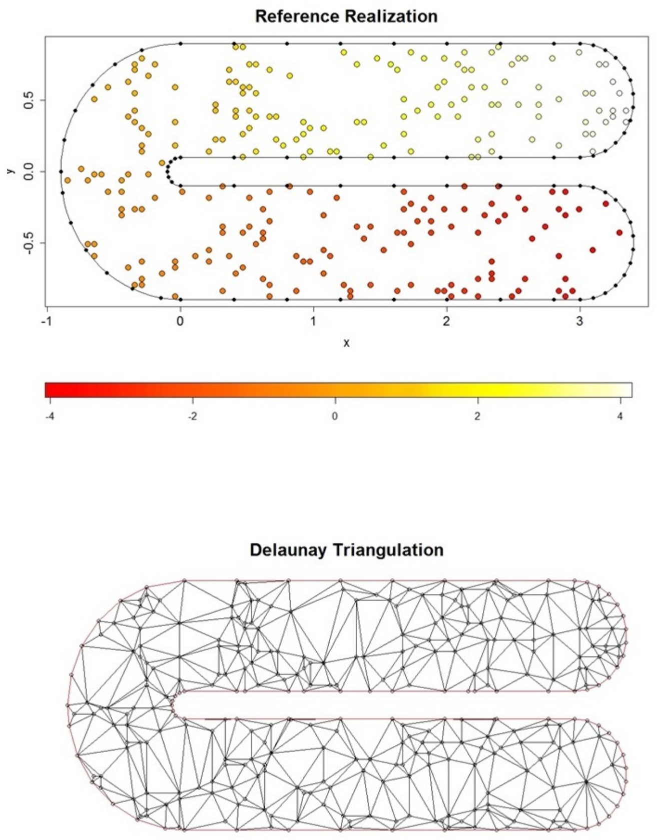

Figure 5.6: *C-shaped domain: example of dataset with $nsub = 250$ observations and the corresponding Delaunay triangulation.*

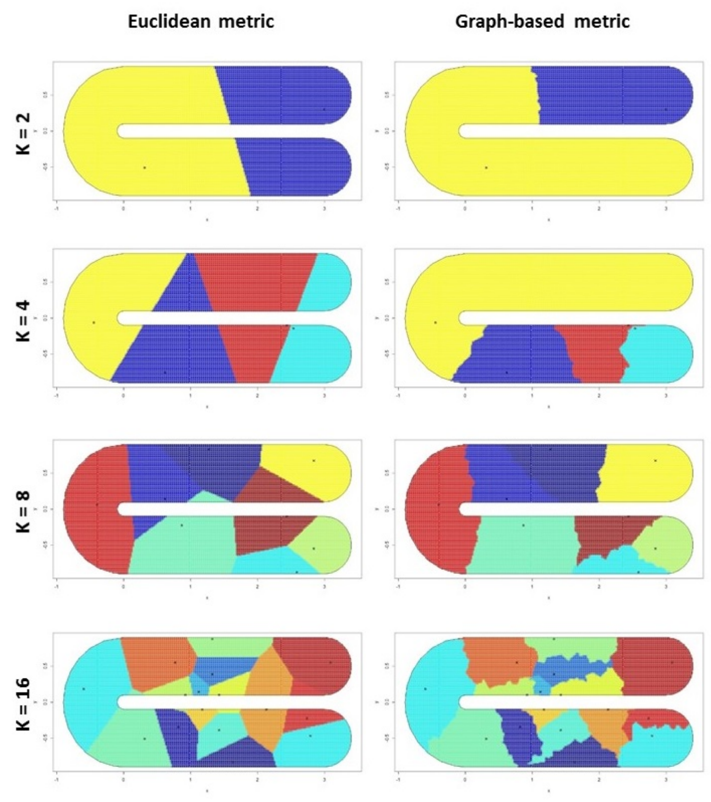

Figure 5.7: *C-shaped domain: examples of Voronoi tessellation of the domain for different values of K and for different metric $d(\cdot,\cdot)$.*

true when the Euclidean metric is used. For higher values of K the differences in the assignments are noticeable in particular for the points close to the inner boundary of the C-shape. The tiles obtained with the two approaches are similar, even if those defined with a graph-based metric show more jagged boundaries.
It is also important to note the differences among the kernel values based on the two metrics. If the simple Euclidean metric is used to define the kernel function according to equation (4.2), the resulting kernel will be isotropic. In that case, having fixed a Voronoi nucleous, all the points at the same (Euclidean) distance from the nucleous will have the same kernel value even if they are on different branches of the C-shape. In fact, the isotropic kernel does not takes into account the properties of the domain, for example the presence of barriers. When a graph-based metric is used for the definition of the domain partition, the kernel function must be coherently defined by using the same metric. In this case, the kernel is not a symmetric kernel, in particular the symmetry property is no longer fulfilled near the barriers. Figure 5.8 shows the differences between an Euclidean kernel (panels on the right side) and a graph-based kernel (panels on the left side). For each grid point $\mathbf{s}_0$, the images refer to the value of the kernel $K_\epsilon(\mathbf{s}_0, \mathbf{c})$, where

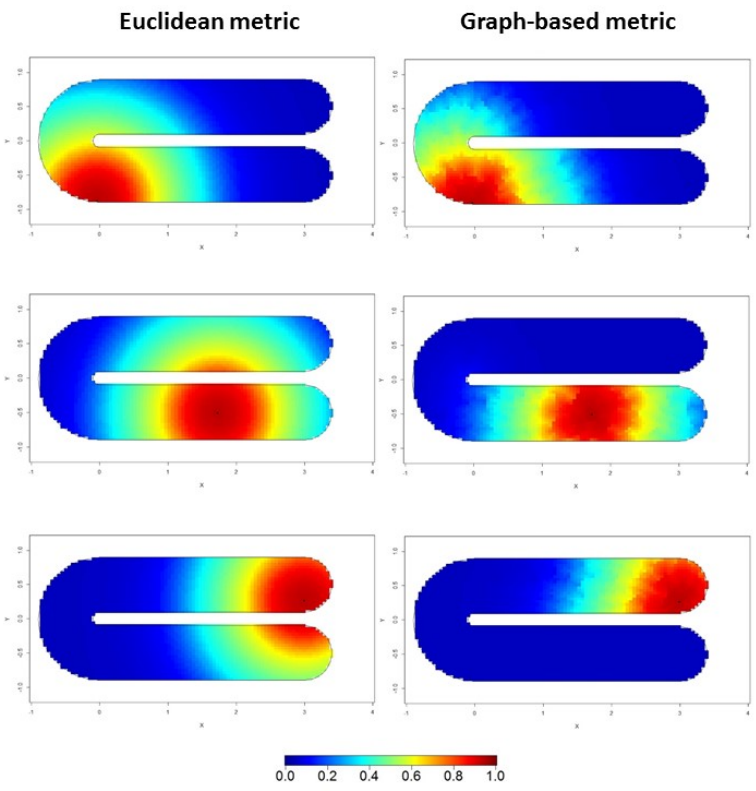

Figure 5.8: *C-shaped domain: examples of kernel weights for different positions of the Voronoi center and for different metric $d(\cdot,\cdot)$.*

c is the center indicated by the black point and the kernel bandwidth was set to $\epsilon = 1$. Although the differences among the two approaches in the first cases (top panels) are not particularly evident, from the central and bottom panels we can appreciate that points on one branch do not contribute on the estimate of variograms in the other branch when the graph-based distance is used. This does not happen if the Euclidean distance is used, since the narrow empty space which divides the two branches is not considered.

As for the previous example, also in this case we performed the whole Kriging-RDDs analysis $M = 50$ times to understand the performance of the algorithm for different values of K. In particular, every time, we used a 250-dimensional dataset, $B = 100$ bootstrap repetitions, a kernel width $\epsilon = 1$ and a final predictor based on an "equal" aggregation. Each dataset has been analysed with the two approaches: one based on the Euclidean distance, the other one based on a graph metric. We used $K = \{1, 2, 4, 8, 16\}$, where $K = 1$ means that an Ordinary Kriging technique has been performed for the predictions. The MSPE has been computed every time and for each fixed value of K and the boxplots of the MSPEs obtained during the 50 repetitions are showed in Figure 5.9.

We can observe that, even with a Kriging-RDDs based on the Euclidean distances, when K increases, we get better results with respect to the case of $K = 1$. If we compare the Euclidean approach with the graph-based approach, we can observe that the latter gives better results in term of MSPE. In particular the MSPEs values are very close to zero for $K = 4, 8, 16$. Figure 5.11 shows the Kriging predictions values obtained via RDDs based on both the Euclidean (left panels) and graph (right panels) metric, with $K = 4, 8$.
It is interesting to understand where the MSPEs assume higher values. As we can observe from Figure 5.10, when the procedure is based on the Euclidean metric, the highest values of the prediction error $\epsilon^*_{\mathbf{s}_0}(\mathbf{Z}) = Z_{\mathbf{s}_0} - Z^*_{\mathbf{s}_0}(\mathbf{Z})$ are associated with those points located close to the inner boundary of the C-shape. In particular, points of the lower branch are estimated with higher values with respect to the real values, due to the use of information coming from locations of the other upper branch, therefore the prediction errors for these points tend to be negative. In-

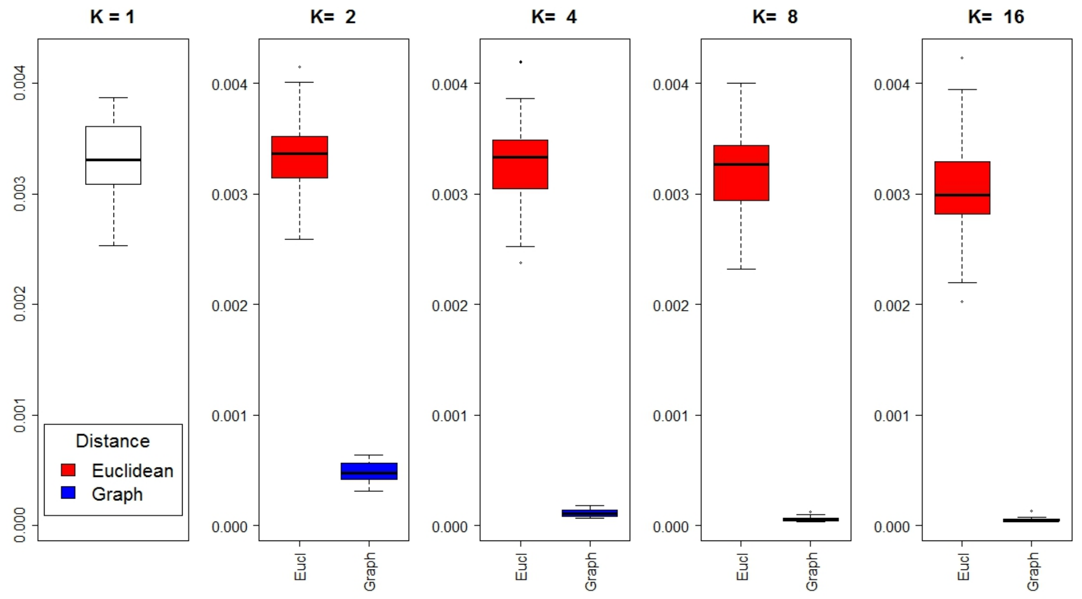

Figure 5.9: *C-shaped domain. MSPE boxplots: different distances.*

Distance	K				
	1	2	4	8	16
Euclidean	$3.287 \cdot 10^{-3}$	$3.365 \cdot 10^{-3}$	$3.307 \cdot 10^{-3}$	$3.207 \cdot 10^{-3}$	$\mathbf{3.029 \cdot 10^{-3}}$
Graph	$3.287 \cdot 10^{-3}$	$0.4807 \cdot 10^{-3}$	$0.1168 \cdot 10^{-3}$	$0.0603 \cdot 10^{-3}$	$\mathbf{0.0551 \cdot 10^{-3}}$

Table 5.2: *C-shaped domain. Kriging-RDDs method with different distances: Average of the MSPEs over the $M = 50$ repetitions.*

stead, for points located on the top branch, the estimated values $Z^*_{\mathbf{s}_0}(\mathbf{Z})$ are lower than the real values $Z_{\mathbf{s}_0}$. This situation does not happen when the graph-based distance is used, since the information of one branch are not used for estimating variables associated with the other branch.
In real applications we cannot compute the MSPE, therefore we need to define a method which is able to provide an optimal value for the K parameter. One possibility may be to set K by cross-validation, although this approach is very computationally expensive. For this reason we explored the concept of variance associated with the final Kriging predictor aiming to grasp information regarding the setting of parameter K.

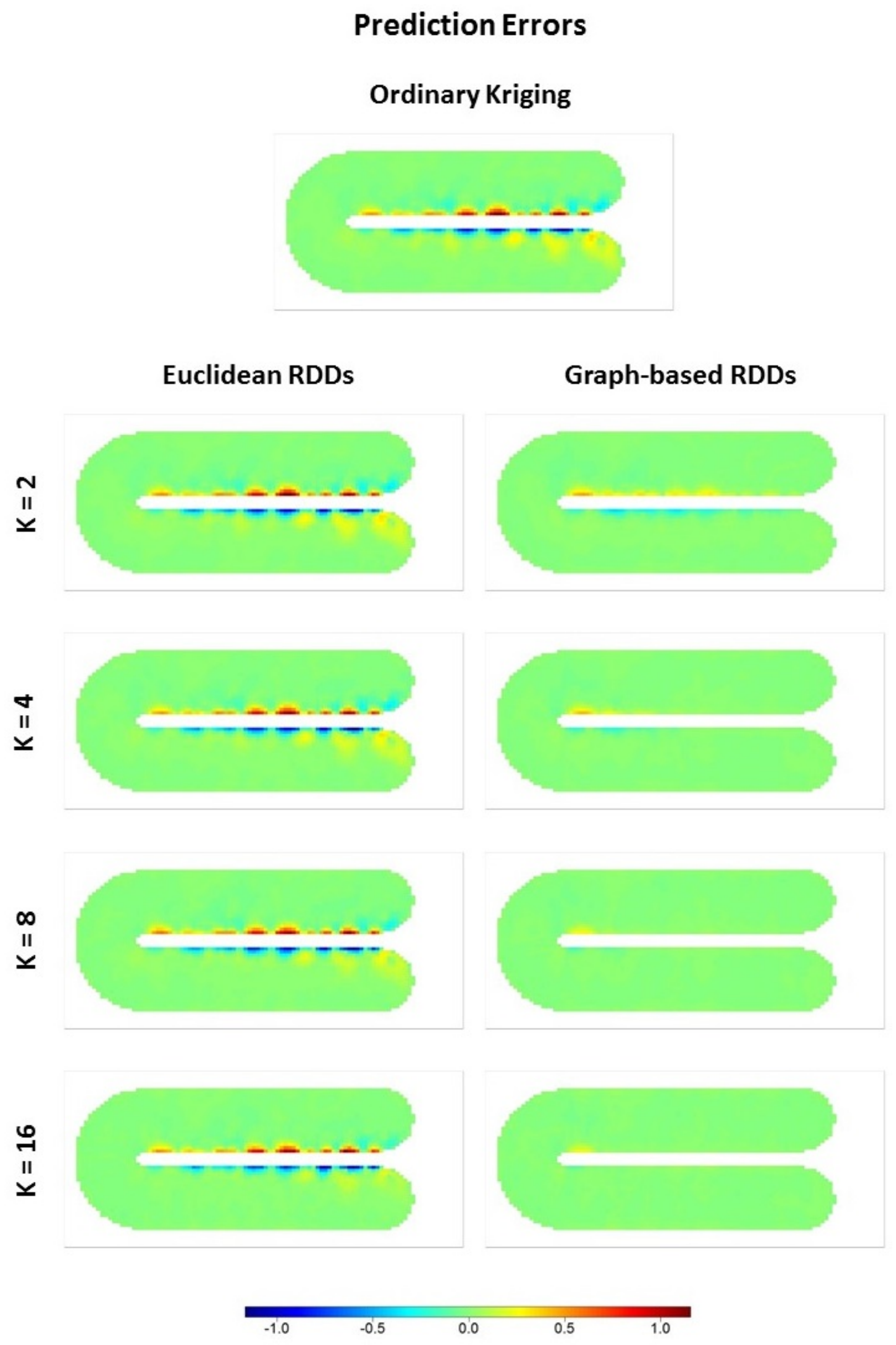

Figure 5.10: *MSPEs computed via Kriging-RDDs applied to data over the C-shaped domain: comparison between Euclidean (left panels) and graph-based (right panels) metrics.*

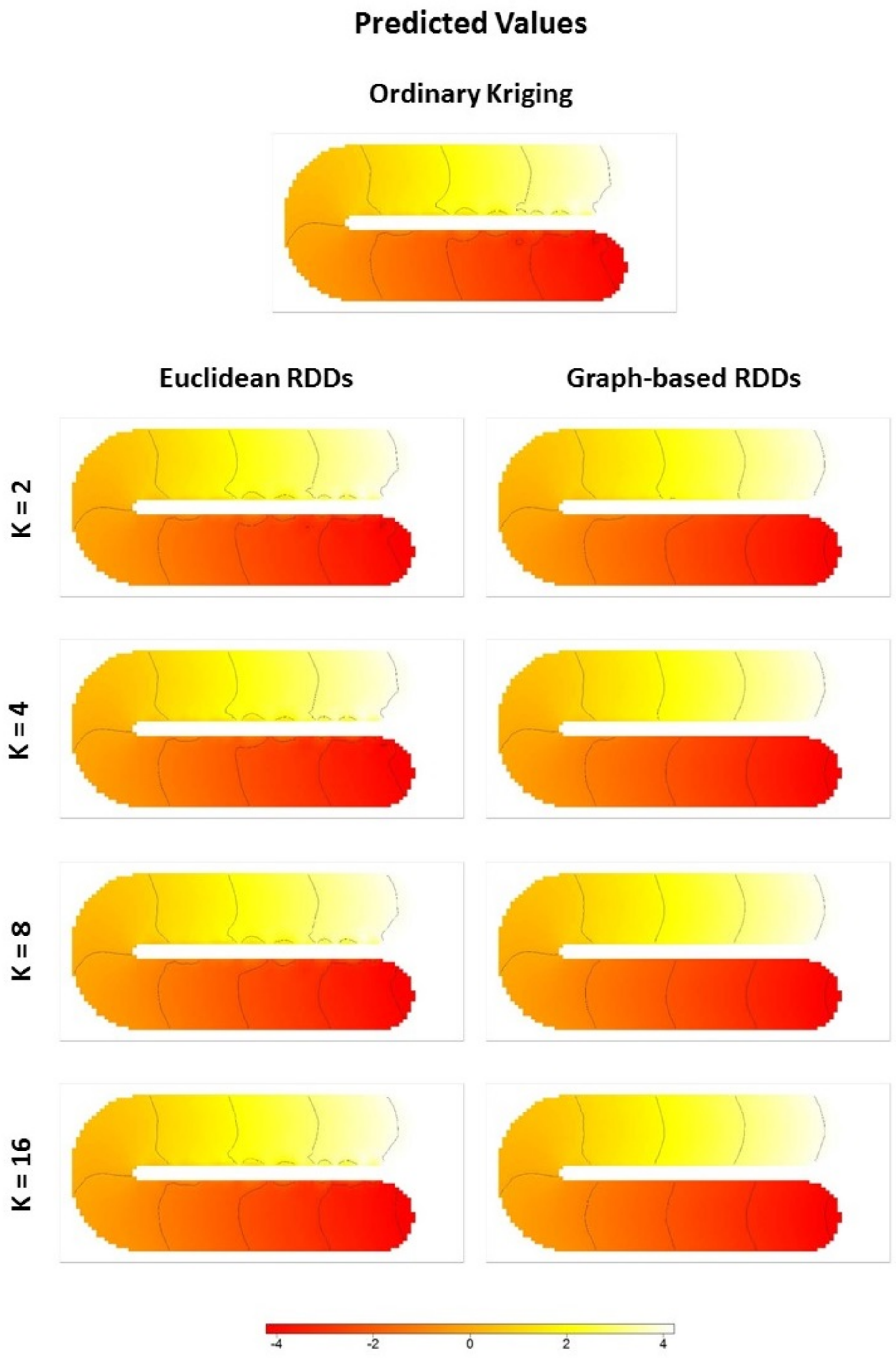

Figure 5.11: *Kriging predictions computed via RDDs applied to data over the C-shaped domain: comparison between Euclidean (left panels) and graph-based (right panels) metrics.*

5.6 Variance

A method which allows the choice for the value of the parameter K is needed for the analysis. In Secchi et al. [2013] a Bagging Voronoi method was used for clustering spatial functional data, therefore the final predictor at an unsampled location indicated the label of the class to which the location was assigned. In that case a criterion to set K was based on the minimization of average *entropy* associated with the cluster assignment distribution along the bootstrap replicates. This entropy was defined as measure of the final predictor uncertainty. For an optimal choice of K, we expect a data classification leading to a map of the spatial entropy mostly equal to zero, with values significantly different from zero only in sites close to the boundaries between regions associated with different labels. Therefore an optimal choice of K implies a small value, close to zero, for the *entropy.* This measure of the accuracy of the predictor can be extended to continuous distributions, hence when the aim of the analysis is to make predictions, but the entropy cannot be easily defined in the O2S2 framework.
In Secchi et al. [2015] a criterion based on the *total average variance* (TAV) was proposed to set the parameter K. In this case the aim of the work was to define a non-parametric method for the analysis of spatially dependent functional data based on a dimensional reduction of these signals, aimed to identify sub-regions of the metropolitan area of Milan sharing a similar pattern. The TAV was defined as the average over the sites $\mathbf{s}_0 \in D$ and over the time intervals of the bootstrap variance. A small value of TAV implied stable scores across bootstrap replicates.
In our case, the choice of a value for the parameter K based on the analysis of the total variance is more complicated. Indeed, this variance includes the variance due to the bootstrap replicates and the Kriging variance associated with the prediction error. Even if we do not define a complete formula for the final variance, both the bootstrap variance and an approximated Kriging variance may provide useful information for the choice of a suitable value, or at least a set of values, for the parameter K.
For each bootstrap iteration, the Kriging variance associated with the prediction error $\epsilon^{*b}_{\mathbf{s}_0}(\mathbf{Z}) = \epsilon^*_{\mathbf{s}_0}(\mathbf{Z}; P = P_b) = Z_{\mathbf{s}_0} - Z^*_{\mathbf{s}_0}(\mathbf{Z}; P = P_b)$ is computed according to the equation (1.14), since Ordinary Kriging has been performed within each Voronoi tile, and it is indicated as $\sigma^2_{OK}(\mathbf{s}_0)^b$. After B replicates and an "equal" aggregation, the final prediction error is defined as

$$\epsilon^*_{\mathbf{s}_0}(\mathbf{Z}; P) = \frac{1}{B}\sum_{b=1}^{B} \epsilon^*_{\mathbf{s}_0}(\mathbf{Z}; P = P_b)$$

whose variance is defined as

$$\mathrm{Var}_{\mathrm{Data}}[\epsilon^*_{\mathbf{s}_0}(\mathbf{Z}; P)] = \mathbb{E}_{\mathrm{Data}}[(\epsilon^*_{\mathbf{s}_0}(\mathbf{Z}; P) - \mathbb{E}_{\mathrm{Data}}[\epsilon^*_{\mathbf{s}_0}(\mathbf{Z}; P)])^2]. \tag{5.4}$$

Indeed, the final predictor $Z^*_{\mathbf{s}_0}(\mathbf{Z}; P)$, aggregated according to an "equal" average of the bootstrap predictors, is unbiased. Note that $Z^*_{\mathbf{s}_0}(\mathbf{Z}; P) =$

$\frac{1}{B}\sum_{b=1}^{B} Z^*_{\mathbf{s}_0}(\mathbf{Z}; P = P_b)$ is the discretized version of

$$\mathbb{E}_P[Z^*_{\mathbf{s}_0}(\mathbf{Z}; P)] = \int_{\mathcal{P}} Z^*_{\mathbf{s}_0}(\mathbf{Z}; p) dP(p)$$

where $\mathcal{P}$ is the set of all the possible partitions of the domain. Due to the independence between the data and each partition, using the Fubini-Tonelli Theorem, we can write

$$\mathbb{E}_{\text{Data}}\Big[\int_{\mathcal{P}} Z^*_{\mathbf{s}_0}(\mathbf{Z}; p) dP(p)\Big] = \int_{\mathcal{P}} \mathbb{E}_{\text{Data}}\Big[Z^*_{\mathbf{s}_0}(\mathbf{Z}; p)\Big] dP(p) = \mathbb{E}[Z_{\mathbf{s}_0}] \tag{5.5}$$

where the second equality is a consequence of the fact that, when the partition is fixed $P = p$, the Kriging predictor $Z^*_{\mathbf{s}_0}(\mathbf{Z}; p)$ is the Best Linear Unbiased Predictor (BLUP) of $Z_{\mathbf{s}_0}$ (as mentioned in Chapter 1, Section 1.1.4). In particular, the unbiasedness equality (5.5) holds true also for the discretized predictor $Z^*_{\mathbf{s}_0}(\mathbf{Z}; P)$. Therefore, equation (5.4) becomes

$$\text{Var}_{\text{Data}}[\epsilon^*_{\mathbf{s}_0}(\mathbf{Z}; P)] = \mathbb{E}_{\text{Data}}[\epsilon^*_{\mathbf{s}_0}(\mathbf{Z}; P)^2]. \tag{5.6}$$

A formula for variance (5.6) requires the definition of an aggregated global (non-stationary) covariance function, which will be the scope of future work. In this dissertation, we focus our attention on a global indication of the uncertainty associated with the Kriging-RDDs, computed as the average of the Kriging variances (AKV) over the B repetitions, i.e.,

$$\text{AKV}(\mathbf{s}_0) = \frac{1}{B}\sum_{b=1}^{B} \sigma^2_{OK}(\mathbf{s}_0)^b. \tag{5.7}$$

The top panels of Figure 5.12 shows an example of the AKV($\cdot$) values for each grid point obtained after Kriging predictions. The left panel refers to an Ordinary Kriging approach: the values of the Kriging variance are within the interval $[0, 1.47]$ and the highest values are associated with the points of the regions of the domain with low density of observations. The central panel shows the values of AKV($\cdot$) when the Kriging prediction is computed via Euclidean-RDDs based on partitions of the domain into $K = 8$ neighborhoods. In this case the AKV($\cdot$) $\in [0, 2.94]$ and the highest values refer to the points on the ending parts of the two branches of the C-shape. The right panel of Figure 5.12 shows the AKV($\cdot$) values when Kriging predictions are computed via graph-based RDDs with $K = 8$. In this case, AKV($\cdot$) $\in [0, 0.32]$ and the highest values are associated with the points close to the curved part of the C-shape.
We can appreciate that the graph-based RDDs approach provides smaller values for the average Kriging variance with respect to the values obtained with Ordinary Kriging or Euclidean-RDDs. Indeed, it is important to highlight the fact that the ranges of values are totally different: the graph-based approach results in AKV

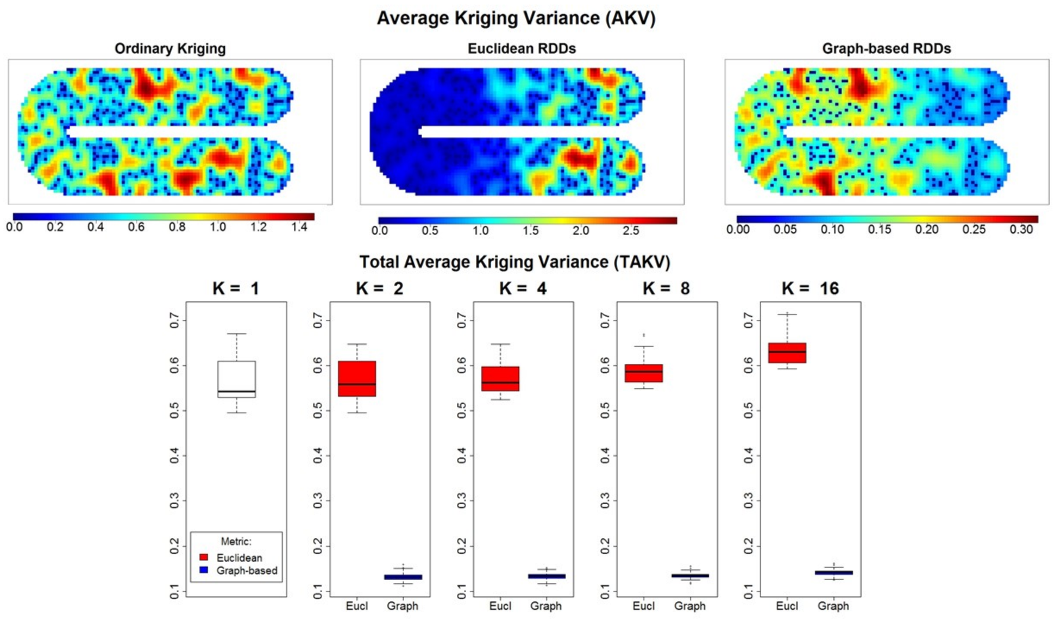

Figure 5.12: *Average Kriging Variance (AKV) plots. Left panel: Ordinary Kriging; central panel: Euclidean-RDDs with $K = 8$; right panel graph-based RDDs with $K = 8$. Bottom panel refers to the boxplots of the TAKV.*

Distance	K				
	1	2	4	8	16
Euclidean	**0.5636**	0.5705	0.5744	0.5902	0.6352
Graph	0.5636	**0.1324**	0.1337	0.1357	0.1425

Table 5.3: *C-shape domain. Average of the TAKV over the $M = 50$ repetitions.*

values of the order of 10^{-1} with respect to those associated with the Euclidean framework.
As before, for each value of K the analysis has been computed $M = 50$ times. At each p-th iteration we computed the total average of the AKV values over the domain:

$$\text{TAKV} = \frac{1}{n_g} \sum_{\mathbf{s}_0 \in \mathcal{G}} AKV(\mathbf{s}_0)$$

The bottom panel of Figure 5.12 shows the boxplots of the TAKV values. We can observe that the use of a graph-based metric allows to drastically reduce the total average Kriging variance with respect to both the Ordinary Kriging and the Euclidean RDDs approaches.
Although the value (5.7) provides an indication on the global prediction uncertainty, another source of variability in the aggregated prediction $Z^*_{\mathbf{s}_0}(\mathbf{Z}; P)$ is pro-

vided by the bootstrap variance $\sigma^2_{BT}(\mathbf{s}_0)$ defined as

$$\sigma^2_{BT}(\mathbf{s}_0) = \mathrm{Var}_P[Z^*_{\mathbf{s}_0}(\mathbf{Z}; P)].$$

It allows to give a measure of how the aggregated predictor deviates from the $\mathbb{E}_P[Z^*_{\mathbf{s}_0}(\mathbf{Z}; P)]$. We can simply compute an approximation of this bootstrap variance as

$$\sigma^2_{BT}(\mathbf{s}_0) \approx \frac{1}{B-1}\sum_{b=1}^{B}\left(Z^*_{\mathbf{s}_0}(\mathbf{Z}; P = P_b) - \frac{1}{B}\sum_{b=1}^{B} Z^*_{\mathbf{s}_0}(\mathbf{Z}; P = P_b)\right)^2.$$

The use of a graph-based metric for the definition of the RDDs implies a reduction of the bootstrap values with respect to the use of Euclidean-RDDs, as we can see from Figure 5.13. With an Euclidean approach the points that show the highest values of bootstrap variance are those close to the inner border

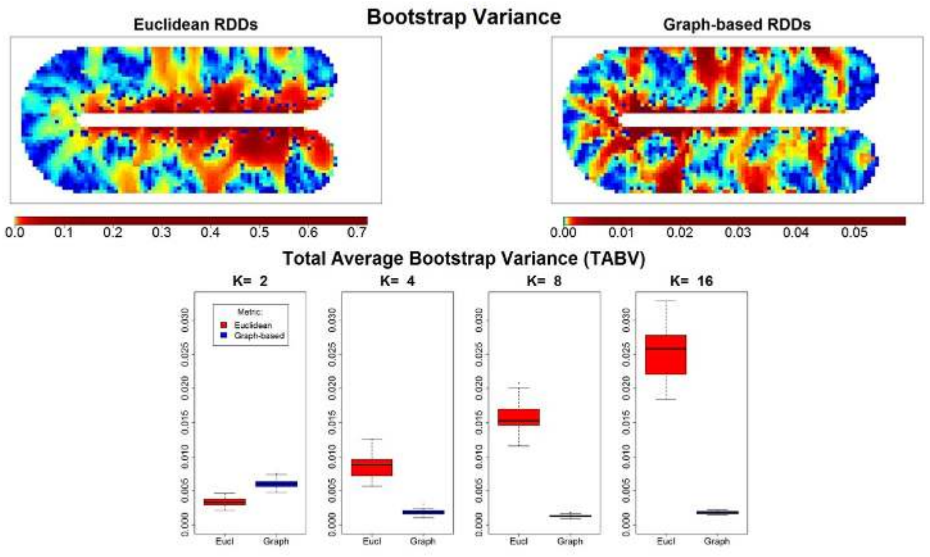

Figure 5.13: *Bootstrap Variance plots. Left panel: Euclidean-RDDs with $K = 8$; right panel graph-based RDDs with $K = 8$. Bottom panel refers to the boxplots of the TABV.*

Distance	**K**			
	2	4	8	16
Euclidean	**0.0033**	0.0087	0.0156	0.0250
Graph	0.0059	0.0018	**0.0013**	0.0018

Table 5.4: *C-shape domain. Average of the TABV over the $M = 50$ repetitions.*

which separates the two branches of the C-shape, while for the other grid points $\sigma^2_{BT}(\mathbf{s}_0) \approx 0$. Instead, with graph-based RDDs the bootstrap variance assumes values very close to zero for almost all the grid points. As before, even in this case the range of values are totally different. In the example reported in Figure 5.13, $\sigma^2_{BT}(\mathbf{s}_0) \in [0, 0.72]$ if the Euclidean metric is employed for the RDDs, while with a graph-based approach $\sigma^2_{BT}(\mathbf{s}_0) \in [0, 0.06]$.
The boxplots of Figure 5.13 refers to the total average bootstrap variance (TABV) over the domain computed during the $M = 50$ repetitions of the analysis

$$\mathrm{TABV} = \frac{1}{n_g} \sum_{\mathbf{s}_0 \in \mathcal{G}} \sigma^2_{BT}(\mathbf{s}_0).$$

When $K = 1$, a single iteration of Ordinary Kriging over the whole domain is performed, therefore the bootstrap variance is 0 for all the grid points.
If we used the information coming from the bootstrap and Kriging variance for the choice of the parameter K when the RDDs are based on the graph metric, our choice could fall on the values $K = 4$ or $K = 8$. Indeed, the boxplots of the TAKV suggest to use $K = 2, 4, 8$, while the boxplots of the TABV suggest to set $K = 4$ or 8. Therefore, when $K = 4$ or $K = 8$ both the approximation of the Kriging variance and the bootstrap variance show the smallest values. This result is coherent with the results coming from the MSPE values. Indeed, when $K = \{4, 8\}$ the MSPEs are very close to zero for almost all the points of the C-shaped domain.

5.A Appendix. Theorems for checking the isometrically embeddable property

Let D be an arbitrary collections of spatial locations $\mathbf{s} \in \mathbb{R}^n$, we can define a function $d(\cdot,\cdot)$ representing the distance function such that $d : D \times D \to [0,\infty)$. The d is said to satisfy the conditions of *metric* if

$$\begin{aligned} &d(\mathbf{s}_i,\mathbf{s}_j) \leq 0 \quad \text{and} \quad d(\mathbf{s}_i,\mathbf{s}_j) = 0 \text{ iff } \mathbf{s}_i = \mathbf{s}_j; \\ &d(\mathbf{s}_i,\mathbf{s}_j) = d(\mathbf{s}_j,\mathbf{s}_i); \\ &d(\mathbf{s}_i,\mathbf{s}_j) \leq d(\mathbf{s}_i,\mathbf{s}_k) + d(\mathbf{s}_k,\mathbf{s}_j) \quad \text{(Triangle inequality)} \end{aligned}$$

for all $\mathbf{s}_i, \mathbf{s}_j \in D$. The space D together with the metric d is called *metric space.* The following theorem, due originally to Schoenberg [1937] (see also Young and Householder [1938]) provides a necessary and sufficient condition for the embedding property of a finite metric space.

Theorem 5.2. (Schoenberg, 1937). The finite metric space (D,d) where $D = \{\mathbf{s}_0,\mathbf{s}_1,\dots,\mathbf{s}_n\}$ with $n > 2$, is embeddable in $\mathbb{R}^n$ if and only if

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j\{d(\mathbf{s}_0,\mathbf{s}_i)^2 + d(\mathbf{s}_0,\mathbf{s}_j)^2 - d(\mathbf{s}_i,\mathbf{s}_i)^2\} \geq 0 \tag{5.8}$$

for all choices of real numbers $\lambda_0, \lambda_1, \dots, \lambda_n$.

As pointed out in Wells and Williams [1975], the condition (5.8) is equivalent to

$$\sum_{i=0}^{n}\sum_{j=0}^{n}\lambda_i\lambda_j d(\mathbf{s}_i,\mathbf{s}_j)^2 \leq 0 \tag{5.9}$$

for all choices of real numbers $\lambda_0, \lambda_1, \dots, \lambda_n$ such that $\sum_{i=0}^{n}\lambda_i = 0$. This is the definition of the conditionally negative property. Therefore a distance function d is embeddable if and only if d^2 is conditionally negative definite. Moreover, thanks to this theorem we can conclude that the square root of the variogram is equivalent to an Euclidean metric. In Wells and Williams [1975], we can also find a connection between the positive definiteness and the conditionally negative property, through the following relation

$$e^{-ad(\cdot,\cdot)} \text{ is positive definite } \forall a > 0 \ \underline{\text{iff}}\ d(\cdot,\cdot) \text{ is conditionally negative definite.}$$

A simple approach for defining if a distance d is embeddable, has been defined by Richards [1985] for the class of norm distances.

Definition 5.2. A *vector norm* is a function $f : \mathbb{R}^n \rightarrow [0, \infty)$ that satisfies the following properties

$$\begin{aligned} &f(\mathbf{h}) \leq 0 \quad \text{ and } \quad f(\mathbf{h}) = 0 \text{ iff } \mathbf{h} = 0; \\ &f(\mathbf{h} + \mathbf{h}^*) \leq f(\mathbf{h}) + f(\mathbf{h}^*); \\ &f(\alpha \mathbf{h}) = |\alpha| f(\mathbf{h}) \end{aligned}$$

for all $\mathbf{h}, \mathbf{h}^* \in \mathbb{R}^n$.

A vector norm becomes a distance metric by defining $d(\mathbf{s}_i, \mathbf{s}_j) = f(\mathbf{h})$, with $\mathbf{h} = \mathbf{s}_i - \mathbf{s}_j$. The common α-norm distance metrics for $\alpha \geq 1$ are defined as

$$||\mathbf{h}||_\alpha = (|h_1|^\alpha + |h_2|^\alpha + \cdots + |h_n|^\alpha)^{\frac{1}{\alpha}}$$

with $\mathbf{h} = (h_1, h_2, \ldots, h_n)^T$. Richards provided a simple rule to define whether a norm distance is embeddable or not, through the following theorem.

Theorem 5.3. (Richards, 1985)

(a) On $\mathbb{R}^2$, $||\mathbf{h}||_\alpha^\beta$ is *conditionally negative* definite if

  (i) $0 < \beta \leq 1$, $1 \leq \alpha \leq \infty$, or

  (ii) $0 < \beta \leq \alpha \leq 2$.

(b) On $\mathbb{R}^n$ with $n \geq 3$, $||\mathbf{h}||_\alpha^\beta$ is *conditionally negative* definite if

  (i) $0 < \beta \leq \alpha \leq 2$, and

  (ii) if $\alpha > 2$ it is not conditionally negative definite if $\beta > 1$

where α is the norm index and β indicates the power.

Therefore, if we combine the results of this theorem with Schoenberg's Theorem 5.2 we can conclude that if $||\mathbf{h}||_\alpha^\beta$ is conditionally negative definite, then $||\mathbf{h}||_\alpha^{\beta/2}$ is embeddable, hence it can be used with existing covariance and variogram functions that are valid in all dimensions.

Chapter 6

Case Study: Chesapeake Bay

We apply the proposed RDDs methodology to perform Kriging prediction for spatial data distributed over the complex and highly irregular domain of the Chesapeake Bay. In particular, we studied a dataset regarding the values of *Dissolved Oxygen* (DO) in the water of the estuary.

6.1 Geographical description

The Chesapeake Bay is the largest estuary in the United States and the third largest in the world. This estuarine ecosystem is approximately 300 km long, from Havre de Grace, Maryland (on the North) to Virginia Beach, Virginia (on the South). Its width ranges between 5 km (the mean width of the mainstream) and 30 km, if one considers the lateral tributaries. The total shoreline, including tributaries, is 18804 km long, and circumnavigates a surface area of 11601 km^2. The average depth is around 6.4 m, reaching a maximum of 53 m in the deepest part of the Bay, located at south-east which is called “The Hole”. The Bay is a mix of fresh and salt water (coming from the Atlantic Ocean).
From Figure 6.1 we can see the major rivers emptying into the Bay, including the James, York, Rappahannock, Potomac, Patuxent, Patapsco and Susquehanna from the west side and the Pocomoke, Wicomico, Nanticoke, Choptank and Chester from the east side.

6.2 Oxygen depletion problem

The Chesapeake Bay was formed about 10000 years ago when glaciers melted and flooded the Susquehanna River valley. It is one of the most productive and complex ecosystems in the US. As all the estuaries, the Bay provides a wide variety of habitats, with thousands species of animals and plants. The Bay is a very important economic resource for the zone: around 500 million euros of seafood per year are produced in the Chesapeake Bay. The extreme use of the land around

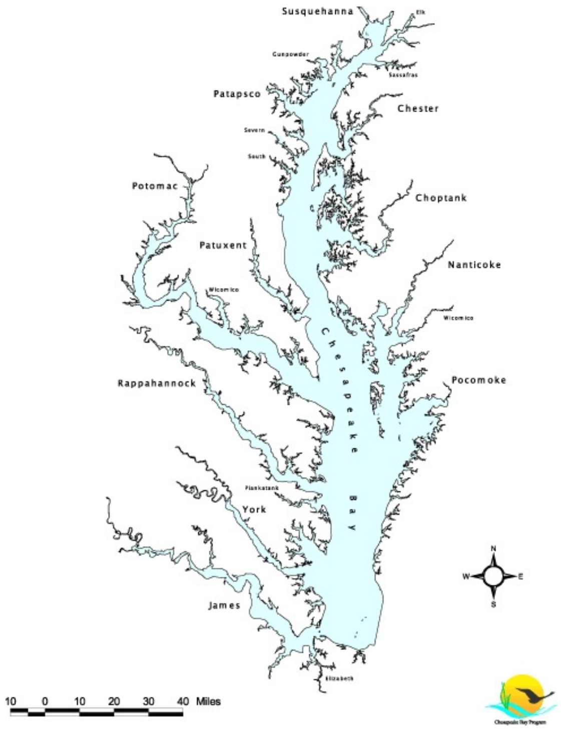

Figure 6.1: *Chesapeake Bay with the names of its main rivers (image from the https://www.chesapeakebay.net/ website).*

the estuary and, in particular, the pollution of close farms and cities changed the Bay over the years. Human activities caused a drastic reduction of oxygen, which must be present underwater, in dissolved form, to guarantee the life of most marine species.

The Bay's degradation problem led to the birth, in 1983, of the *Chesapeake Bay Program* (CBP). The CBP is a regional partnership whose partners include federal and state agencies, local governments, non-profit organizations and academic institutions. Its aim is to provide a support for the Chesapeake Bay restoration and protection activities.

The most critical areas of the Bay, with the lowest values of DO, are called *Dead zones*. These are the areas of the estuary where the presence of oxygen in the water is below 2 milligrams per litre. In these areas most of the marine species cannot move quickly enough and, consequently, they usually suffocate. The presence of these *Dead zones* is not a specific characteristic of the Chesapeake Bay. The distinctive fact is that in many of the other estuaries the causes of the presence of dead zones are related to natural phenomena (such as, e.g., the decomposition of algae blooms), while in the Chesapeake Bay the main cause is the pollution generated by human activities. For this reason, the CBP has a supervisioning goal for avoiding the development of these zones, by controlling the activities of urban areas near the Bay's shoreline.

In this work we apply the Kriging-RDDs method to predict the value of dissolved oxygen in the Chesapeake Bay given the data observed at different monitoring stations within the estuary.

6.2.1 Dataset

The dataset includes the value of dissolved oxygen (DO) measured at 119 monitoring stations across the Bay and the coordinates related to these stations (Figure 6.3, top left panel). The data refer to the year 2005 and were extracted from the US Environmental Protection Agency Chesapeake Bay Program (US EPA-CBP) database.
In general, the oxygen depletion problem becomes a very critical situation during the summer months. Indeed, in May the heavy spring rains start bringing all the fertilizers used by the farms and suburban areas in the water of the Bay, with the consequence of a dramatic DO reduction during the following summer months. For this reason we focused our study on the values of DO measured during the months from May to August.

6.2.2 Kriging prediction of DO values via RDDs method

The Chesapeake Bay shows highly irregular boundaries which do not allow to formulate a global stationarity assumption over the whole region. Indeed, even if few kilometres separate two points lying on adjacent and *parallel* tributaries, these rivers are separated by long and narrow areas of land. These areas of land represent barriers for the distribution of many aquatic variables. The different uses of the watersheds from people who live near to the estuary reflect different chemical and biological characteristics in the water of adjacent tributaries. Moreover, the Bay with all its tributaries cover a region of large size. Therefore, the size of the domain, as well as the land-barriers between adjacent tributaries do not allow to employ approaches based on a global model for spatial dependence. Instead, the use of local models can be employed to describe the variability in the spatial dependence structure induced by the complex and irregular properties of the domain.
Furthermore, even if local approaches are employed for the analysis, the presence of barriers is not taken into account if the Euclidean metric is used for the definition of the separation distance between points. For this reason, we opted to perform Kriging prediction via RDDs by using a graph-based metric approach in addition to the standard Euclidean approach. The construction of the graph is based on the Delaunay triangulation of the domain, as described in Chapter 5. Since the boundary of the Chesapeake Bay is highly jagged, we considered a simplified description of the Bay's border in order to define the triangulation. The boundaries have been defined by using straight segments, such that they could be approximate the real boundary of the estuary, taking into account the

presence of all the main tributaries and the land areas which separate them, but without considering the short and tight channels coming out from the tributaries and from the central unit of the Bay. The definition of this simplified boundary required the addition of points which must be considered only for the Delaunay triangulation, but they will not be taken into account for the analysis.
The available observations come from the monitoring stations located in the Bay. These stations are only 119, therefore the size of the dataset is very small, in particular if it is compared with the size of the whole estuarine system. For this reason a triangulation, whose nodes are only the observed locations, may imply the definition of very elongated triangles. Therefore, we opted for a constrained Delaunay triangulation: some points (which will not be considered for the analysis) have been added with the aim to define a set of triangles without too small angles. The top-right panel of Figure 6.3 shows the constrained Delaunay triangulation of the domain, where the grey points indicate the points which have been added to get a denser triangulation and for the boundaries definition. Furthermore, the constrained triangulation gives the possibility to better approximate the adjacency relationships between points within the estuarine region.

Moreover, the few available observations led us to the use of a kernel function to make more robust the estimation of variograms within sub-regions with low sampling density. We recall that, from Figure 3.3 in Chapter 3, we observed that the values $\epsilon = 1$ assigns high kernel weights to information coming from data very

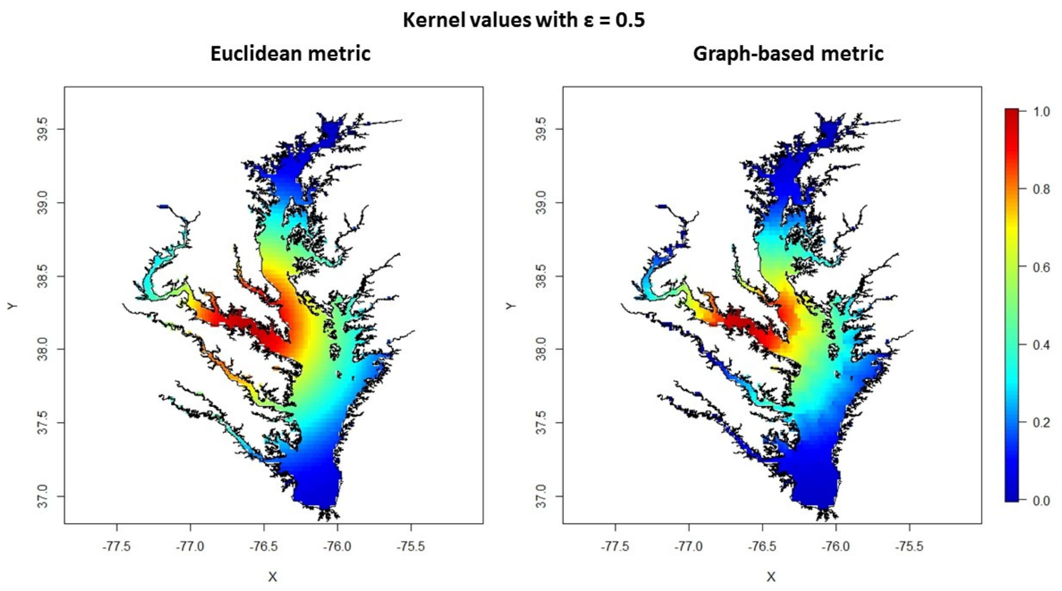

Figure 6.2: *Gaussian kernel values with bandwidth parameter* $\epsilon = 0.5$. *Left panel: Euclidean metric, right panel: graph-based metric.*

far away from the center where the local variogram must be estimated, while the value $\epsilon = 0.5$ seems to be a good compromise between the need to provide robust variogram estimates and the need of considering only information from *close* observations. Figure 6.2 shows the difference between Gaussian kernel weights obtained by using the Euclidean metric (left panel) and those obtained with a graph-based metric (right panel), when $\epsilon = 0.5$. In particular, we can observe that the use of a graph-based metric guarantees that a local variogram estimate centered in the black point of the Potomac river will not be based on information coming from monitoring stations of the Rappahannock river (on the South). Indeed, the weights assigned to these points are very close to zero. Furthermore, smaller kernel values, with respect to those based on an Euclidean approach, are assigned to the points of the Patuxent river (on the North with respect to the black point). Hence, the use of a graph-based metric gives the possibility to take into account the irregular shape of the region and all the land-barriers between adjacent tributaries.

The central and bottom panels of Figure 6.3 show the predicted values of DO obtained by using Ordinary Kriging, an Euclidean RDDs and a graph-based RDDs approach, respectively. In the latter two cases the domain has been partitioned into $K = 32$ sub-regions, the *weak* and local analyses have been repeated $B = 100$ and the final aggregation has been computed according to an "equal" average.

The most evident differences regard the values of DO close to the mouth of the Choptank river (on the north-east side) where a RDDs approach assigns higher values with respect those assigned by Ordinary Kriging predictions. Differences in the predicted DO values also regard the locations of the main tributaries on the west side of the Bay.

To investigate the differences in the results between a Kriging-RDDs approach based on the Euclidean distance and the one based on a graph-metric we performed cross-validation. For each repetition 10% of the observations has been left out from the analysis and used as test set, while the remaining 90% of observations has been used as training set. Figure 6.4 refers to the boxplots of the MSPEs obtained through $M = 100$ repetitions, where each repetition computed the prediction errors on different test sets left out from the training set. These boxplots do not show a strong improvement in term of MSPE values for a Kriging-RDDs approach with respect to those obtained with Ordinary Kriging (case $K = 1$). In addition, even the differences between Kriging-RDDs with Euclidean distance or with a graph-based metric are not so evident.

The general cross-validation method assumes that the errors are independent. This is not a viable assumption if the data are spatial distributed, since we know that in this case a spatial dependence exists among the observed locations. For this reason we tried to investigate in which parts of the study region the Euclidean (or the graph-based) Kriging-RDDs method generates the smallest

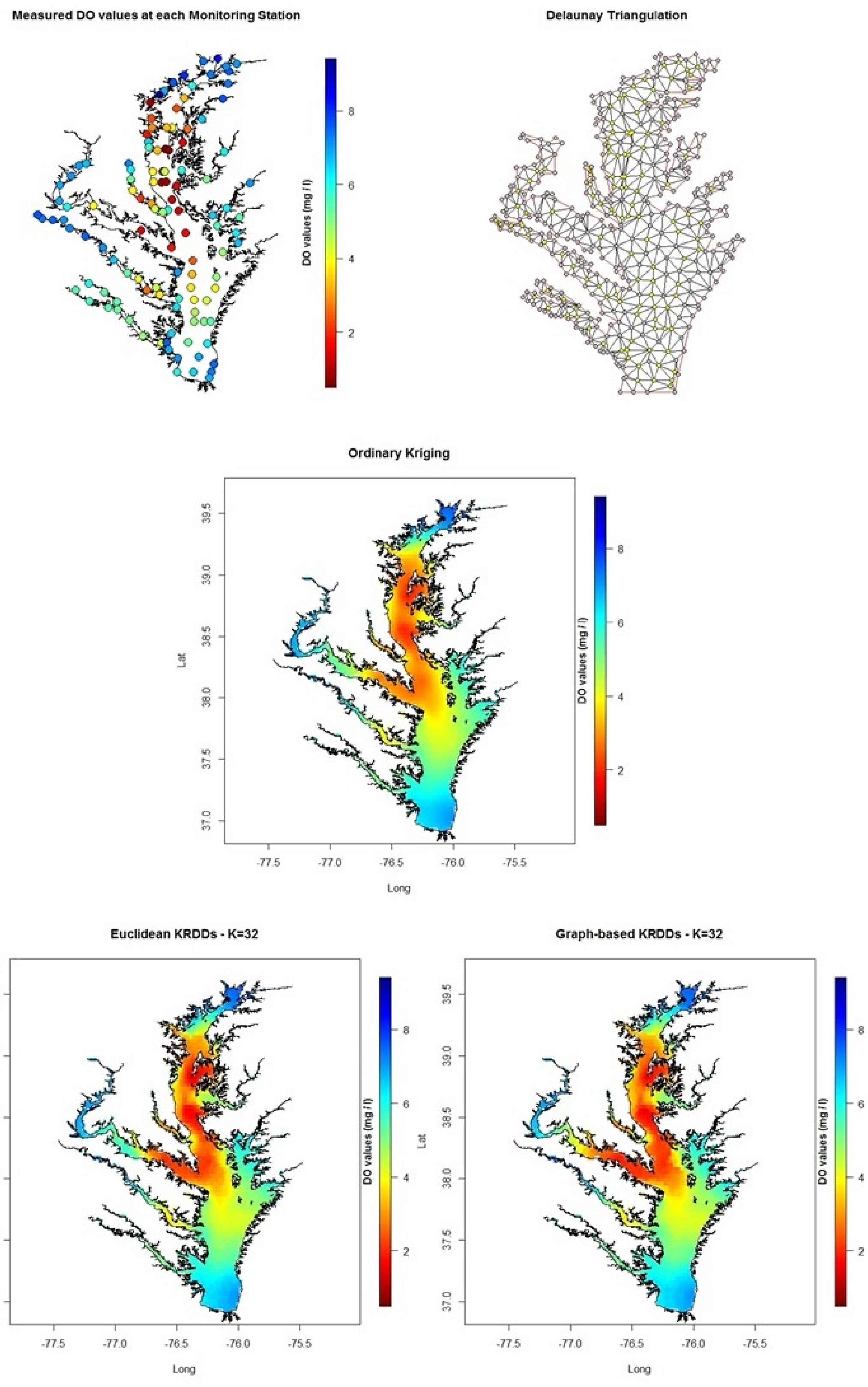

Figure 6.3: *Chesapeake Bay. Top-left panel: monitoring stations. Top-right panel: constrained Delaunay triangulation. Central panel: Ordinary Kriging prediction. Bottom panels: Euclidean and graph-based RDDs kriging prediction with $K = 32$.*

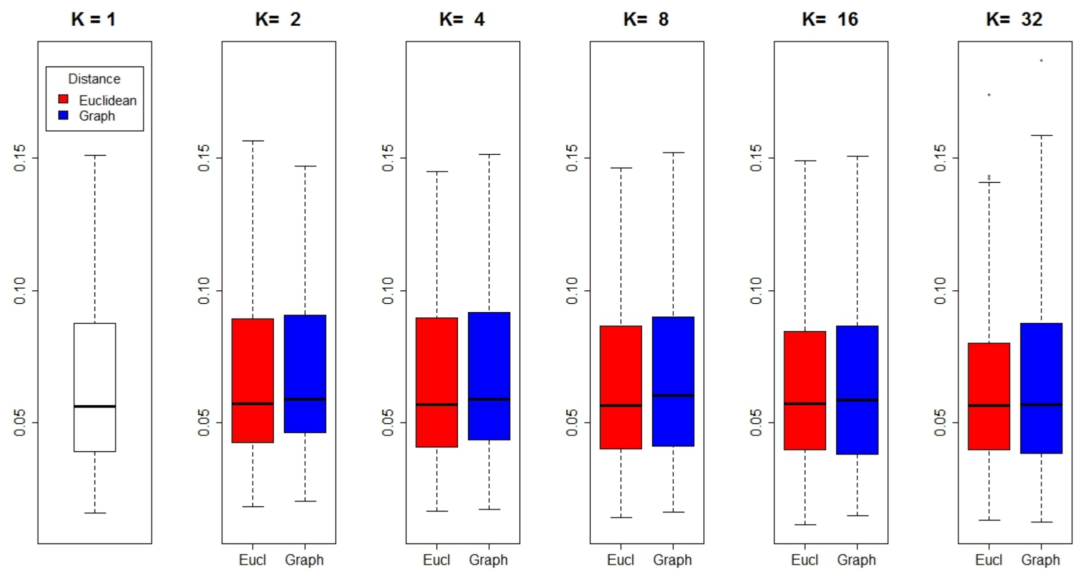

Figure 6.4: *Kriging-RDDs in the Chesapeake Bay: MSPE boxplots obtained via cross-validation.*

Distance	**K**					
	1	2	4	8	16	32
Euclidean	0.0650	0.0669	0.0663	0.0651	**0.0642**	0.0642
Graph	0.0650	0.0687	0.0684	0.0677	0.0657	**0.0647**

Table 6.1: *Chesapeake Bay. Kriging-RDDs method with different distances: Average of the MSPEs over the $M = 100$ repetitions.*

values of prediction errors.
The goal is the define a spatial map of the mean predictions errors. Hence, at each point is assigned the average of the prediction errors obtained through the M repetitions. The final average errors for the sampled locations are computed on sets of different dimensions, which depend on the number of times in which a given point belonged to the test set. Figure 6.5 shows the values of these mean cross-validation (MCV) errors for each monitoring station.
The Kriging-RDDs results reported by Figure 6.5 are based on random partitions of the domain into $K = 32$ sub-regions. Smaller values of parameter K do not show obvious differences between the approaches based on two different metrics. We can observe that with a graph-based Kriging-RDDs approach the errors of points within the tributaries on the west side are very close to zero (in particular those on the York and Rappahannock rivers). The errors for the central points of

the Potomac river are generally higher, but still much smaller if the Kriging-RDDs approach is based on the graph metric (MCV errors between 0.092% and 0.184%) than on the Euclidean distance (MCV errors between 0.184% and 0.828%).
For the monitoring stations within the tributaries on the north-east region of the Bay the conclusions are different. Here, the Kriging-RDDs approach based on the Euclidean distance shows better results in terms of MCV errors. A possible reason can be related to the presence of few monitoring stations within these rivers. Here, if the graph-based metric is used, only few observations are considered to estimate local variograms. Instead, the Euclidean RDDs approach gives the possibility to ground local variogram estimates on a larger set of observations coming from monitoring stations which are far away, but which registered similar values of DO. A possible solution to this situation may be the modulation of the kernel bandwidth such that each neighborhood contains a comparable number of observations.

Another possibility is to ground the kernel definition on prior knowledge of

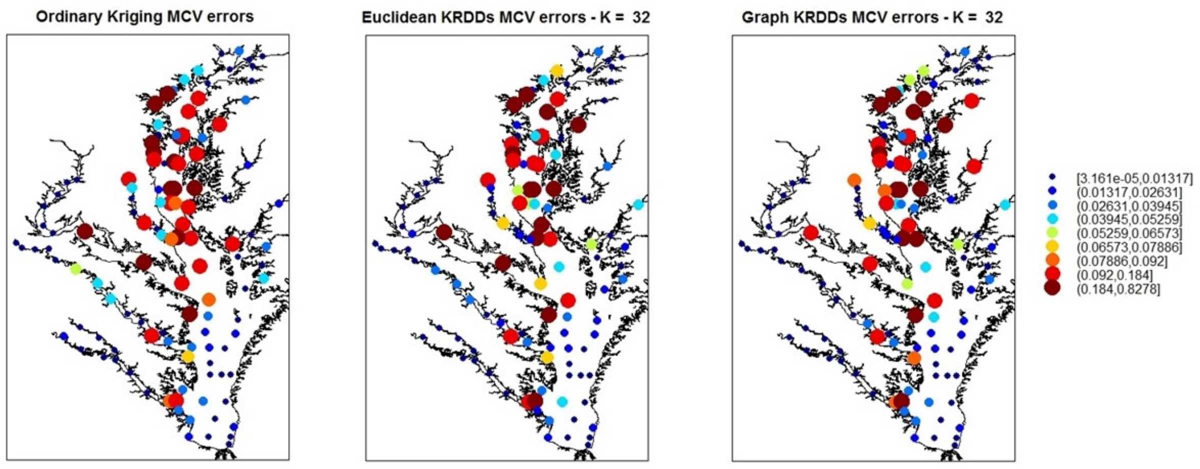

Figure 6.5: *Chesapeake Bay. Mean cross-validation (MCV) errors for each sampled point.*

the study phenomenon. Scully [2010] demonstrated that the depletion, or the growth, of dissolved oxygen in the water of Chesapeake Bay is influenced by the direction of the wind. In particular, winds from the South have an important effect for the increase of dissolved oxygen in the *dead zones* of the Bay, while winds of the west are shown to be less effective. This dependence of the oxygen dynamics in Chesapeake Bay on wind direction can be used as information for the definition of the kernel function. Indeed, a possible way can be to define an anisotropic kernel depending on the summertime wind direction.

Conclusions

In this dissertation we propose a methodology for the analysis of spatially distributed random fields when complexities associated with the observed data and/or with the domain make inappropriate the use of classical techniques usually employed in geostatistical settings. Our work gives a contribution to the research in the O2S2 direction, whose focus is on the analysis of spatially distributed object data (e.g., functional and/or constrained data, tree-structured data, elements of a manifold or images). In particular the proposed method is able to handle situations in which one cannot assume the existence of a global stationary model describing the spatial field underlying the object data.
The impossibility of defining a global stationary model motivated us to explore the possibility of employing local models. We propose to perform repeated domain partitions, through a *Random Domain Decompositions* (RDDs) approach. This choice was motived by the will of defining a non-parametric methodology which does not require strong assumptions on the distribution generating the data, hence prone to be applied for the analysis of object data. We assume that each realization of a random domain partition defines a system of homogeneous neighborhoods, which represents the starting point for performing local analyses. Therefore, local and simple analyses are performed instead of a global and complex non-stationary analysis. According to a bagging approach, the repeated analyses are then aggregated into a final analysis.
Here, we applied the RDDs methodology to perform Kriging predictions over a possibly complex spatial domain. Nevertheless, the method is entirely general and it can be employed for different types of analysis (e.g., classification, drift estimate). Furthermore, with respect to existing partitioning-based approaches used to handle non-stationary random fields, the RDDs method overcomes a critical drawback: the presence of discontinuities over the prediction maps. Indeed, discontinuity in correspondence of the boundary between two disjoint sub-regions are here smoothed away by the aggregation phase, that consists of averaging the multiple bootstrap replicates. Moreover, we propose a possible solution to overcome the independence hypothesis between disjoint neighborhoods within the domain, which is often assumed in many partitioning-based approaches. The solution is grounded on the use of a geographically weighted estimate for local variograms: within a neighborhood the variogram estimate is based on both the observations

within the neighborhood itself, and on those within neighboring sub-regions, the weights being defined through a kernel weighting function. Here, the use of a kernel function contributes to the flexibility and robustness of the analysis, in particular for those sub-regions with few observations.
The RDDs method, together with the use of kernel weighted variogram estimates, can be employed for the analysis of directional data. In recent years an important area of statistics has focused on the study of methods able to analyse directional data, typical of many scientific fields (e.g., astronomy, geology, biology, medicine and meteorology), where each observation is recorded as a direction, i.e., a vector with an angle of rotation and a length. In this case, the directional property of the datum does not allow to assume stationarity, hence a RDDs can be employed. Moreover, even if for the simulation examples, as well as for the case study described in this dissertation, we used a Gaussian kernel, in general one can take advantage from a possible prior knowledge on the study phenomenon for the definition of a more appropriate type of kernel. The use of an anisotropic kernel allows to incorporate information on a possible spatial directional dependence existing among the observations. For example, the depletion problem of dissolved oxygen within the Chesapeake Bay is influenced by the summertime wind direction, as mentioned in Chapter 6. This information was used by Scully [2010] to model the average dissolved oxygen concentration. His results are showed in Figure 6.6 and could eventually drive the definition of the spatial kernel.

The possible non-convex geometry of the domain may require the use of a non-Euclidean metric as measure of the adjacency relations among the locations. The Euclidean metric is commonly used in Geostatistics, and the validity of the most theoretical variogram models (i.e., those mentioned in Chapter 1, Section 1.1.3) is guaranteed only if the variogram estimates are based on the Euclidean distance. The RDDs method is able to deal with possible complex domains with a simple approach by combining two different metrics. Indeed, the partitions of the domain are based on a non-Euclidean metric, which takes into account the geographical constraints of the domain. Once a system of homogeneous sub-regions has been defined, within each neighborhood the adjacency relationships are described by the Euclidean distance, avoiding to incur non valid local variograms. Here, for the study of complex domains we propose to employ a non-Euclidean metric based on the shortest path computed on a mesh which is used to triangulate the study region (e.g., we proposed the use of the Delaunay triangulation).
However, an interesting future direction of research may concern the development of a theory to perform Kriging prediction directly on the graphs by using a graph-based metric. In this case, the aim to predict the value of a variable at an unsampled location of the domain is replaced by the goal to make prediction of the value of a feature at one node in the graph. Xu et al. [2010] established some connections between the Kriging technique and several semi-supervised learning approaches for prediction on graphs, which estimates the correlation structure

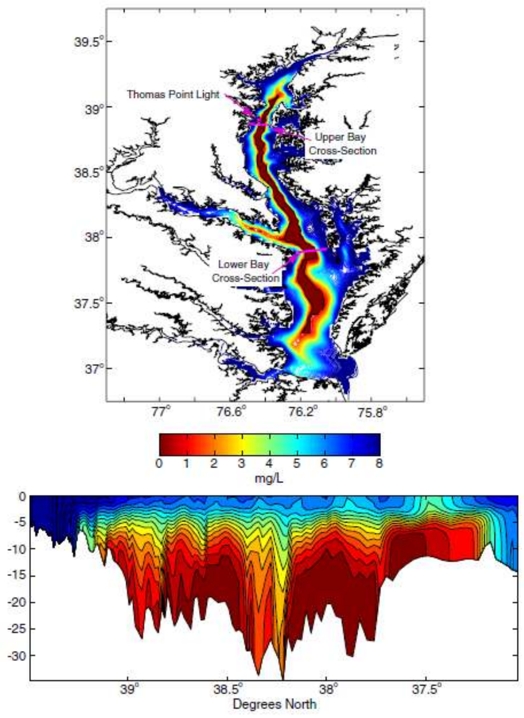

Figure 6.6: *Chesapeake Bay: modeled average of dissolved oxygen concentration using information on the wind directions. Figure extracted [Scully, 2010, pag. 1167]*

between values observed at the vertices of the graph and uses it to make smooth prediction respect to a notion of distance on the graph.
Another possible future development for the RDDs method may concern the type of decomposition used to define the partitions of the domain of interest. In all the examples described in this dissertation we used a simple Voronoi tessellation of the domain, although this can be computed in different ways. Moreover, we randomly and uniformly sampled the Voronoi nuclei among the set of available observations, but one could make more precise assumptions on the distribution of the sampled points (e.g., in case they are irregularly spaced within the region) and use non-homogeneous schemes to select the neighborhoods centers (e.g., by employing a tessellation based on non-homogeneous Poisson processes or more complex point patterns).
An improvement to the RDDs method may concern the addition of covariates both for the Kriging prediction and for the definition of the kernel weights. In fact, in all the examples of this dissertation, we based the Kriging predictions and kernel weights only on information regarding the positions of the locations. Instead, one could take advantage even from other information which can be measured at these locations. For example, for the dissolved oxygen study one may consider information coming from the values of water temperature, salinity and dissolved nutrient concentrations often collected at each monitoring station, and use them

for the definition of an empirical model to predict dissolved oxygen levels within the Chesapeake Bay, as proposed by Prasad et al. [2011].
Further developments will concern the use of a data-driven approach for setting the value of the kernel bandwidth. The parameter ϵ - which controls the range of influence of the observations on the estimate of local variograms - can be set by using cross-validation, as mentioned in Chapter 4. It may be influenced by the value chosen for the parameter K or, more precisely, by the observation density. Tavakoli et al. [2016] pointed out the possible drawbacks associated with the use of a fixed bandwidth parameter: an oversmoothing problem can occur in particular within regions with denser observations, while an undersmoothing problem may concern neighborhoods with few observations. Tavakoli et al. [2016] proposed the idea of adjusting the bandwidth parameter on the basis of the number of available observations: the parameter is then adapted by using a modulation of a global bandwidth, on the basis of the distance from the k-th nearest *location* (or *observation*), guaranteeing that a comparable number of observations is used for the estimate at each point. This solution can be useful even when the observations are not uniformly distributed over the domain.
A further important development regarding the possibility of improving the proposed method may concern the formalization of an aggregated final covariance function defining the structure of spatial dependence on the whole domain. This aggregated covariance could be useful to obtain an explicit expression for the variance of the aggregated prediction error $\epsilon^*_{\mathbf{s}_0}(\mathbf{Z}; P) = Z_{\mathbf{s}_0} - Z^*_{\mathbf{s}_0}(\mathbf{Z}; P)$ which has been mentioned in Chapter 5, Section 5.6. Such variance would be key to support the uncertainty assessment for Kriging-RDDs, besides allowing for further inferential tools for the scope of prediction (e.g., Kriging prediction bands).

Appendix A

R Codes

The proposed RDDs methodology for performing Kriging prediction over simple and complex domains has been implemented with R [R Development Core Team, 2008]. This appendix includes the codes implemented in R of the main functions to perform all the steps of the RDDs methodology described in Chapters 4 and 5.

A.1 Kernel values

The function `kerfn` computes the values of the kernel for sampled (or unsampled) points given a reference center (e.g., a Voronoi nucleous).

```
kerfn= function(newdata,center,dist='euclidean',ker.type='Gau',
                param,distance.matrix = NULL)
{
# INPUT:
# newdata = matrix with the coordinates points
# center =  it's a 3-dimension vector:
#           x[1] = x-coordinate,
#           x[2] = y-coordinate,
#           x[3] = the row-index of the center in
#                  the matrix of data
#
# dist = method to compute distances
#        ('euclidean' or 'graph.distance')
#
# ker.type = type of kernel (only Gaussian kernel implemented so far)
#
# param = parameters that define the kernel (bandwidth parameter)
#
# distance.matrix is needed only if dist = 'graph.distance'
# distance.matrix :
#        1. if the points are observed locations (data points)
#           distance.matrix is a nsub*nsub distance matrix
#           such that: the (i,j) element is the length of the
#           shortest path between the i-th and j-th observed
#           points on the Delaunay graph;
#        2. if the points are unobserved locations (grid points)
#           distance.matrix is a K*ngrid distance matrix
#           such that the (k,l) element is the distance
#           on the Delaunay graph between the k-th center
#           and the l-th grid point.
# OUTPUT:
#
# value of the kernel function
center = as.numeric(center)
if(dist == 'euclidean'){
  d=as.matrix(dist(rbind(center[1:2],newdata), 'eucl'))[1,-1]
}

if(dist == 'graph.distance'){
  # d = k-th row of the distance matrix
  d = distance.matrix[as.integer(center[3]),]
}

if(ker.type!='Gau')
{
  print('Error: only Gaussian kernel
        implemented so far (*Gau*)')
}
eps=param[1]
return(exp(-1/(2*eps^2)*(d^2)))
}
```

A.2 Delaunay graph

Given the Delaunay triangulation of the domain, an undirected graph has to be constructed which includes the same set of vertices and edges defining the triangulation. The weight of each edge is its length. The function `distance.on.graph` defines an $(nsub \cdot nsub)$-dimensional graph-based distance matrix.

```
distance.on.graph = function(data, mesh)
{
# INPUT
# data = data matrix
# mesh = mesh obtained via Delaunay triangulation
#        (with tringulate function of RTriangle package)
# OUTPUT
# graph.distance =
#     nsub*nsub-dimensional symmetric matrix.
#     The element (i,j) is the shortest length path
#     between the i-th and j-th points
#     on the Delaunay graph.

edge.list = mesh$E
edge.num = dim(edge.list)[1]
# edges.list: edge.num * 2-dimensional matrix
#            (each row contains the indices of the nodes
#             i and j defining the edge e = {ij})


# Construct the undirected graph
graph = graph_from_edgelist(edge.list, directed = FALSE)

# Given two linked nodes i and j
# the weigth of edge (i,j) is:
# w(i,j) = euclidian distance (i, j)
edge.weights = rep(0, edge.num)

for(l in 1:edge.num){
  node.i = edge.list[l,1]
  node.j = edge.list[l,2]
  point.i = data[node.i,1:2]
  point.j = data[node.j,1:2]
  edge.weights[l] = dist(rbind(point.i,point.j),
                         method = 'euclidean')
}

graph.distance = distances(graph, v = V(graph),
                           to = V(graph),
                           mode = c("all"),
                           weights = edge.weights,
                           algorithm = c("automatic"))
return(graph.distance)
}
```

A.3 Voronoi tessellation

Given a set of Voronoi nuclei, for each sampled point the function `datapoints2centers.distance` computes the distance vector between the point and each center of the Voronoi nuclei set. If the graph-based metric is employed, the assignment of an unsampled point to a Voronoi nucleous will require: (1) finding the triangle of the Delaunay triangulation where it is located (this assignment is computed by the `grid2triangle` function); and (2) computing the distance between the point and each center according to the equation (5.1) if the grid point is located within a triangle, to the equation (5.2) if it is located on a common edge shared by two triangles (`gridpoints2centers.distance`).
Given a K-dimensional vector with the distances between a sample/unsampled point and each Voronoi center, the `assign2center` function returns the Voronoi nucleous to which the point is assigned.

```
datapoints2centers.distance = function(x,centers,
                                       method='euclidean',
                                     graph.distance=NULL)
{
# This function compute the K-dimensional
# distance vector
# (according to the specified method) between
# an observed point and each center
# INPUT :
# x= it's a 3-dimension vector
#     (x[1] = x-coordinate,
#      x[2] = y-coordinate,
#      x[3] = it indicates the row-index
#             in the data matrix corresponding
#             to the observed point)
#
# centers= K*3-dimensional matrix
#         (centers[k,1:2] = coordinates of the k-th center,
#          centers[k, 3] = indicates the row index
#                 in the data matrix of the k-th center)
#
# method= 'euclidean' or 'graph.distance'
#
# graph.distance= nsub*nsub matrix
#                 (graph.distance(i,j) =
#                  shortest path between i and j on
#                  the Delaunay graph)
# OUTPUT
# d = K-dimensional vector with the distance
#     between the x-point
#     and each of the selected Voronoi nuclei
x = as.numeric(x)
if(method=='euclidean'){
  d = as.matrix(dist(rbind(c(x[1],x[2]),centers[,1:2]),
                     method))[1,-1]}
if(method=='graph.distance'){
  ncenters = dim(centers)[1]
  d = rep(NA, ncenters)
  for(i in 1:ncenters){
    d[i] = graph.distance[x[3],centers[i,3]]}
}
return(d)
}
#############################################################
grid2triangle = function(grid.matrix, data.matrix, mesh)
{
# INPUT:
# grid= ngrid*2-dimensional matrix
#       (the i-th row contains the
#       coordianates of the i-th grid point)
# data= nsub*2-dimensional matrix
#       (the i-th row contains the
#       coordinates of the i-th data point)
# mesh= mesh obtained with Delauny Triangulation
#       (using tringulate function of
#        RTriangle package)
# OUTPUT:
# assign.matrix=
#        ngrid*num.triangles-dimensional matrix
#        (the (i,j) elemenent = TRUE if the point grid[i,]
#         is inside the j-th triangle, FALSE otherwise)
# no.assg.complet=
#        indices of grid points that were not assigned
#        to any triangle

triangles = mesh$T
num.triangles = dim(triangles)[1]

ngrid = dim(grid.matrix)[1]
assign.matrix = matrix(NA, ngrid, num.triangles)

for(j in 1:num.triangles){
  bnd.triangles = triangles[j,]
  assign.matrix[,j]=
    in.out(bnd = cbind(data.matrix[bnd.triangles,1],
                       data.matrix[bnd.triangles,2]),
           x = cbind(as.numeric(grid.matrix[,1]),
                     as.numeric(grid.matrix[,2])))
}
check.complete = rep(NA, ngrid)
for(i in 1:ngrid){
check.complete[i] = length(which(assign.matrix[i,]==TRUE))
}
no.assg.complete = which(check.complete == 0)

output = list()
output[[1]]=assign.matrix
output[[2]]=no.assg.complete

return(output)
}
#############################################################
gridpoints2centers.distance = function(x, centers,
                                  method='euclidean',
                                  graph.distance=NULL,
                                  assign.matrix = NULL,
                                  data.grid.distance = NULL,
                                  triangles = NULL)
{
# INPUT :
# x= it's a 4-dimension vector
#     (x[1] = x-coordinate,
#      x[2] = y-coordinate,
#      x[3] = row-index of the gridmatrix
#              corresponding x[1:2] coodinates,
#       x[4] = 0 if the point is UNSAMPLED locations is 0,
#              otherwise it indicates the
#              row-index of the data matrix
#              corresponding to x[1:2] coodinates)
#
# centers= K*3 matrix
#         (centers[k,1:2] = coordinates of the k-th center,
#          centers[k, 3] = row-index of the data matrix
#                          corresponding to the k-th center)
#
# method= 'euclidean' or 'graph.distance'
#
# graph.distance= nsub*nsub-dimensiona matrix
#                (graph.distance(i,j) = shortest path
#                 between i and j on
#                 the Delaunay graph)
#
# assign.matrix= ngrid*nuum.traingles-dimensional matrix
#               (assign.matrix[i,j] = 1 if the i-th
#                grid point is assigned to the
#                j-th triangle, 0 otherwise)
#
# data.grid.distance= nsub*ngrid-dimensional matrix
#                    (data.grid.distance(i,j) is the
#                     euclidean distance
#                     between the i-th observed point and
#                     the  j-th grid point)
#
# triangles= num.triangles*3 dimensional matrix
#            (the i-th row contains the 3 indices of
#             the 3 vertices defining the i-th triangle)
# OUTPUT :
# d : K-dimensional vector (d[k] is the distance between
#     the point x and the k-th center)
x = as.numeric(x)
ncenters = dim(centers)[1]
if(method=='euclidean'){
d = as.matrix(dist(rbind(c(x[1],x[2]),centers[,1:2]),
                   method))[1,-1]}
if(method=='graph.distance'){
# If x[4] > 0 it means that the grid point
# is an already observed location.
# Hence, it's a vertex of the graph, the distance between
# the point and each center is computed directly as
# shortest distance path on the Delaunay graph
if(x[4] > 0){
  d = rep(NA, ncenters)
  for(i in 1:ncenters){
    d[i] = graph.distance[x[4],centers[i,3]]}
}
```

```
# If x[4] == 0 it means that the grid points is
# an unobserved location
if(x[4] == 0){
  # ind.tringles containes the index/indices
  # of the row/rows of the triangles matrix
  # corresponding to the triangle/s where the point is located
  ind.triangle = which(assign.matrix[x[3],]==1)
  # It can be assigned to one and only one triangle
  # (i.e., the point is inside the triangle) or
  # it can be assigned to two adjacent triangles
  # (if it's on the common border of the triangles)
if(length(ind.triangle) == 1){
  # d.Pj = euclidian distance between the point x and
  #       the j-th vertex of the triangle where it is located
  d.P1 = data.grid.distance[triangles[ind.triangle,][1],x[3]]
  d.P2 = data.grid.distance[triangles[ind.triangle,][2],x[3]]
  d.P3 = data.grid.distance[triangles[ind.triangle,][3],x[3]]
  d = rep(NA, ncenters)
  for(i in 1:ncenters){
   d[i] = min(d.P1 + graph.distance[triangles[ind.triangle,][1],
                                    centers[i,3]],
              d.P2 + graph.distance[triangles[ind.triangle,][2],
                                    centers[i,3]],
              d.P3 + graph.distance[triangles[ind.triangle,][3],
                                    centers[i,3]])
  }
}
if(length(ind.triangle) == 2){
  # There are 4-vertices to evaluate d.Pj
  ind.vertex = union(triangles[ind.triangle[1],],
                     triangles[ind.triangle[2],])
  d.P1 = data.grid.distance[ind.vertex[1],x[3]]
  d.P2 = data.grid.distance[ind.vertex[2],x[3]]
  d.P3 = data.grid.distance[ind.vertex[3],x[3]]
  d.P4 = data.grid.distance[ind.vertex[4],x[3]]
  d = rep(NA, ncenters)
  for(i in 1:ncenters){
    d[i] = min(d.P1 + graph.distance[ind.vertex[1],centers[i,3]],
               d.P2 + graph.distance[ind.vertex[2],centers[i,3]],
               d.P3 + graph.distance[ind.vertex[3],centers[i,3]],
               d.P4 + graph.distance[ind.vertex[4],centers[i,3]])
  }
}
  # If the grid point is UNSAMPLED it cannot happen
  # that length(ind.triangle)>2, because otherwise it's a
  # vertex of a triangle,
  # hence it's a data point

  # There are some grid points which are not assigned
  # to any triangle. These grid points are not used
  # to perform prediction.
  if(length(ind.triangle)==0)
  {d=rep(NA, ncenters)}
}
}
return(d)
}
############################################################
assign2center=function(distance.vector)
{
# INPUT :
# distance.vector : K-dimensional vector
#                  (distance.vector[k] = distance between
#                   a fixed point and the k-th center)
# OUTPUT :
# the index of the center to which the
# point is assigned
# (i.e., an integer between 1 and K)

assigned.center = which.min(distance.vector)
if(!is.empty(assigned.center)){
  return(assigned.center)}
# If the distance.vector is a vector with only NA's
# it means that it's the grid point has not be assigned
# to any triangle, therefore it will not assigned to
# any Voronoi center
if(is.empty(assigned.center)){
  return(0)
}
}
```

A.4 Bootstrap step

The function `OKBV` (or its parallelized version `OKBVpar`) performs the bootstrap step of the method. In receives as input almost all the parameters chosen by the user (e.g., the number of sub-regions K, the number of bootstrap repetitions B, the type of kernel and its bandwidth parameter, a theoretical model to fit the empirical variograms and the metric for the definition of the RDDs and the kernel weights). In addition, if we want to use a graph-based metric, we have to give as input of the function a matrix which defines the positions of the grid points with respect to the triangles of the mesh and a matrix defining the distance between each pair of sampled points. Other needed parameters are described within the comments regarding the function `OKBV` reported below.
At each bootstrap iteration, the realization of a RDD is used as system of neighborhoods for performing K local analysis under local stationarity assumption. For each Voronoi tile (and for each bootstrap iteration), the function defines a geographically weighted estimate for the local variogram by using the kernel func-

tion. After the B repetitions, for each grid point the function returns the set of bootstrap predictors, the set of local variograms and the kernel values (i.e., the value $K_\epsilon(\mathbf{s}_0, \mathbf{c}^b)$, where $\mathbf{c}^b$ is the Voronoi center of the neighborhood where $\mathbf{s}_0$ was located at the b-th iteration).

```
OKBV = function(data, K, grid, nk_min=1, B=100, model,
                sill_ini=NULL, range_ini=NULL, nugget_ini=NULL,
                spdist='euclidean', suppressMes=F, tol=1e12,
                ker.width=0, mesh = NULL, graph.distance= NULL,
                assign.matrix = NULL, no.assg.grid = NULL,
                data.grid.distance = NULL, is.observed = NULL,
                border.length=0)
{
# INPUT:
# data = data.frame containing the data
#        (the first two columns contain the coordinates,
#         the third contains the data)
# K = number of neighborhoods
# grid = the grid of prediction
# nk_min = minimum number of observations within a
#          neighborhood (default: 1)
# B = number of bootstap iterations (default: 100)
# model = parametric variogram model
#         (as in package gstat - only Sph and Exp used so far)
# sill_ini, range_ini, nugget_ini = initial parameters
#      for variogram estimations. If NULL the inizial parameters
#      are estimated from the data
# spdist = method to compute distances in the space
#         (only 'euclidean' or 'graph.distance' implemented so far)
# suppressMes = {T, F} controls the level of interaction and warnings
# tol = parameter used in variogram fitting
# ker.width = parameter controlling the width
#              of the Gaussian kernel (if 0, no kernel is used)
# mesh = contains the parameters defining the Delaunay triagulation
# graph.distance = nsub*nsub dimensional graph-based
#                  distance matrix
# assign.matrix =  ngrid*num.triangles-dimensional matrix
#                 (the (i,j) elemenent = TRUE if the point grid[i,]
#                  is inside the j-th triangle, FALSE otherwise)
# no.assg.grid = indices of grid points that were not assigned
#                  to any triangle
# data.grid.distance = nsub*ngrid dimensional matrix
#                     (the (i,j) element is the distance beteween
#                      the i-th observed point and the j-th grid point.
#                      If data.grid.distance(i,j)==0 then the
#                      j-th grid point is an already observed value)
# is.observed = ngrid-dimensional vector.
#               If the i-th grid point is already
#               observed, then is.observed[i] contains the index
#               of the  corrisponding point in the data matrix,
#               otherwise is.observed[i]=0
# border.length = number of the NA-points used for the definition
#                  of the boundaries of the domain
#                 (the coordiantes of these points
#                  occupy the first border.length rows of the
#                  data matrix)
#
#
# OUTPUT:
# list(fpred, variofit=vfit), with
# fpred = list of B components, each containing a grid realization
# variofit = list of B components, each containing the
#             parameter of the variogram over the grid
#             (i.e., for each grid point, the parameter used
#              to make the prediction at that points are given)

# Set parameters and initialize lists
colnames(data)=c('x','y','z')
nsub = dim(data)[1]
fpredk=fpred=gridk=vfitk=vfit=list()
nk=rep(0,K)
coord.list=as.list(data.frame(t(cbind(data$x, data$y))))
coordg.list=as.list(data.frame(t(cbind(grid$x, grid$y))))
nugget_ini_user=nugget_ini
sill_ini_user=sill_ini
range_ini_user=range_ini
coordinates(data)=c('x','y')
ngrid=dim(grid)[1]
nstr=length(model)

for(b in 1:B) # Repeat over the B bootstrap replicates
{
if(suppressMes==F)
print(paste("Repetition B = ", b, sep=''))

# Step 1: Extract centers
ind=sort(sample((border.length+1):nsub, size=K))
# We need also the indices of the data matrix
# corresponding to each center
centers = cbind(data$x[ind],data$y[ind],ind)

## 1.1 Assign data to centers
assign = rep(0, nsub)
for(i in (border.length+1):nsub){
d = datapoints2centers.distance(x=cbind(data$x[i],data$y[i],i),
                                centers = centers, method = spdist,
                                graph.distance = graph.distance)
assign[i] = assign2center(d)}

while(min(table(assign))<nk_min)
# this is done only if a neighborhood has too few points
{
```

```
if(suppressMes==F)
  print("Repeating sampling")

ind=sort(sample((border.length+1):nsub, size=K))
centers = cbind(data$x[ind],data$y[ind],ind)
assign = rep(0, nsub)
for(i in (border.length+1):nsub){
  d = datapoints2centers.distance(x=cbind(data$x[i],data$y[i],i),
                                  centers = centers,
                                  method = spdist,
                                  graph.distance = graph.distance)
  assign[i] = assign2center(d)}
}

## 1.2 Assign grid points to centers
grid.new = cbind(grid, 1:ngrid, is.observed)
grid.new = data.frame(grid.new)
colnames(grid.new)=c('x','y','grid.position','is.observed')

assigng = rep(0, ngrid)
# graph.distance.grid.centers : K*ngrid-dimensional matrix
#           (the element (k,i) is the distance on the graph
#           between the k-th center and the i-th grid point)
# (This matrix will be used to compute
#  the kernel values if ker.width > 0)
graph.distance.grid.centers = matrix(NA, K, ngrid)
triangles = mesh$T

for(i in 1:ngrid){
d = gridpoints2centers.distance(x=grid.new[i,], centers = centers,
                  method = spdist,
                  graph.distance = graph.distance,
                  assign.matrix = assign.matrix,
                  data.grid.distance = data.grid.distance,
                  triangles = triangles)
graph.distance.grid.centers[,i]=d
assigng[i]=assign2center(d)
}

for(k in 1:K)
{ gridk[[k]]=grid[assigng==k,]
coordinates(gridk[[k]])=c('x','y')
}

# If a kernel is given, initialize the quantities
# for variogram estimate
if(ker.width>0)
{
v=variogram(z~1, data=data[which(!is.na(data$z)),], cloud=T)
vB=variogram(z~1, data=data[which(!is.na(data$z)),])
tmp=bbox(coordinates(data)[which(!is.na(data$z)),1:2])
```

```
cutoff=sqrt(diff(tmp[1,])^2+diff(tmp[2,])^2)/3
width=cutoff/(length(vB$dist))
punto_lag=cbind((0:(length(vB$gamma)-1))*width,
                (1:(length(vB$gamma)))*width)
kergrid=list()
}

for(k in 1:K) # repeat in each neighborhood
{
# Step 2.k: estimate the variogram
# Select the data of the neighborhood
datak=data.frame(data[assign==k,])[,1:3]
coordinates(datak)=c('x','y')
if(ker.width==0)
{ # if no kernel is given, compute
  # classical variogram estimator
  vk=variogram(z~1, data=datak)
}
if(ker.width>0)
{ # else compute the weighted estimator
  vkB=vB
  center = centers[k,]
  KER=kerfn(newdata=coordinates(data)[,1:2],
            center=center, dist=spdist, ker.type = 'Gau',
            param = ker.width, distance.matrix = graph.distance)
  KER=KER%*%t(KER)

  for(l in 1:dim(punto_lag)[1])
  {
    inVh=which(v$dist>=punto_lag[l,1] & v$dist<punto_lag[l,2])
    indKER=as.matrix(as.data.frame(v)[inVh,6:7])

    vkB$gamma[l]=sum(v$gamma[inVh]*KER[indKER])/(sum(KER[indKER]))
  }
  vk=vkB
}

# If the initial parameters are not given, estimate from the data
if(is.null(nugget_ini_user))
  nugget_ini=.25*median(c(rep(vk$gamma[1],vk$np[1]),
                        rep(vk$gamma[2],vk$np[2])))

if(is.null(sill_ini_user))
  sill_ini=median(c(rep(vk$gamma[length(vk$gamma)-1],
                        vk$np[length(vk$gamma)-1]),
                    rep(vk$gamma[length(vk$gamma)-2],
                        vk$np[length(vk$gamma)-2]),
                    rep(vk$gamma[length(vk$gamma)-3],
                        vk$np[length(vk$gamma)-3]),
                    rep(vk$gamma[length(vk$gamma)-4],
                        vk$np[length(vk$gamma)-4])))-nugget_ini
```

```
if(is.null(range_ini_user))
{
  if(nstr>1)
  { # if nested structure are given,
    # initial ranges are asked to the user
    print("Please provide initial range
          for variogram estimation")
    return(-1)
  }
  if(model=='Exp')
  {range_ini=vk$dist[min(which(vk$gamma>.95*
                              (sill_ini+nugget_ini)))]/3
}else range_ini=vk$dist[min(which(vk$gamma>(sill_ini+nugget_ini)))]
}

# Build initial variogram
vm=vgm(sill_ini,model[1],range_ini[1],nugget_ini[1])
nstr=length(model)
if(nstr>1)
{for(j in 2:nstr)
  vm=vgm(psill = sill_ini, model = model[j],
         range = range_ini[j], add.to=vm)}

# Fit variogram (several attempts are done,
# it none of them works the initial variogram is used)
vk.fit = try(fit.variogram(vk, vm, fit.sills=T, fit.method = 7,
                           debug.level = 1), silent = FALSE)
if(class(vk.fit)[1]=="try-error")
{
  print("Trying fitting again")
  vk.fit = fit.variogram(vk, vm, fit.sills=T,
                         fit.method = 2, debug.level = 0)
}
if(vk.fit$psill[1]<0 | vk.fit$range[2]>tol)
{
  print("Trying with fixed nugget")
  sill_ini=sill_ini+nugget_ini
  nugget_ini=0
  vm=vgm(sill_ini,model[1],range_ini[1],nugget_ini[1])
  nstr=length(model)
  if(nstr>1)
  {for(j in 2:nstr)
    vm=vgm(psill = sill_ini, model = model[j],
           range = range_ini[j], add.to=vm)
  }
  vk.fit = try(fit.variogram(vk, vm, fit.sills = c(0, rep(1, nstr)),
                             fit.method = 7, debug.level = 1))
  if(class(vk.fit)[1]=="try-error")
  {
    print("Trying fitting again")
    vk.fit = fit.variogram(vk, vm, fit.sills = c(0, rep(1, nstr)),
                           fit.method = 2, debug.level = 1)
  }
}

vk.fit$range[vk.fit$range<0]=range_ini[which(vk.fit$range<0)-1]
if(sum(vk.fit$psill)==0)
{
  print('Warning: unable to fit the variogram, using the initial one')
  vk.fit=vm
}

# Step 3.k: perform kriging (within the neighborhood)
datak.gstat = gstat(formula = z ~ 1,
                    data = datak, nmax = 50, model=vk.fit,
                    set = list(gls=1))
fpredk[[k]] = predict(datak.gstat, gridk[[k]],debug.level=0)
if(ker.width>0)
{
if(spdist=='euclidean'){
  kergrid[[k]]=kerfn(newdata=coordinates(gridk[[k]])[,1:2],
                     center=c(center[1],center[2],k),
                     dist = spdist,
                     ker.type = 'Gau', param = ker.width)
}
if(spdist=='graph.distance'){
  kergrid[[k]]=kerfn(newdata=coordinates(gridk[[k]])[,1:2],
               center=c(center[1],center[2],k), dist = spdist,
               ker.type = 'Gau', param = ker.width,
               distance.matrix =
               graph.distance.grid.centers)[which(assigng == k)]
}
}
vfitk[[k]]=vk.fit
}

# Store results
fpred[[b]]=matrix(NA,ngrid,2+(ker.width>0))
vfit[[b]]=matrix(NA,ngrid,1+2*nstr)
for(k in 1:K)
{
vfit[[b]][which(assigng==k),]=matrix(rep(c(vfitk[[k]][1,2],
                                           vfitk[[k]][-1,2],
                                           vfitk[[k]][-1,3]),
                                           sum(assigng==k)),
                                     nrow = sum(assigng==k),
                                     ncol = 1+2*nstr,byrow = T)
fpred[[b]][which(assigng==k),1:2]=as.matrix(slot(fpredk[[k]],'data'))
if(ker.width>0)
{
  fpred[[b]][which(assigng==k),3]=kergrid[[k]]
}
```

```
}

# There are some grid points where I could not compute the prediction,
# Here the function returns NA values
fpred[[b]][no.assg.grid,]=NA
vfit[[b]][no.assg.grid,]=NA

if(ker.width>0)
{
colnames(fpred[[b]])=c("V1","V1.var","Ker.val")
} else colnames(fpred[[b]])=c("V1","V1.var")

}
list.ret=list(fpred=fpred, variofit=vfit)
return(list.ret)

}

###############################################################
#####     PARALLEL adaptation of the Bootstrap step      ######
###############################################################

OKBVpar = function(data, K, grid, nk_min=1, B=100,
          model, sill_ini=NULL,
          range_ini=NULL, nugget_ini=NULL,
          spdist='euclidean',
          suppressMes=F, tol=1e12, ker.width=0,
          mesh = NULL, graph.distance= NULL,
          assign.matrix = NULL, no.assg.grid = NULL,
          data.grid.distance = NULL,
          is.observed = NULL,border.length=0)
{
colnames(data)=c('x','y','z')
nsub = dim(data)[1]
fpredk=fpred=gridk=vfitk=vfit=list()
nk=rep(0,K)
coord.list=as.list(data.frame(t(cbind(data$x, data$y))))
coordg.list=as.list(data.frame(t(cbind(grid$x, grid$y))))
nugget_ini_user=nugget_ini
sill_ini_user=sill_ini
range_ini_user=range_ini
coordinates(data)=c('x','y')
ngrid=dim(grid)[1]
nstr=length(model)

fpred = foreach(b=1:B, .combine='cbind',
        .packages=c('gstat', 'sp','spatstat','rgl',
                    'igraph','Cairo',
                    'fdaPDE','mgcv','geoR')) %dopar%
{
if(suppressMes==F)
```

```
print(paste("Repetition B = ", b, sep=''))
ind=sort(sample((border.length+1):nsub, size=K))
centers = cbind(data$x[ind],data$y[ind],ind)
assign = rep(0, nsub)
for(i in (border.length+1):nsub){
d = datapoints2centers.distance(x=cbind(data$x[i],data$y[i],i),
                            centers = centers, method = spdist,
                            graph.distance = graph.distance)
assign[i] = assign2center(d)}

while(min(table(assign))<nk_min)
{
if(suppressMes==F)
print("Repeating sampling")

ind=sort(sample((border.length+1):nsub, size=K))
centers = cbind(data$x[ind],data$y[ind],ind)
assign = rep(0, nsub)
for(i in (border.length+1):nsub){
d = datapoints2centers.distance(x=cbind(data$x[i],data$y[i],i),
                              centers = centers, method = spdist,
                              graph.distance = graph.distance)
assign[i] = assign2center(d)}
}
grid.new = cbind(grid, 1:ngrid, is.observed)
grid.new = data.frame(grid.new)
colnames(grid.new)=c('x','y','grid.position','is.observed')

assigng = rep(0, ngrid)
graph.distance.grid.centers = matrix(NA, K, ngrid)
triangles = mesh$T

for(i in 1:ngrid){
d = gridpoints2centers.distance(x=grid.new[i,], centers = centers,
                            method = spdist,
                            graph.distance = graph.distance,
                            assign.matrix = assign.matrix,
                            data.grid.distance =
                              data.grid.distance,
                            triangles = triangles)
graph.distance.grid.centers[,i]=d
assigng[i]=assign2center(d)
}
for(k in 1:K)
{ gridk[[k]]=grid[assigng==k,]
coordinates(gridk[[k]])=c('x','y')
}
if(ker.width>0)
{
v=variogram(z~1, data=data[which(!is.na(data$z)),], cloud=T)
vB=variogram(z~1, data=data[which(!is.na(data$z)),])
```

```
tmp=bbox(coordinates(data)[which(!is.na(data$z)),1:2])
cutoff=sqrt(diff(tmp[1,])^2+diff(tmp[2,])^2)/3
width=cutoff/(length(vB$dist))
punto_lag=cbind((0:(length(vB$gamma)-1))*width,
             (1:(length(vB$gamma)))*width)
kergrid=list()
}

for(k in 1:K)
{
datak=data.frame(data[assign==k,])[,1:3]
coordinates(datak)=c('x','y')
if(ker.width==0)
{
vk=variogram(z~1, data=datak)
}
if(ker.width>0)
{
vkB=vB
center = centers[k,]
KER=kerfn(newdata=coordinates(data)[,1:2],
         center=center, dist=spdist,
         ker.type = 'Gau',
         param = ker.width,
         distance.matrix = graph.distance)
KER=KER%*%t(KER)

for(l in 1:dim(punto_lag)[1])
{
inVh=which(v$dist>=punto_lag[l,1] & v$dist<punto_lag[l,2])
indKER=as.matrix(as.data.frame(v)[inVh,6:7])

vkB$gamma[l]=sum(v$gamma[inVh]*KER[indKER])/(sum(KER[indKER]))
}
vk=vkB
}
if(is.null(nugget_ini_user))
nugget_ini=.25*median(c(rep(vk$gamma[1],vk$np[1]),
                        rep(vk$gamma[2],vk$np[2])))

if(is.null(sill_ini_user))
sill_ini=median(c(rep(vk$gamma[length(vk$gamma)-1],
                      vk$np[length(vk$gamma)-1]),
                  rep(vk$gamma[length(vk$gamma)-2],
                      vk$np[length(vk$gamma)-2]),
                  rep(vk$gamma[length(vk$gamma)-3],
                      vk$np[length(vk$gamma)-3]),
                  rep(vk$gamma[length(vk$gamma)-4],
                      vk$np[length(vk$gamma)-4])))-nugget_ini
if(is.null(range_ini_user))
{
```

```
if(nstr>1)
{
print("Please provide initial range for variogram estimation")
return(-1)
}
if(model=='Exp')
{range_ini=vk$dist[min(which(vk$gamma>.95*
                          (sill_ini+nugget_ini)))]/3
}else range_ini=vk$dist[min(which(vk$gamma>
                          (sill_ini+nugget_ini)))]
}
vm=vgm(sill_ini,model[1],range_ini[1],nugget_ini[1])
nstr=length(model)
if(nstr>1)
{for(j in 2:nstr)
vm=vgm(psill = sill_ini, model = model[j],
     range = range_ini[j], add.to=vm)}
vk.fit = try(fit.variogram(vk, vm, fit.sills=T,
                       fit.method = 7,
                       debug.level = 1), silent = FALSE)
if(class(vk.fit)[1]=="try-error")
{
print("Trying fitting again")
vk.fit = fit.variogram(vk, vm, fit.sills=T,
                     fit.method = 2, debug.level = 0)
}
if(vk.fit$psill[1]<0 | vk.fit$range[2]>tol)
{
print("Trying with fixed nugget")
sill_ini=sill_ini+nugget_ini
nugget_ini=0
vm=vgm(sill_ini,model[1],range_ini[1],nugget_ini[1])
nstr=length(model)
if(nstr>1)
{for(j in 2:nstr)
vm=vgm(psill = sill_ini, model = model[j],
       range = range_ini[j], add.to=vm)}

vk.fit = try(fit.variogram(vk, vm,
                         fit.sills = c(0, rep(1, nstr)),
                         fit.method = 7, debug.level = 1))
if(class(vk.fit)[1]=="try-error")
{
print("Trying fitting again")
vk.fit = fit.variogram(vk, vm,
                       fit.sills = c(0, rep(1, nstr)),
                       fit.method = 2, debug.level = 1)
}
}

vk.fit$range[vk.fit$range<0]=range_ini[which(vk.fit$range<0)-1]
```

```
if(sum(vk.fit$psill)==0)
{
print('Warning: unable to fit the variogram, using the initial one')
vk.fit=vm
}

datak.gstat = gstat(formula = z ~ 1,
                data = datak, nmax = 50,
                model=vk.fit, set = list(gls=1))
fpredk[[k]] = predict(datak.gstat, gridk[[k]],debug.level=0)
if(ker.width>0)
{
if(spdist=='euclidean'){
kergrid[[k]]=kerfn(newdata=coordinates(gridk[[k]])[,1:2],
                    center=c(center[1],center[2],k),
                    dist = spdist, ker.type = 'Gau',
                    param = ker.width)
}
if(spdist=='graph.distance'){
kergrid[[k]]=kerfn(newdata=coordinates(gridk[[k]])[,1:2],
                    center=c(center[1],center[2],k),
                    dist = spdist, ker.type = 'Gau',
                    param = ker.width,
                    distance.matrix =
                   graph.distance.grid.centers)[which(assigng == k)]
}       }
vfitk[[k]]=vk.fit
}

# Store results
fpredb=matrix(0,ngrid,2+(ker.width>0))
vfitb=matrix(0,ngrid,1+2*nstr)
for(k in 1:K)
{
vfitb[which(assigng==k),]=matrix(rep(c(vfitk[[k]][1,2],
                           vfitk[[k]][-1,2],
                           vfitk[[k]][-1,3]),
                           sum(assigng==k)),nrow = sum(assigng==k),
                           ncol = 1+2*nstr,byrow = T)
fpredb[which(assigng==k),1:2]=as.matrix(slot(fpredk[[k]],'data'))
if(ker.width>0)
{
fpredb[which(assigng==k),3]=kergrid[[k]]
}

}
fpredb[no.assg.grid,]=NA
vfitb[no.assg.grid,]=NA

if(ker.width>0)
{
```

```
colnames(fpredb)=c("V1","V1.var","Ker.val")
} else colnames(fpredb)=c("V1","V1.var")
list(fpredb, vfitb)
}
list.ret=list(fpred=fpred[1,], variofit=fpred[2,])
return(list.ret)

}
```

A.5 Aggregation step

The results of the *B weak* analyses are aggregated into a final *strong* analysis by using the `aggrOKBV` function.

```
aggrOKBV=function(fOKBV, weight='equal', N_samples=NULL, vOKBV=NULL)
{
# INPUT:
# fOKBV = result of OKBV, as returned by
#         the first element of output of OKBV()
# weight = how the results are aggregated
#         (default : equal)
# N_samples = sample size
# vOKBV = variance of OK (needed only if variance weight are used)
# OUTPUT:
# Average prediction (on the grid used to produce fOKBV)

if(weight != 'equal' & weight!='median' &
   weight!='variance' & weight!='variovar' &
   weight!='pvariance' & weight!='kernel')
{
  print('Error: unexpected weight.
        Please choose betwen *equal*, *median*, *variance*,
        *variovar*, *pvariance* and *kernel*')
  return(-1)
}
fOKBV1=simplify2array(fOKBV)[,1,]
fOKBV2=simplify2array(fOKBV)[,2,]
if(weight=='equal')
  return(apply(fOKBV1,1, function(x)
    mean(x, na.rm = TRUE)))

if(weight=='median')
  return(apply(fOKBV1,1,median))

if(weight=='variance' & is.null(N_samples))
{
  print('Error: please provide N sample')
  return(-1)
}
B=dim(fOKBV1)[2]
if(weight=='variance')
{
  ngrid=dim(fOKBV1)[1]
  W.b=1/(apply(fOKBV2,2,sum)/(ngrid-N_samples))
  W.tot=sum(W.b)

  fpred.ave=apply(fOKBV1*matrix(rep(W.b/W.tot,ngrid),
                  nrow = ngrid,ncol=B,byrow = T),
                  1,sum)
  return(fpred.ave)
}

if(weight=='variovar' & is.null(vOKBV))
{
  print('Please provide estimated variograms')
  return(-1)
}

if(weight=='variovar')
{
  vOKBV=simplify2array(vOKBV)
  W.b=1/(apply(vOKBV[,2,],2,max))
  W.tot=sum(W.b)
  fpred.ave=apply(fOKBV1*matrix(rep(W.b/W.tot,ngrid),
                  nrow = ngrid,ncol=B,byrow = T),
                  1,sum)
}
if(weight=='pvariance')
{
  ngrid=dim(fOKBV1)[1]
  W.b=matrix(1,nrow = ngrid, ncol=B)
  W.b[which(fOKBV2>=1e-10)]=1/fOKBV2[which(fOKBV2>=1e-10)]
  W.tot=apply(W.b,1,sum)

  fpred.ave=apply(fOKBV1*W.b/W.tot, 1,sum)
  return(fpred.ave)
}
if(weight=='kernel')
{
  ngrid=dim(fOKBV1)[1]
  W.b=matrix(1,nrow = ngrid, ncol=B)
  W.b=simplify2array(fOKBV)[,3,]
  W.tot=apply(W.b,1,sum)

  fpred.ave=apply(fOKBV1*W.b/W.tot, 1,sum)
  return(fpred.ave)
}
return(-1)
}
```

A.6 Application to the Chesapeake Bay study

As mentioned in Chapter 6, due to the small size of the dataset with respect to the size of the Chesapeake Bay region, the graph-based analysis of this real case has been performed by computing a constrained Delaunay triangulation of the domain. For this reason some of the previously described functions have been partially modified. Here, we report the changes occurred to the `distance.on.graph` and `grid2triangle` functions. If the analysis is based on the graph metric, the `OKBV` function (or its parallelized adaptation) receives as input a `graph.distance` matrix which indicates the graph-based distance between each pair of vertices of the constrained Delaunay graph, instead of a $(nsub \cdot nsub)$-dimensional matrix only with the distances between sampled locations. The other difference is the input regarding the `data.grid.distance` matrix, which contains the graph-based distances between each grid point and each vertex of the mesh.

```
distance.on.graph.constrained = function(data, mesh)
{
# INPUT
# data = data matrix
# mesh = mesh obtained with a constrained
#        Delaunay triangulation
#        (with triangle function)
# OUTPUT
# graph.distance = n*n-dimensional symmetric matrix
#                  n = number of vertices
#                  (nsub + number of added nodes)
#                  (element (i,j) is the shortest length path
#                  between the i-th and the j-th vertices
#                  of the constrained Delaunay graph)
edge.list = mesh$E
edge.num = dim(edge.list)[1]
graph = graph_from_edgelist(edge.list, directed = FALSE)
edge.weights = rep(0, edge.num)

for(l in 1:edge.num){
  node.i = edge.list[l,1]
  node.j = edge.list[l,2]
  point.i = mesh$P[node.i,1:2]
  point.j = mesh$P[node.j,1:2]
  edge.weights[l] = dist(rbind(point.i,point.j),
                         method = 'euclidean')
}

graph.distance = distances(graph, v = V(graph), to = V(graph),
                           mode = c("all"),
                           weights = edge.weights,
                           algorithm = c("automatic"))
return(graph.distance)
}
```

```
grid2triangle.constrained = function(grid.matrix, data.matrix, mesh)
{
# INPUT:
# grid,matrix and data.matrix as before
# mesh = constrained Delaunay triangulation
# OUTPUT:
# as in the non-constrained case
triangles = mesh$T
num.triangles = dim(triangles)[1]

ngrid = dim(grid.matrix)[1]
assign.matrix = matrix(NA, ngrid, num.triangles)

for(j in 1:num.triangles){
  bnd.triangles = triangles[j,]
  assign.matrix[,j] = in.out(bnd = cbind(mesh$P[bnd.triangles,1],
                                         mesh$P[bnd.triangles,2]),
                        x = cbind(as.numeric(grid.matrix[,1]),
                                  as.numeric(grid.matrix[,2])))
}

check.complete = rep(NA, ngrid)
for(i in 1:ngrid){
  check.complete[i] = length(which(assign.matrix[i,]==TRUE))
}
no.assg.complete = which(check.complete == 0)

output = list()
output[[1]]=assign.matrix
output[[2]]=no.assg.complete

return(output)
}

#######################################################
#####          Dissolved Oxygen dataset           #####
#######################################################

# Upload the dataset
# data.file = nsub*3-dimensional matrix with
#             long-coordinate
#             lat -coordinate
#             DO(x,y) (or NA if boundary point)
data.file=read.table(file='Data.txt', header=TRUE)
data.file=data.frame(data.file)
colnames(data.file)=c('x','y','z')
zlim = range(data.file[,3], na.rm = T)

# Upload the grid
grid=read.table(file='Grid2Predict.txt', header=TRUE)
grid=data.frame(grid)
colnames(grid)=c('x','y')
```

```
X=as.vector(as.matrix(read.table('X.txt', header = T)))
Y=as.vector(as.matrix(read.table('Y.txt', header = T)))
lenX=length(X)
lenY=length(Y)
ngrid=dim(coordinates(grid))[1]

# Matrix with information regarding which
# grid points are inside the shape
in.poly.matrix = as.matrix(read.table('InPolyMatrix.txt'))

shape = readShapePoly('estuaries.shp')
shape = gUnaryUnion(shape1)

# Boundary points (which z = NA)
border.index = which(is.na(data.file$z))
border.coordinates = data.file[border.index,1:2]
border.length = length(border.index)

nsub=dim(data.file)[1]

##########    Parameter Settings    ############
B = 100
rangeK=2^(0:5)
ker.width = 0.5
aggregation.type = 'equal'
spdist.vector = c('euclidean','graph.distance')
model=c('Exp')
################################################

nstr=length(model)
nk_min=1

sill_ini=NULL; range_ini=NULL; nugget_ini=NULL;

data=data.file
data = data.frame(data)
colnames(data) = c('x','y','z')
N_samples=dim(data)[1]
n=dim(data)[2]-2

# Constrained Dekaunay Traingulation
segment.bnd = matrix(0, border.length, 2)
for(i in 1:(border.length-1))
{
segment.bnd[i,1]=i
segment.bnd[i,2]=i+1
}
segment.bnd[border.length,1]=border.length
segment.bnd[border.length,2]=1

planar.slg = pslg(P = data[,1:2], S = segment.bnd)
```

```
mesh = RTriangle::triangulate(planar.slg, a = 0.01)

graph.distance.complete = distance.on.graph.constrained(data,
                               mesh = mesh)
output.grid2triangle = grid2triangle.constrained(grid.matrix = grid,
    data.matrix = cbind(data$x, data$y), mesh = mesh)
assign.matrix = output.grid2triangle[[1]]
no.assg.grid = output.grid2triangle[[2]]
data.grid.distance = loccoords(coords = cbind(data$x,data$y),
             locations = grid)
already.observed = which(data.grid.distance == 0, arr.ind = TRUE)
is.observed = rep(0,ngrid)
is.observed[already.observed[,2]]=already.observed[,1]
vertices.grid.distances = loccoords(coords = cbind(mesh$P[,1],mesh$P[,2]),
                  locations = grid)

M= 100
parallel=TRUE
if (parallel)
worker.number<-2


for(spdist in spdist.vector){
resOKBV=fpred=f.ave=ds.pred=vfit=list()
ds.pred.Spatial = list()

for(K in rangeK)
{
print(K)
if(K==1) # No tessellation
{
resOKBV[[K]]=OKBV(data, K, grid, nk_min, B=1, model,
      sill_ini=sill_ini, suppressMes=T,
      range_ini, nugget_ini,
      spdist='euclidean', ker.width = 0,
      no.assg.grid = no.assg.grid,
      data.grid.distance = vertices.grid.distances,
      is.observed = is.observed,
      border.length=border.length)
fpred[[K]]=simplify2array(resOKBV[[K]]$fpred)[,,1]
vfit[[K]]=simplify2array(resOKBV[[K]]$variofit)[,,1]
ds.pred[[K]]=in.poly.matrix
ds.pred[[K]][which(!is.na(in.poly.matrix))]=fpred[[K]][,1]
x11(width=8, height=8)
image.plot(X,Y,ds.pred[[which(rangeK==K)]],col = tim.colors(M),
main=paste('Ordinary Kriging - K=',K,sep=''), zlim = zlim ,asp=1)
plot(shape, add=T)

}
if(K>1)
{
```

```
# if parallel implementation is used
if(parallel)
{
cl <- makeCluster(worker.number)
registerDoParallel(cl)
resOKBV[[which(rangeK==K)]]=OKBVpar(data, K, grid, nk_min, B=B,
                          model, sill_ini=NULL, suppressMes=F,
                          range_ini=NULL, nugget_ini=NULL,
                          spdist=spdist, ker.width = ker.width,
                          mesh = mesh, graph.distance.complete =
                            graph.distance.complete,
                          assign.matrix = assign.matrix,
                          no.assg.grid = no.assg.grid,
                          data.grid.distance = vertices.grid.distances,
                          is.observed = is.observed,
                          border.length=border.length)
stopCluster(cl)
}
if (!parallel)
{
resOKBV[[which(rangeK==K)]]=OKBV(data, K, grid, nk_min, B=B, model,
                       sill_ini=NULL, suppressMes=F,
                       range_ini=NULL, nugget_ini=NULL,
                       spdist=spdist, ker.width = ker.width,
                       mesh = mesh,
                       graph.distance.complete =
                         graph.distance.complete,
                       assign.matrix = assign.matrix,
                       no.assg.grid = no.assg.grid,
                       data.grid.distance = vertices.grid.distances,
                       is.observed = is.observed,
                       border.length=border.length)
}

f.ave[[which(rangeK==K)]]=aggrOKBV(resOKBV[[which(rangeK==K)]]$fpred,
                       weight=aggregation.type,
                       N_samples,
                       resOKBV[[which(rangeK==K)]]$variofit )

ds.pred[[which(rangeK==K)]]=in.poly.matrix
ds.pred[[which(rangeK==K)]][which(!is.na(in.poly.matrix))]=
f.ave[[which(rangeK==K)]]
x11(width=8, height=8)
image.plot(X,Y,ds.pred[[which(rangeK==K)]],col = tim.colors(M),
main=paste('KrigRDD - K=',K,sep=''), zlim = zlim , asp=1)
plot(shape, add=T)
}
}

}
```

Bibliography

L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

J. P. Chilès and P. Delfiner. *Geostatistics: Modeling Spatial Uncertainty.* John Wiley & Sons, New York, 1999.

A. Cohen and R. H. Jones. Regression on a random field. *Journal of the American Statistical Association*, 64:1172–1182, 1969.

N. Cressie. *Statistics for Spatial Data.* John Wiley & Sons, New York, 1993.

G. Csardi and T. Nepusz. *The igraph software package for complex network research*, 2006.

F. C. Curriero. On the use of non-Euclidean distance measures in geostatistics. *Mathematical Geology*, 38:907–926, 2006.

M. de Berg, M. van Kreveld, M. Overmars, and O. C. Schwarzkopf. *Delaunay Triangulations*, pages 183–210. Springer Berlin Heidelberg, 2000a.

M. de Berg, M. van Kreveld, M. Overmars, and O. C. Schwarzkopf. *Polygon Triangulation*, pages 45–61. Springer Berlin Heidelberg, 2000b.

B. Delaunay. Sur la sphère vide. *Izvestia Akademii Nauk SSSR*, 1934.

E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

F. Fouedjio. Second-order non-stationary modeling approaches for univariate geostatistical data. *Stochastic Environmental Research and Risk Assessment*, pages 1–20, 2016.

F. Fouedjio, N. Desassis, and J. Rivoirard. A generalized convolution model and estimation for non-stationary random functions. *Spatial Statistics*, 16:35 – 52, 2016.

M. Fuentes. A high frequency kriging approach for non-stationary environmental processes. *Environmetrics*, 12:469–483, 2001.

M. Fuentes. Interpolation of nonstationary air pollution processes: a spatial spectral approach. *Statistical Modelling*, 2:281–298, 2002.

P. Guttorp and P. D. Sampson. Methods for estimating heterogeneous spatial covariance functions with environmental applications. *Handbook of Statistics*, 12:661–689, 1994.

T. C. Haas. Kriging and automated variogram modeling within a moving window. *Atmospheric Environment. Part A. General Topics*, 24:1759 – 1769, 1990.

T. C. Haas. Local prediction of a spatio-temporal process with an application to wet sulfate deposition. *Journal of the American Statistical Association*, 90: 1189–1199, 1995.

P. Harris, M. Charlton, and A. S. Fotheringham. Moving window kriging with geographically weighted variograms. *Stochastic Environmental Research and Risk Assessment*, 24:1193–1209, 2010.

M. J. Heaton, W. F. Christensen, and M. A. Terres. Nonstationary gaussian process models using spatial hierarchical clustering from finite differences. *Technometrics*, 2015.

D. Higdon. A process-convolution approach to modelling temperatures in the North Atlantic Ocean. *Environmental and Ecological Statistics*, 5:173–190, 1998.

D. Higdon, J. Swall, and J. Kern. Non-stationary spatial modeling. *Bayesian statistics*, 6:761–768, 1999.

O. P. Jensen, M. C. Christman, and T. J. Miller. Landscape-based geostatistics: a case study of the distribution of blue crab in chesapeake bay. *Environmetrics*, 17:605–621, 2006.

H. M. Kim, B. K. Mallick, and C. C. Holmes. Analyzing nonstationary spatial data using piecewise gaussian processes. *Journal of the American Statistical Association*, 100:653–668, 2005.

K. Krivoruchko and A. Gribov. Geostatistical interpolation and simulation with non-Euclidean distances. In *geoENV IV*, pages 331–342. Kluwer Academic Publishers, 2002.

J. Lin, C. Chen, and J. Wu. Cd-graph: planar graph representation for spatial adjacency and neighbourhood relation with constraints. *International Journal of Geographical Information Science*, 27:1902–1923, 2013.

L. S. Little, D. E., and D. E. Porter. Kriging in estuaries: as the crow flies, or as the fish swims? *Journal of Experimental Marine Biology and Ecology*, 213:1 – 11, 1997.

A. Løland and G. Høst. Spatial covariance modelling in a complex coastal domain by multidimensional scaling. *Environmetrics*, 14:307–321, 2003.

J. S. Marron and A. M. Alonso. Overview of object oriented data analysis. *Biometrical Journal*, 56:732–753, 2014.

G. Matheron. *The Theory of Regionalized Variables and Its Applications.* Centre de Morphologie Mathématique Fontainebleau: Les cahiers du Centre de Morphologie Mathématique de Fontainebleau. École national supérieure des mines, 1971.

A. Menafoglio and P. Secchi. Statistical analysis of complex and spatially dependent data: a review of object oriented spatial statistics. *European Journal of Operational Research, in press.*, 2016.

A. Menafoglio, P. Secchi, and M. Dalla Rosa. A Universal Kriging predictor for spatially dependent functional data of a Hilbert Space. *Electronic Journal of Statistics*, 7:2209–2240, 2013.

A. Menafoglio, A. Guadagnini, and P. Secchi. Stochastic simulation of soil particle-size curves in heterogeneous aquifer systems through a Bayes space approach. *Water Resources Research*, 52:5708–5726, 2016.

C. L. Newcombe and W. A. Horne. Oxygen-poor waters of the Cesapeake Bay. *Science*, 88:80–81, 1938.

D. Nychka, C. Wikle, and J. A. Royle. Multiresolution models for nonstationary spatial covariance functions. *Statistical Modelling*, 2:315–331, 2002.

M. D. Penrose. Laws of large numbers in stochastic geometry with statistical applications. *Bernoulli*, 13:1124–1150, 2007.

M. B. K. Prasad, W. Long, X. Zhang, R. J. Wood, and R. Murtugudde. Predicting dissolved oxygen in the Chesapeake Bay: applications and implications. *Aquatic Sciences*, 73:437–451, 2011.

R Development Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2008.

T. Ramsay. Spline smoothing over difficult regions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64:307–3019, 2002.

S. L. Rathbun. Spatial modelling in irregularly shaped regions: kriging estuaries. *Environmetrics*, 9:109–129, 1998.

D. S. Richards. Positive definite symmetric functions on finite-dimensional spaces II. *Statistics and Probability Letters*, 3:325–329, 1985.

P. D. Sampson and P. Guttorp. Nonparametric estimation of nonstationary spatial covariance structure. *Journal of the American Statistical Association*, 87:108–119, 1992.

L. M. Sangalli, J. O. Ramsay, and T. O. Ramsay. Spatial spline regression models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75: 681–703, 2013.

I. J. Schoenberg. On certain metric spaces arising from euclidean spaces by a change of metric and their imbedding in Hilbert space. *Annals of Mathematics*, 38:787–793, 1937.

M. E. Scully. Wind modulation of dissolved oxygen in Chesapeake Bay. *Estuaries and Coasts*, 33:1164–1175, 2010.

P. Secchi, S. Vantini, and V. Vitelli. Bagging voronoi classifiers for clustering spatial functional data. *International Journal of Applied Earth Observation and Geoinformation*, 22:53–64, 2013.

P. Secchi, S. Vantini, and V. Vitelli. Analysis of spatio-temporal mobile phone data: a case study in the metropolitan area of Milan. *Statistical Methods & Applications*, 24:279–300, 2015.

J. R. Shewchuk, D. C. Sterratt, and E. Pipping. *RTriangle: Triangle - A 2D Quality Mesh Generator and Delaunay Triangulator*, 2016. R package version 1.6-0.8.

S. Tavakoli, D. Pigoli, and J. A. D. Aston. Spatial modeling of object data: Analysing dialect sound variations across the UK. Technical report, arXiv:1610.10040, 2016. URL `https://arxiv.org/pdf/1610.10040v1.pdf`.

H. Wang and J. S. Marron. Object oriented data analysis: Sets of trees. *The Annals of Statistics*, 35:1849–1873, 2007.

J. Wells and L. Williams. *Embeddings and extensions in analysis*. Ergebnisse der Mathematik und ihrer Grenzgebiete. Springer-Verlag, 1975.

S. N. Wood, M. V. Bravington, and S. L. Hedley. Soap film smoothing. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70:931–955, 2008.

Y. Xu, J. S. Dyer, and A. B. Owen. Empirical stationary correlations for semi-supervised learning on graphs. *The Annals of Applied Statistics*, 4:589–614, 2010.

G. Young and A. S. Householder. Discussion of a set of points in terms of their mutual distances. *Psychometrika*, 3:19–22, 1938.

H. Zhang and Y. Wang. Kriging and cross-validation for massive spatial data. *Environmetrics*, 21:290–304, 2010.