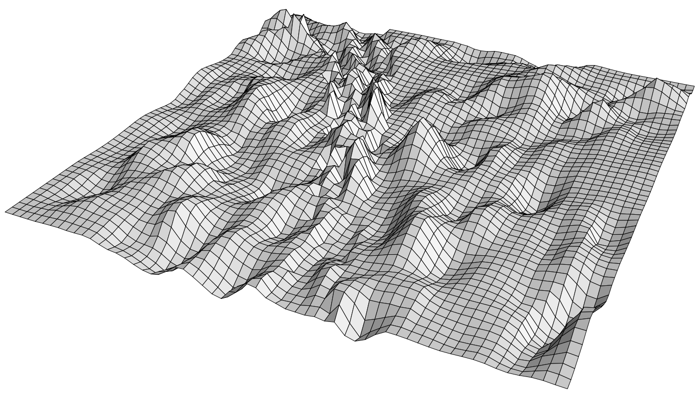
Freiberg Online Geoscience
Volume 6

Geostatistics without Stationarity Assumptions within Geographical Information Systems

Alexander Brenning



Diplomarbeit zur Erlangung des akademischen Grades „Diplom-Mathematiker“

1. Gutachter: Prof. Dr. Helmut Schaeben
2. Gutachter: Prof. Dr. Wolfgang Näther

Contents

Abstract

The present work deals with two challenging problems of applied geostatistics: (i) Stationarity assumptions often do not hold under real-world conditions. (ii) Geostatistical methods have to be linked with spatial databases in order to be applicable in non-stationary situations. Solutions for both problems are proposed and implemented.

(i) A central assumption in geostatistics is the stationarity of the process. However the spatial variability of many natural phenomena heavily depends on the local geology, which is non-stationary in most cases. To deal with this, the concept of process stationarity is replaced by a stationarity of the governing influence relating the local semivariogram and the local geology as stored in a Geographical Information System (GIS). A construction method is used, which can meaningfully incorporate additional spatial information from GIS, e.g. smoothly varying geology in the investigated area, spatially varying anisotropy induced by mountainous morphology, or geological faults interrupting continuity. Least-squares parameter estimation is used for fitting instationary semivariogram models in typical example situations, leading to non-linear optimization problems. Furthermore, a method for semivariogram parameter estimation in the present of linear trend is proposed.

(ii) Geostatistical tools that make use of the local geology need direct access to the data stored in the GIS. A link between the presented geostatistical tools and the GIS software ArcView was established. Thus, spatial data such as measured contaminant concentrations, soil properties and morphology can be incorporated in geostatistical analyses.

R code that fits instationary semivariogram models and performs kriging was implemented and can be obtained from the author[1]. It is applied to simulated datasets.

Zusammenfassung

Die vorliegende Diplomarbeit befasst sich mit zwei wichtigen Problemen der angewandten Geostatistik: (i) Stationaritätsannahmen werden unter realweltlichen Bedingungen oft nicht erfüllt. (ii) Geostatistische Methoden müssen mit räumlichen Datenbanken verbunden werden, um unter nichtstationären Bedingungen anwendbar zu sein. Lösungen für beide Probleme werden vorgeschlagen und implementiert.

(i) In der Geostatistik ist die Stationarität des Prozesses eine zentrale Annahme. Die räumlich Variabilität vieler Phänomene in unserer Umwelt hängt jedoch stark von lokalen geologischen Verhältnissen ab, die meist aber instationär sind. Um damit umgehen zu können, wird das Konzept der Stationarität des Prozesses ersetzt durch eine Stationarität des Einflusses der lokalen Geologie, wie sie in einem GIS gespeichert ist, auf das lokale Semivariogramm. Es wird eine Konstruktionsmethode benutzt, die auf sinnvolle Art räumliche Informationen aus dem GIS in Semivariogrammmodelle einbinden kann, etwa sich über das Untersuchungsgebiet gleichmäßig verändernde geologische Verhältnisse, sich räumlich verändernde Anisotropie im Gebirgsrelief oder geologische Störungen, die die Kontinuität unterbrechen. Kleinste-Quadrate Schätzung wird für die Anpassung instationärer Semivariogrammmodelle in typischen Beispielsituationen verwendet. Dies führt zu nichtlinearen Optimierungsproblemen. Des weiteren wird eine Methode der Schätzung von Semivariogrammparametern in Modellen mit linearem Trend vorgestellt.

(ii) Geostatistische Werkzeuge, die lokalen geologischen Verhältnisse berücksichtigen, benötigen einen direkten Zugang zu Daten, die in einem GIS gespeichert sind. Im Rahmen dieser Arbeit wurde eine Verbindung zwischen den vorgestellten geostatistischen Werkzeugen und dem GIS-Programm ArcView erstellt. Auf diese Weise können räumliche Daten wie etwa Schadstoffkon-

[1]Current address: Universität Erlangen–Nürnberg, Institut für Geographie, Kochstr. 4/4, D–91054 Erlangen; e-mail: `ali@proforma.de`.

zentrationen, Bodeneigenschaften oder die Morphologie in geostatistische Analysen einbezogen werden.

R-Code, der instationäre Semivariogrammmodelle anpasst und Kriging durchführt, wurde erstellt und auf simulierte Datensätze angewandt. Der Code kann über den Author[2] bezogen werden.

[2]Gegenwärtige Anschrift: Universität Erlangen–Nürnberg, Institut für Geographie, Kochstr. 4/4, D–91054 Erlangen; e-mail: `ali@proforma.de`.

Preface

The present work shows how geostatistical models and methods can be adapted in order to facilitate the integration of geostatistical data analysis and Geographical Information Systems (GIS) and make better use of the information stored within the latter. The main focus is on models that represent local anisotropies given by geological covariables (Section 2.2) and on those with linear trend (Section 2.6). Both models and respective fitting methods will be implemented within the data analysis language R and linked with the GIS ArcView (Sections 3.3, 3.4).

The dataset of humidity indices from the Ecuadorian Andes (Prof. Dr. Michael Richter, Erlangen) that I intended to study in Chapter 4 will only appear as an example for a sample session; it is not presented in detail because of the great deal of work that would have been necessary for completing the covariable data using a digital elevation model and for doing comprehensive exploratory data analysis.

I wish to thank Prof. Dr. Helmut Schaeben and Dipl.-Math. Gerald van den Boogaart for supervising my work and supporting its interdisciplinary scope. Furthermore thanks to Prof. Dr. Wolfgang Näther for being the second reviewer of this Diploma thesis.

This work was only possible within a multidisciplinary environment including the fields of mathematics, geosciences and geocomputation. I am glad that I have found such an environment at the Freiberg University of Technology and Mining.

Freiberg, June 2001

Alexander Brenning

Chapter 1

Introduction

1.1 Why Use Geostatistical Methods within GIS?

Over thousands of years, the physical support for spatial information has been evolving, shifting from wood and stone to paper, which made more efficient production and reproduction and accurate representation of information possible. In the 20th century, technological innovation has lead to a breath-taking acceleration of spatial data acquisition and has simultaneously created the tools that are necessary for efficiently managing great amounts of spatial information, namely information systems and, in this particular case, Geographical Information Systems (GIS). These are instruments that enable us to store, modify and extract spatial data, as well as to visualize and analyze it.

In connection with local networks and the world-wide web, spatial data can now be made virtually omnipresent. Currently, GIS are more and more used in environmental planning, agronomy, hydrology, logistics and many other scientific and economic fields. At the same time, the amount of spatial information that is stored and processed within GIS still increases rapidly because of the wide use of data derived from Remote Sensing and Global Positioning Systems (Kraas 1993, Longley et al. 1999).

These developments make it possible and necessary to apply modern data analysis techniques to a much larger extent than in paper-based times, including the application of statistical methods for modeling spatial processes and distributions. There is still a need for adapting these techniques and integrating them into GIS. The present work contributes to the solution of this problem from a geostatistical point of departure, Geostatistics being understood as the mathematical discipline that studies stochastic processes with continuous spatial indices in two or more dimensions (Cressie 1993).

1.2 Why Drop Stationarity Assumptions?

Precisely the above described development of spatial information processing creates a need for improving geostatistical techniques. The existence of detailed thematic data now permits the application of more sophisticated and complex models that also reflect an improved understanding of our physical environment and, if necessary, make use of the computer power available today.

How can these models look like, in the case of Geostatistics?

For example, consider mineral concentrations measured within an ore deposit that was created through hydrothermal alteration, i. e. the intrusion of hot solutions into rock due to deep magmatism. Suppose an underlying pattern of tectonic faults that shows a preferred orientation, say North–South, and that it already existed prior to the intrusion of hot solutions. As a consequence, the mineral concentrations found today at two different points can be assumed to show higher dependencies (or correlations) if they are aligned North–South than in an East–West configuration. This direction-dependency of correlations, called anisotropy, can be modeled in some special cases without great difficulty, e. g. if there is only one global direction of anisotropy, as in the example above. However, more complex patterns of orientations may occur, for instance due to foldings or depending on mountainous relief.

Another problematic situation occurs when a linear trend is present in the data. Until now, mathematically unsatisfactory strategies have been applied to model fitting under these conditions (see Section 2.6).

Both anisotropy and presence of trend imply instationarity of the underlying stochastic process, and solutions to both problems of variogram parameter estimation were proposed by van den Boogaart (1999) and van den Boogaart (2000) and are studied and applied in this work.

Chapter 2

Some Geostatistical Theory

In the present chapter the reader is presumed to be familiar with the basic notions and results of probability theory (see e. g. Bauer (1978), Billingsley (1995)). An introduction to geostatistical theory will be given, including the presentation and application of the results obtained by van den Boogaart (1999) and van den Boogaart (2000) and of some examples that will reappear in the rest of this work.

2.1 Second-order Stochastic Processes

2.1.1 Introduction

Let $(\Omega, \mathcal{A}, P)$ be a probability space and let $(\mathbb{R}^d, \mathcal{B}^d, \lambda)$ denote the Lebesgue-Borel measure space and $\|\cdot\|$ the Euclidian norm on $\mathbb{R}^d$, $d \geq 1$. Furthermore, let D be an arbitrary non-empty set.

Definition 2.1.1 A *real-valued random variable* or *random function* on $(\Omega, \mathcal{A}, P)$ is an $\mathcal{A}$-$\mathcal{B}^1$-measurable function $Z : \Omega \to \mathbb{R}$. A *real-valued stochastic process* (or just *process*) with the parameter set D is a collection $Z = (Z_t)_{t \in D}$ of real-valued random variables on $(\Omega, \mathcal{A}, P)$. If $D \in \mathcal{B}^2$, Z is also called a *random field.* A process Z is *of second order*, if every random variable Z_t, $t \in D$, is square integrable.

In continuation we consider real-valued random variables and processes only. If a random variable is integrable,

$$E(Z) = \int_\Omega Z \, \mathrm{d}P$$

denotes its *expected value* or *mean value.*

Remark and Definition 2.1.2 Let Z be a second-order process. Then the expected value $E(Z_t)$ and the *covariance*

$$\operatorname{Cov}(Z_s, Z_t) = E((Z_s - E(Z_s)) \cdot (Z_t - E(Z_t)))$$

exist for all $s, t \in D$.

The mapping

$$m : D \to \mathbb{R}, \quad m(t) := E(Z_t)$$

is called the *expected value* or *mean* of Z, and

$$C : D \times D \to \mathbb{R}, \quad C(s,t) := \operatorname{Cov}(Z_s, Z_t)$$

the *covariance function* of Z. In geostatistics, the latter is commonly called the *covariogram* and m the *trend* or *drift* of Z.

Definition 2.1.3 A process Z is called a *Gaussian process*, if for all $n \in \mathbb{N}$ and all $t_1, \dots, t_n \in D$ it holds: The joint distribution of $Z_{t_1}, \dots, Z_{t_n}$ is an n-dimensional normal distribution.

Remark 2.1.4 Every Gaussian process is a second-oder stochastic process, because every normal random variable is square integrable.

Question 2.1.5 For which functions $m : D \to \mathbb{R}$ and $C : D^2 \to \mathbb{R}$ exists a second-oder process with expected value m and covariance function C?

Definition 2.1.6 A function $F : D \times D \to \mathbb{R}$ is *positive semidefinite*, if

$$\forall\, n \in \mathbb{N}\ \forall\, t_1, \dots, t_n \in D\ \forall\, a_1, \dots, a_n \in \mathbb{R} : \quad \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j F(t_i, t_j) \geq 0.$$

Theorem 2.1.7 Consider two functions $m : D \to \mathbb{R}$ and $C : D^2 \to \mathbb{R}$. Then the following conditions are equivalent:

i) C is positive semidefinite and symmetric.

ii) There exist a probability space and a second-order process defined on it that has expected value m and covariance function C.

Proof: ii) $\Rightarrow$ i):

Let $(Z_t)_{t \in D}$ be an arbitrary stochastic process on D with expected value m and covariance function C. C is symmetric because it holds

$$C(s,t) \;=\; \mathrm{E}((Z_s - m(s)) \cdot (Z_t - m(t))) \;=\; \mathrm{E}((Z_t - m(t)) \cdot (Z_s - m(s))) \;=\; C(t,s).$$

Furthermore, we have

$$\forall\, n \in \mathbb{N} \quad \forall\, t_1, \dots, t_n \in D \quad \forall\, a_1, \dots, a_n \in \mathbb{R} :$$

$$\sum_{i,j=1}^{n} a_i a_j C(t_i, t_j) \;=\; \sum_{i,j=1}^{n} a_i a_j \mathrm{E}((Z_{t_i} - m(t_i)) \cdot (Z_{t_j} - m(t_j))) \;=$$

$$=\; \mathrm{E}\Big(\sum_{i,j=1}^{n} a_i a_j (Z_{t_i} - m(t_i)) \cdot (Z_{t_j} - m(t_j)) \Big) \;=\; \mathrm{E}\Big(\sum_{i=1}^{n} a_i (Z_{t_i} - m(t_i)) \Big)^2 \;\geq\; 0.$$

i) $\Rightarrow$ ii): Draft of the proof:

We will construct a collection of finite-dimensional Gaussian processes. Their existence implies the existence of of a Gaussian process on D, due to Colmogorov's theorem.

Let $C : D \times D \to \mathbb{R}$ be positive semidefinite and $m : D \to \mathbb{R}$ an arbitrary function. Furthermore, let $\mathcal{H}(D)$ denote the collection of all finite subsets of D, and for $I \subset J \subset D$, let

$$\pi_I^J : \mathbb{R}^J \to \mathbb{R}^I, \quad \tau \mapsto \tau|_I$$

be the restriction mapping from J to I. For every $J \in \mathcal{H}(D)$ we define a measuring space $(\Omega_J, \mathcal{A}_J) := (\mathbb{R}^J, (\mathcal{B}^1)^J)$. To $(\Omega_J, \mathcal{A}_J)$, we put the normal distribution P_J with expected value 0 and covariance matrix $C|_{J \times J}$ as probability measure. (Note that P_J is well-defined, because $C|_{J \times J}$ is positive semidefinite.)

It can be shown that for all $I, J \in \mathcal{H}(D)$ with $I \subset J$, it holds

$$\pi_I^J P_J \;=\; P_I.$$

A collection $(P_J)_{J\in\mathcal{H}(D)}$ of probability measures with this property is called *projective*.

Colmogorov's Theorem (Bauer 1978) guarantees the existence of one unique probability measure P on[1] $(\Omega,\mathcal{A}) = (\mathbb{R}^D,\mathcal{B}^D)$ satisfying

$$\forall\, I \in \mathcal{H}(D): \qquad \pi_I^D P \;=\; P_I.$$

The probability measure P is called the *projective limit* of $(P_I)_{I\in\mathcal{H}(D)}$.

If for all $t \in D$ we define random variables $Z_t = \omega(t) + m(t)$ on the probability space $(\Omega,\mathcal{A},P)$, then we have a second-order process on D with mean m and covariance function C. It is, by construction, a Gaussian process. □

Remark 2.1.8 As a consequence of Theorem 2.1.7, a symmetric positive semidefinite function $C : D^2 \to \mathbb{R}$ is often called a *covariance function*, even if a corresponding second-order process has not been introduced explicitely.

Remark 2.1.9 Let C, D be positive semidefinite functions and $\lambda \geq 0$. Then $C + D$, λC and $C + \lambda$ are also positive semidefinite.

2.1.2 Semivariograms and Semivariogram Models

In contrast to time series analysis, where the autocorrelation function is the most important object that is studied, in Geostatistics the so-called semivariogram is usually preferred to the covariance function.

Throughout the rest of this work, we always consider processes of second order with parameter sets $D \subset \mathbb{R}^d$.

Definition 2.1.10 For a stochastic process Z on D, the function

$$\gamma : D\times D \to \mathbb{R}, \qquad \gamma(s,t) := \tfrac{1}{2}\mathrm{Var}(Z(s) - Z(t))$$

is well-defined and is called the *semivariogram* of the process Z, 2γ its *variogram*.

Remark 2.1.11 If Z is a stochastic process with covariance funtion C and semivariogram γ, then it holds:

$$2\gamma(s,t) = C(s,s) + C(t,t) - 2C(s,t). \tag{2.1}$$

Example 2.1.12 i) *Spherical semivariogram*: For $\sigma^2 \geq 0$ and $a > 0$, the function $\gamma : D\times D \to \mathbb{R}$ defined by

$$\gamma^{\mathrm{sph}}(s,t) = \begin{cases} \sigma^2\left(\frac{3}{2}\frac{\|t-s\|}{a} - \frac{1}{2}\frac{\|t-s\|^3}{a^3}\right) & \text{if } \|t-s\| < a, \\ \sigma^2 & \text{otherwise} \end{cases} \tag{2.2}$$

is the semivariogram of a stochastic process on $D \subset \mathbb{R}^d$, $d = 1, 2, 3$. (This will be shown in Example 2.2.9.)

The spherical semivariogram as a function of $h = t - s \in \mathbb{R}^d$ has only one continuous derivative at $h \in \partial B^d(0,a)$ and is continuous but not differentiable at $0 \in \mathbb{R}^d$, $d > 1$.

ii) *Exponential semivariogram*: For $\sigma^2 \geq 0$ and $a > 0$,

$$\gamma^{\exp}(s,t) = \begin{cases} \sigma^2\exp(-\|t-s\|/a) & \text{if } \|t-s\| > 0, \\ 0 & \text{if } \|t-s\| = 0, \end{cases}$$

[1]Remark: The product σ-algebra $\mathcal{B}^D := \bigotimes_{t\in D}\mathcal{B}$ of $\mathcal{B}$ is defined to be the smallest σ-algebra in $\mathbb{R}^D$, with respect to which all projections $\pi^D_{\{t\}}$ are $\mathcal{B}^D$-$\mathcal{B}$-measurable (Bauer 1978).

defines the exponential semivariogram of a stochastic process on $D \subset \mathbb{R}^d$, $d \geq 1$. At 0 it is continuous, but not differentiable. The exponential semivariogram does not reach a maximum; it converges to σ^2 for $\|t - s\| \to \infty$.

iii) *Nugget effect*: Sometimes it is desired to take into account measuring errors or microscale variability of the measurements. This can be achieved by adding a so-called nugget effect semivariogram

$$\gamma^{\text{nug}}(s,t) = \begin{cases} 0 & \text{if } s = t, \\ \sigma^2 & \text{if } s \neq t \end{cases}$$

to a given semivariogram.

Further semivariograms are presented by Cressie (1993) and Stein (1999), for example.

Remark 2.1.13 i) Semivariograms are symmetric and *conditionally negative semidefinite*, in the sense that for all $n \in \mathbb{N}$, for all $s_1, \ldots, s_n \in D$ and for all $a_1, \ldots, a_n \in \mathbb{R}$ with $\sum_i a_i = 0$, it holds

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j \gamma(s_i, s_j) \leq 0.$$

(Proof: Use (2.1), the condition on the a_is and the positive semidefiniteness of C.)

ii) In general it is difficult to determine whether a conditionally negative semidefinite function $\gamma : D \times D \to \mathbb{R}$ is the semivariogram of a second-order process (Cressie 1993, pp. 86–90). Hence it is important to keep in mind that, when talking about semivariograms, we make the non-trivial assumption that there exists a corresponding stochastic process.

However, as a consequence of Theorem 2.1.7, it is a safe method to derive semivariograms from covariograms using equation (2.1).

Remark 2.1.14 i) Different covariograms may yield the same semivariogram: Suppose that X and Y are random fields with covariograms C_X and $C_Y = C_X + \Delta$, $\Delta > 0$, respectively. ($C_X + \Delta$ is positive semidefinite, see Remark 2.1.9.) Thus, using (2.1) we get

$$2\gamma_Y(s,t) = C_X(s,s) + \Delta + C_X(t,t) + \Delta - 2(C_X(s,t) + \Delta) = 2\gamma_X(s,t).$$

ii) Let C, $\widetilde{C}$ be covariograms and γ_C, $\gamma_{\widetilde{C}}$ the corresponding semivariograms. Then $\gamma_C + \gamma_{\widetilde{C}}$ is the semivariogram corresponding to $C + \widetilde{C}$.

Definition 2.1.15 Let Θ be an arbitrary non-empty set. Suppose that for every $\theta \in \Theta$, $\gamma_\theta : D \times D \to \mathbb{R}$ is a semivariogram (of a suitable process on D). Then the function $\gamma : D \times D \times \Theta \to \mathbb{R}$ is called a *semivariogram model*. We also denote it as $\gamma = (\gamma_\theta)_{\theta \in \Theta}$. Θ is called the *parameter set* of γ.

2.1.3 (In-) Stationarity and (An-) Isotropy

Stationary and isotropic processes have second-order structures that are in certain sense invariant in space. This makes life much easier in mathematics, but in practice these properties generally cannot be guaranteed. Nevertheless, stationary and isotropic processes constitute a firm point of departure for exploring the instationary world, which will be seen later. First we have to take a closer look at stationary and isotropic processes.

Definition 2.1.16 Let Z denote a stochastic process with mean m and covariance function C. Then we define:

i) C is *stationary*, if a function $C_{\text{stat}} : \mathbb{R}^d \to \mathbb{R}$ exists such that for all $s, t \in D$ it holds

$$C(s,t) = C_{\text{stat}}(t-s).$$

ii) C is *isotropic*, if a function $C_{\text{iso}} : \mathbb{R}_0^+ \to \mathbb{R}$ exists such that for all $s, t \in D$ it holds

$$C(s,t) = C_{\text{iso}}(\|t-s\|).$$

iii) The process Z is *second-order stationary* (or *weakly stationary*), if $m(\cdot)$ is constant and C is stationary. Furthermore, if C is isotropic, then the process is *isotropic.* For convenience, in this work second-order stationary processes will just be called *stationary.*

iv) *Instationary* and *anisotropic* processes and covariance functions are defined canonically.

Furthermore, if γ denotes the semivariogram of Z, we define:

v) γ is *stationary*, if a function $\gamma_{\text{stat}} : \mathbb{R}^d \to \mathbb{R}$ exists such that for all $s, t \in D$,

$$\gamma(s,t) = \gamma_{\text{stat}}(t-s).$$

vi) γ is *isotropic*, if a function $\gamma_{\text{iso}} : \mathbb{R}_0^+ \to \mathbb{R}$ exists such that for all $s, t \in D$,

$$\gamma(s,t) = \gamma_{\text{iso}}(\|t-s\|).$$

vii) The process Z is *intrinsically stationary*, if $m(\cdot)$ is constant and γ is stationary.

Definition 2.1.17 (Sill and range) If γ is a stationary semivariogram on $D = \mathbb{R}^d$ and $v \in \mathbb{R}^d$ a unit vector, the value

$$\sigma^2(v) := \lim_{h\to\infty} \gamma(\|hv\|),$$

if existent, is called the *sill* of γ in the direction of v. Furthermore,

$$a(v) := \inf\{d \geq 0 : \gamma(\|hv\|) = \sigma^2(v) \; \forall\, h > d\}$$

is the *range* of γ in the direction of v.

Example 2.1.18 The spherical and the exponential semivariogram presented in Example 2.1.12 are stationary and isotropic because they only depend on $\|t-s\|$. The parameters σ^2 and a of the spherical semivariogram are its (omnidirectional) sill and range, respectively.

The exponential semivariogram approaches its parameter σ^2 asymptotically as $h \to \infty$. Hence σ^2 is its sill, but a range in the above sense is not defined. In practice, a value

$$a_\varepsilon := \inf\{d \geq 0 : \gamma(\|hv\|) \geq \sigma^2 - \varepsilon \quad \forall h \geq d\}$$

can be used instead.

Remark 2.1.19 (Geometric anisotropy) For $\theta = (\theta_{\text{sill}}, \theta_{\text{rg}})^T \in \Theta_{\text{sph}}$, let C_θ^{iso} be an isotropic covariogram.

i) For an arbitrary regular matrix $R \in \mathbb{R}^{d\times d}$ consider the function $\widetilde{C}_R : D \times D \to \mathbb{R}$,

$$\widetilde{C}_R(s,t) = C^{\text{iso}}(R(t-s)).$$

If the eigenvalues of R are not identical, $\widetilde{C}_R$ will be an anisotropic covariogram. This kind of anisotropy is called *geometric anisotropy.* In the direction of the greatest eigenvalue, the range of $\widetilde{C}_R$ is $\rho(R)$ times the range of C^{iso}, where $\rho(R) = \max\{|\lambda| : \lambda \text{ eigenvalue of } R\}$ denotes the spectral radius of R.

ii) Geometrically anisotropic processes are stationary, because C^{iso} and R both depend on $t-s$ only.

Remark 2.1.20 (Intrinsic vs. second-order stationarity) The class of all second-order stationary processes is strictly contained in the class of all intrinsically stationary processes.

We give an example from Cressie (1993) that is intrinsically stationary but not second-order stationary. Let W be a zero-mean Gaussian process on $\mathbb{R}^d$, $d > 1$, with covariance

$$C(s, s+h) = \operatorname{Cov}(W(s), W(s+h)) = \tfrac{1}{2}(\|s\| + \|s+h\| - \|h\|).$$

(W is a Brownian motion on $\mathbb{R}^d$.) The process is second-order instationary because C depends on s and h, not only on h.

Using (2.1), we calculate the semivariogram of Z,

$$\gamma(s, s+h) = \tfrac{1}{2}\|s\| + \tfrac{1}{2}\|s+h\| - \tfrac{1}{2}(\|s\| + \|s+h\| - \|h\|) = \|h\|,$$

which is a function of $\|h\|$ only. Hence W is intrinsically stationary.

Note that the semivariogram of W is even isotropic, although the process W is not.

2.2 Van den Boogaart's Method of Semivariogram Construction

2.2.1 Introduction

Now we study processes with covariance functions that are induced by a special kind of weight function. We will see that these approximate stationary covariance functions arbitrarily well.

Weight functions are a very comfortable kit for constructing easy-to-understand covariogram and hence semivariogram models. This is of particular importance in the instationary case and motivates the use of weight functions when dropping stationarity assumptions.

Let $E \subset \mathcal{B}^d$ denote a non-empty measurable set.

Definition 2.2.1 A *weight function* on $D \times E$ is an arbitrary function $w : D \times E \to \mathbb{R}$ such that for all $s \in D$

$$\int_E w(s,p)^2 \, \mathrm{d}p < \infty.$$

Theorem 2.2.2 For an arbitrary weight function w on $D \times E$ and all $s, t \in D$ there exists the integral

$$C_w(s,t) := \int_E w(s,p) w(t,p) \, \mathrm{d}p, \tag{2.3}$$

and the function C_w is positive semidefinite. Furthermore, C_w is the covariance function of a second-order process on D; it is called the covariance function *induced by* w.

Proof: The integral $C_w(s,t)$ exists and is finite, because the product of two square integrable functions $w(s,\cdot)$, $w(t,\cdot)$ is integrable. Furthermore, if for $n \in \mathbb{N}$, we choose arbitrary $t_1, \dots, t_n \in$ and $a_1, \dots, a_n \in \mathbb{R}$, then we obtain

$$\sum_{i=1}^n \sum_{j=1}^n a_i a_j C_w(t_i, t_j) = \sum_{i=1}^n \sum_{j=1}^n a_i a_j \int_E w(s_i,p) w(s_j,p) \, \mathrm{d}p$$
$$= \int_E \sum_{i=1}^n a_i w(s_i,p) \sum_{j=1}^n w(s_j,p) \, \mathrm{d}p = \int_E \left(\sum_{i=1}^n a_i w(s_i,p) \right)^2 \mathrm{d}p \ \geq \ 0.$$

Hence C_w is positive semidefinite. C_w is also symmetric, so Theorem 2.1.7 applies and shows that there exists a second-order stochastic process on D with covariance function C_w. □

Remark 2.2.3 The semivariogram γ corresponding to a covariogram C that is induced by an arbitrary weight function w, is of the form

$$\begin{aligned}\gamma(s,t) &= \tfrac{1}{2}C(s,s) + \tfrac{1}{2}C(t,t) - C(s,t) \\ &= \tfrac{1}{2}\int_{\mathbb{R}^d} \big(w(s,p)^2 + w(t,p)^2 - 2w(s,p)w(t,p)\big)\,\mathrm{d}p \\ &= \tfrac{1}{2}\int_{\mathbb{R}^d} \big(w(s,p) - w(t,p)\big)^2\,\mathrm{d}p. \qquad (2.4)\end{aligned}$$

Definition 2.2.4 (Essential supremum) Recall the following definition: A measurable function $f : \Omega \to \mathbb{R}$ on a region $T \subset \mathbb{R}^d$, is said to be *essentially bounded*, if the following expression exists and is finite:

$$\operatorname{ess.sup}_{x\in T} |f(x)| := \inf_{\substack{Z\subset T\\ \lambda^d(Z)=0}} \sup_{x\in T\setminus Z} |f(x)|.$$

Then $\operatorname{ess.sup}_{x\in T} |f(x)|$ is called the *essential supremum* of f, and we put

$$\|f\|_\infty = \operatorname{ess.sup}_{x\in T} |f(x)|.$$

The following Theorem and Remark are due to van den Boogaart (1999).

Theorem 2.2.5 (Approximation of stationary covariograms) Every stationary covariogram C on $D = \mathbb{R}^d$ that has a spectral density $g(\omega) = \mathrm{d}G/\mathrm{d}\lambda$ can be approximated arbitrarily well with respect to $\|\cdot\|_\infty$ by a covariogram (2.3) induced by a weigth function, i. e.: For every $\varepsilon > 0$ there exists a weight function w on $\mathbb{R}^d \times \mathbb{R}^d$ that induces a covariance function C_w with

$$\|C - C_w\|_\infty < \varepsilon.$$

Proof: The proof consists of two parts: First we construct a sequence $(C_{r_i})_{i\in\mathbb{N}}$ of functions that approximates C arbitrarily well with respect to $\|\cdot\|_\infty$, and then weight functions w_{r_i} are given that induce C_{r_i}, $i \in \mathbb{N}$.

C is a real function on $\mathbb{R}^d$ and has spectral density $g(\omega) = \mathrm{d}G/\mathrm{d}\lambda$, hence it can be written as

$$C(h) = \int_{\mathbb{R}^d} \cos(\omega^T h) g(\omega)\,\mathrm{d}\omega,$$

where $\int_{\mathbb{R}^d} g(\omega)\,\mathrm{d}\omega < \infty$ because G is a bounded measure. For $0 < r_1 \le r_2 \le \ldots$, consider the sequence $(g_{r_i})_{i\in\mathbb{N}}$ of measurable functions

$$g_{r_i} : \mathbb{R}^d \to \mathbb{R}, \qquad g_{r_i}(\omega) = \min\big(r_i, \mathbf{1}_{[0,r_i]}(\|\omega\|) g(\omega)\big)\,.$$

For all $i \in \mathbb{N}$ and $\omega \in \mathbb{R}^d$ it holds

$$0 \le g_{r_i} \le g_{r_{i+1}} \quad \text{and} \quad \lim_{k\to\infty} g_{r_k}(\omega) = g(\omega).$$

Thus, the monotone convergence theorem shows

$$\lim_{i\to\infty} \int_{\mathbb{R}^d} g_{r_i}(\omega)\,\mathrm{d}\omega = \int_{\mathbb{R}^d} g(\omega)\,\mathrm{d}\omega.$$

Putting

$$C_{r_i}(h) = \int_{\mathbb{R}^d} \cos(\omega^T h) g_{r_i}(\omega)\,\mathrm{d}\omega,$$

we get for all $i \in \mathbb{N}$ and $h \in \mathbb{R}^d$

$$\begin{aligned} |C(h) - C_w(h)| &= \left| \int_{\mathbb{R}^d} \cos(\omega^T h)(g(\omega) - g_{r_i}(\omega)) \, \mathrm{d}\omega \right| \\ &\le \int_{\mathbb{R}^d} |\cos(\omega^T h)(g(\omega) - g_{r_i}(\omega))| \, \mathrm{d}\omega \\ &\le \|g - g_{r_i}\|_1; \end{aligned}$$

this does not depend on h, and it is finite because $\|g\|_1$ and $\|g_{r_i}\|_1$ are. Hence $\|C - C_{r_i}\|_\infty < \|g - g_{r_i}\|_1$ for all $i \in \mathbb{N}$, and

$$\lim_{i \to \infty} \|g - g_{r_i}\|_1 = 0$$

then implies

$$\|C - C_{r_i}\|_\infty \to 0$$

as $i \to \infty$.

We know that for all $i \in \mathbb{N}$, g_{r_i} is a bounded function with compact support. Therefore, $\sqrt{g_{r_i}}$ and $\omega \mapsto \cos(\omega^T(s-p))\sqrt{g_{r_i}(\omega)}$ are integrable, the latter for all $s, p \in \mathbb{R}^d$, and we can define a real function $w_{r_i} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ by

$$w_{r_i}(s,p) = \pi^{-d/2} \int_{\mathbb{R}^d} \cos(\omega^T(s-p))\sqrt{g_{r_i}(\omega)} \, \mathrm{d}\omega.$$

We only give a draft of the rest of the proof. Now for all $s, t, p \in \mathbb{R}^d$ it holds

$$\begin{aligned} \int_{\mathbb{R}^d} w_{r_i}(s,p) w_{r_i}(t,p) \, \mathrm{d}p &= \pi^{-d} \int_{\mathbb{R}^d}\int_{\mathbb{R}^d}\int_{\mathbb{R}^d} \cos(\omega_s^T(s-p)) \cos(\omega_t^T(t-p)) \\ &\qquad \cdot \sqrt{g_{r_i}(\omega_s)}\sqrt{g_{r_i}(\omega_t)} \, \mathrm{d}\omega_t \, \mathrm{d}\omega_s \, \mathrm{d}p \\ &= \int_{\mathbb{R}^d} \cos(\omega_s^T(s-t)) g_{r_i}(\omega_s) \, \mathrm{d}\omega_t \\ &= C_{r_i}(s-t). \end{aligned}$$

We have shown that the $\|C - C_{r_i}\|_\infty \to 0$ for $i \to \infty$, and that every C_{r_i} is induced by a weight function w_{r_i}. □

Remark 2.2.6 Assuming the existence of a spectral density in Theorem 2.2.5 implies that neither a nugget effect nor a covariance function not vanishing as $\|h\| \to \infty$ can be approximated arbitrarily well by weight functions. However a nugget effect can be added a posteriori to the induced covariogram (van den Boogaart 1999).

Definition 2.2.7 (Translation invariant weight functions) A weight function w on $D \times E \subset \mathbb{R}^d \times \mathbb{R}^d$ is called *translation invariant*, if there exists a function $w_s : \mathbb{R}^d \to \mathbb{R}$ such that $w(s,p) = w_s(p-s)$ for all $s \in D$, $p \in E$. w is called *isotropic*, if furthermore w_s only depends on $\|p - s\|$.

Theorem 2.2.8 Translation invariant (isotropic) weight functions on $\mathbb{R}^d \times \mathbb{R}^d$ induce stationary (isotropic) covariograms.

Proof: Let w_s be a translation invariant weight function on $\mathbb{R}^d \times \mathbb{R}^d$. Then, putting $q = p - x$, the induced covariogram is

$$C_w(s,t) = \int_{\mathbb{R}^d} w(s,p)w(t,p) \, \mathrm{d}p = \int_{\mathbb{R}^d} w_s(p-x)w_s(p-y) \, \mathrm{d}p = \int_{\mathbb{R}^d} w_s(q)w_s(q-(t-s)) \, \mathrm{d}q,$$

which only depends on $h := t - s$.

Now let w be isotropic. Consider an arbitrary $t' \in \mathbb{R}^d$ with $\|h'\| = \|h\|$, $h' := t' - s$. Let $R \in \mathbb{R}^{d \times d}$ be an orthogonal matrix with $Rh = h'$. Then we have $|\det R| = 1$ and hence, putting $r := Rq$ we get

$$w_s(q)w_s(q-h) = w_s(r)w_s(r-h')$$

and

$$C_w(h) = \int_{\mathbb{R}^d} w_s(q)w_s(q-h)\,\mathrm{d}q = \int_{\mathbb{R}^d} w_s(r)w_s(r-h') \cdot |\det R|\,\mathrm{d}r = C_w(h').$$

We have shown that C_w does not depend on the orientation of h, i. e. it is isotropic. □

Example 2.2.9 (Generalized spherical covariograms) For fixed $R > 0$ and $D \subset E = \mathbb{R}^d$, consider the weight function

$$w : D \times \mathbb{R}^d \to \mathbb{R}, \qquad w(s,p) = \mathbf{1}_{[0,R]}(\|p-s\|), \qquad s \in D, p \in \mathbb{R}^d.$$

We write $\nu^d(\|t-s\|) = \lambda^d(B^d(s,R) \cap B^d(t,R))$. Theorem 2.2.2 implies that the function $C_w : D \times D \to \mathbb{R}$, defined by

$$C_w(s,t) = \int_{\mathbb{R}^d} w(s,p)w(t,p)\,\mathrm{d}p = \begin{cases} \nu^d(\|t-s\|), & \text{if } \|t-s\| < 2R, \\ 0, & \text{otherwise,} \end{cases} \tag{2.5}$$

is a covariogram.

i) $d = 3$: In the not vanishing case, we have to determine the volume of the dissection of two spheres of radius R in $\mathbb{R}^3$, i. e. twice the volume outlined in figure 2.1 (left) with a solid line. Using an equation from Rottmann (1991),

$$\nu^3(\|t-s\|) = 2\frac{\pi\bar{h}}{6}(3r_G^2 + \bar{h}^2),$$

where

$$\bar{h} = R - \tfrac{1}{2}\|t-s\|, \qquad r_G^2 = R^2 - \tfrac{1}{4}\|t-s\|^2.$$

Simple transformations yield

$$\nu^3(\|t-s\|) = \tfrac{4}{3}R^3\pi\left(1 - \frac{3}{2}\frac{\|t-s\|}{2R} + \frac{1}{2}\left(\frac{\|t-s\|}{2R}\right)^3\right).$$

We normalize w dividing it by $\sqrt{\nu^3(0)} = \sqrt{\frac{4}{3}R^3\pi}$ and get $\widetilde{w}$. We set $h := \|t-s\|$ and $a := 2R$, introduce a linear scaling parameter σ^2 and finally yield

$$C_{\widetilde{w}}(h) = \begin{cases} \sigma^2 - \sigma^2\left(\frac{3h}{2a} - \frac{1}{2}\left(\frac{h}{a}\right)^3\right), & \text{if } 0 \le h < a, \\ \sigma^2 & \text{otherwise.} \end{cases} \tag{2.6}$$

This is, by construction, the covariance function of an isotropic second-order process with parameter set D on an appropriate probability space, and hence $\gamma_{\widetilde{w}}(h) := C_{\widetilde{w}}(0) - C_{\widetilde{w}}(h)$ is an isotropic semivariogram. $C_{\widetilde{w}}$ and $\gamma_{\widetilde{w}}$ can be transferred to $D \subset R^l$, $l = 1, 2$, taking $D' := D \times \{0\}$, which is isometrically isomorph to D. Note that when doing this, we still integrate over $E = \mathbb{R}^3$ in (2.5), so we get the same expression (2.6). For $l = 2$, $\gamma_{\widetilde{w}}$ is identical to γ^{sph} from Example 2.1.12.

ii) $d = 2$: Now w is a function on $D \times \mathbb{R}^2$. Put $h := \|t-s\| < 2R$. We have to calculate the area $\nu^2(h)$ of the dissection of two circles of radius R in $\mathbb{R}^2$. See figure 2.1 (right) for notation. Subtracting the triangle from the sector, we get

$$\nu^2(h) = 2\left(R^2\tfrac{\alpha}{2} - \tfrac{1}{2}\xi\tfrac{h}{2}\right) = R^2\alpha - \tfrac{1}{2}sh,$$

where $\alpha = 2\arccos\frac{h}{2R}$ and $\xi = \sqrt{R^2 - h^2/4}$. Thus, we obtain

$$\nu^2(h) = 2R^2\arccos\tfrac{h}{2R} - \tfrac{1}{2}h\sqrt{R^2 - h^2/4}.$$

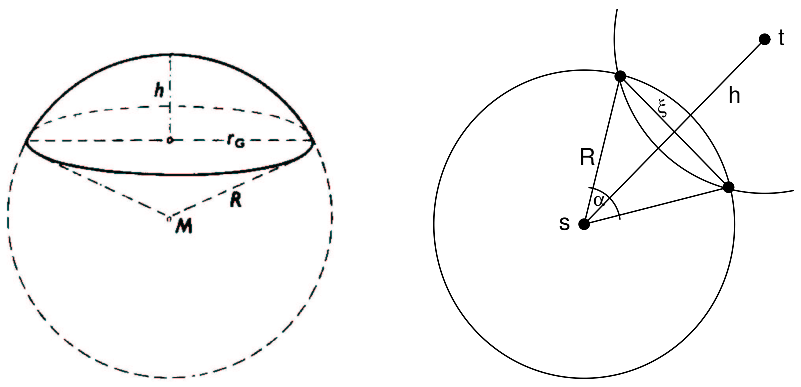


Figure 2.1: Left: A sphere in $\mathbb{R}^3$ with notations from Example 2.2.9 i) (Rottmann 1991). Right: Supports of $w(s,\cdot)$ and $w(s,\cdot)w(t,\cdot)$ (intersection) in Example 2.2.9 ii).

We normalize w and introduce a scaling parameter σ^2:

$$\widetilde{w}(s,p) := \frac{\sigma^2 w(s,p)}{\sqrt{\int_{\mathbb{R}^2} w(s,p)^2 \mathrm{d}p}} = \frac{\sigma^2 w(s,p)}{\sqrt{\lambda(B^2(s,R))}} = \frac{\sigma^2 w(s,p)}{R\sqrt{\pi}}, \qquad s,p \in \mathbb{R}^2,$$

$$C_{\widetilde{w}}(h) := \begin{cases} \frac{\sigma^2}{R^2\pi} \cdot \left(2R^2 \arccos \frac{h}{2R} - \frac{h}{2}\sqrt{R^2 - h^2/4}\right), & \text{if } h < 2R, \\ 0 & \text{otherwise.} \end{cases}$$

The induced covariogram $C_{\widetilde{w}}$ is isotropic, we write $C_{\widetilde{w}}(s,t) = C_{\widetilde{w}}(\|t-s\|)$, and the corresponding semivariogram (see figure 2.2) $\gamma_{\widetilde{w}}(h) = C_{\widetilde{w}}(0) - C_{\widetilde{w}}(h) = \sigma^2 - C_{\widetilde{w}}(h)$ is also isotropic. $C_{\widetilde{w}}$ is continuous, but its first derivative has a singularity at $h = 2R$ (or, more precisely, at every $(s, s+h) \in D \times D$ with $\|h\| = 2R$), unlike its "brother" C^{sph} constructed above for $d = 3$, $l = 2$.

Remark and Definition 2.2.10 The normalizing procedure used in the preceding example will be applied frequently. For convenience, we define:

Let $w : D \times E \to \mathbb{R}$ be an arbitrary weight function. Then for all $s \in D$,

$$\nu(w,s) := \left(\int_E w(s,\cdot)\,\mathrm{d}p\right)^{1/2} \qquad \text{and} \qquad \mathcal{N}(w) := w/\nu(w,\cdot)$$

are well-defined, and $\mathcal{N}(w)$ is a weight function. It induces a covariogram satisfying

$$C(s,s) = 1 \qquad \text{for all } s \in D.$$

2.2.2 Covariance Functions as Convolutions

In this section, we study stationary covariance functions that are induced by translation invariant weight functions on $D = E = \mathbb{R}^d$. We will show that these covariance functions are twice as many times differentiable as the corresponding weight function. First a few technical results will be proven.

Definition 2.2.11 (Convolution) Consider two functions $f, g \in L^1(\mathbb{R}^d)$. If the integral

$$(f * g)(s) := \int_{\mathbb{R}^d} f(r)g(s-r)\,\mathrm{d}r \tag{2.7}$$

exists at $s \in \mathbb{R}^d$, then $(f*g)(s)$ is called the *convolution integral* of f and g at s. If $(f*g)(s)$ exists at least for almost all $s \in \mathbb{R}^d$, then $f*g$ is called the *convolution* of f and g.

Theorem 2.2.12 For $f, g \in L^1(\mathbb{R}^d)$, the convolution integral $(f * g)(s)$ exists for all $s \in \mathbb{R}^d$ and is integrable.

Proof: We follow Grabmüller (1999). The functions $(s,p) \mapsto f(p)$ and $(s,p) \mapsto g(s-p)$ are by inspection measurable with respect to the product σ-algebra $\mathcal{B}^d \times \mathcal{B}^d$ on $\mathbb{R}^d \times \mathbb{R}^d$, because f and g are measurable. Hence $\phi(s,t) := f(p)g(s-p)$ is also measurable. Integrating $|\phi(s,t)|$,

$$\int_{\mathbb{R}^d}\left(\int_{\mathbb{R}^d}|\phi(s,t)|\,\mathrm{d}s\right)\mathrm{d}t \overset{u:=s-p}{=} \int_{\mathbb{R}^d}|f(t)|\left(\int_{\mathbb{R}^d}|g(u)|\,\mathrm{d}u\right)\mathrm{d}t = \|f\|_1\|g\|_1 < \infty,$$

hence Fubini's theorem shows $f * g \in L^1(\mathbb{R}^d)$. □

Theorem 2.2.13 (Derivatives of a parametric integral) Let $T \subset \mathbb{R}^k$, $k \geq 1$, and $E \subset \mathbb{R}^d$ be regions. Consider a function $f : T \times E \to \mathbb{R}$, and assume that for all $\tau \in T$, $p \mapsto f(\tau, p)$ is integrable. Define a function $I : T \to \mathbb{R}$ by

$$I(\tau) = \int_E f(\tau, p)\,\mathrm{d}p.$$

Now suppose that $V \subset T$ is a neighbourhood of $\tau^* \in T$ such that the following two conditions hold:

i) For almost all $p \in E$, $f(\cdot, p) : \tau \mapsto f(\tau, p)$ is continuously differentiable on V.

ii) There exists an integrable function $g : V \to \mathbb{R}$ such that for all $\tau \in V$:

$$\left|\frac{\partial}{\partial\tau} f(\tau, p)\right| \leq g(p) \quad \text{almost everywhere.}$$

Then I is differentiable at τ^*, and it holds

$$\frac{\mathrm{d}}{\mathrm{d}\tau} I(\tau^*) = \int_E \frac{\partial}{\partial\tau} f(\tau^*, p)\,\mathrm{d}p.$$

Proof: We follow Gasquet and Witomski (1999). Let $(\tau_n)_{n\in\mathbb{N}}$ be an arbitrary sequence in V that converges to τ^*. Define

$$d_n(p) = \frac{f(\tau_n, p) - f(\tau^*, p)}{\tau_n - \tau^*}, \qquad p \in E.$$

Due to the mean value theorem, for all $p \in E$ and all $n \in \mathbb{N}$ there exists a $\widetilde{\tau}_n(p) \in V$ such that

$$d_n(p) = \frac{\partial}{\partial\tau} f(\widetilde{\tau}_n(p), p).$$

From the hypothesis, $\partial f / \partial\tau$ is continuous in τ^* for almost all $p \in E$, so

$$\lim_{n\to\infty} d_n(p) = \frac{\partial}{\partial\tau} f(\tau^*, p)$$

for almost all $p \in E$, and

$$|d_n(p)| = \left|\frac{\partial f}{\partial\tau}(\widetilde{\tau}_n(p), p)\right| \leq g(p)$$

λ^d-almost everywhere. Lebesgue's dominant convergence theorem then implies

$$\lim_{n\to\infty}\int_E d_n(p)\,\mathrm{d}p = \int_E \lim_{n\to\infty} d_n(p)\,\mathrm{d}p$$

and hence the proposition holds. □

Theorem 2.2.14 Suppose $f \in L^p(\mathbb{R}^d)$ and $g \in L^q(\mathbb{R}^d)$, where $1 \leq p, q \leq \infty$ and $\frac{1}{p} + \frac{1}{q} = 1$. Then the following propositions hold:

i) $(f * g)(\tau)$ is defined for all $\tau \in \mathbb{R}^d$.

ii) $f * g$ is uniformly continuous and bounded.

iii) $\|f * g\|_\infty \leq \|f\|_p \|g\|_q$.

Proof: i) holds because of Theorem 2.2.12 and $\forall\, p > 1 : \; L^p(\mathbb{R}^d) \subset L^1(\mathbb{R}^d)$.

We now show iii). Without loss of generality, assume $q < \infty$. From Hölder's inequality we have

$$|(f * g)(s)| \leq \|f\|_p \left(\int_{\mathbb{R}^d} |g(s-r)|^q \mathrm{d}r \right)^{1/q} = \|f\|_p \|g\|_q, \qquad \textit{for all } s \in \mathbb{R}^d,$$

and hence $\|f * g\|_\infty \leq \|f\|_p \|g\|_q$.

ii) Now we only have to prove uniform continuity. We write

$$\begin{aligned} |(f * g)(s) - (f * g)(t)| &\leq \int_{\mathbb{R}^d} |f(r)| \cdot |g(s-r) - g(t-r)|^q \, \mathrm{d}r \\ &\leq \|f\|_p \left(\int_{R^d} |g(s-r) - g(t-r)|^q \, \mathrm{d}r \right)^{1/q} \end{aligned}$$

for all $s, t \in \mathbb{R}^d$. We first establish continuity when g is continuous with compact support. Let $A \supset \operatorname{supp} g$ be a region. For $\|s - t\|$ sufficiently small,

$$\begin{aligned} \int_{\mathbb{R}^d} |g(s-r) - g(t-r)|^q \, \mathrm{d}r &\overset{u:=t-r}{=} \int_A |g(s-t+u) - g(u)|^q \, \mathrm{d}u \\ &\leq \quad 2\lambda^d(A) \cdot \sup_{u \in \overline{A}} |g(s-t+u) - g(u)|. \end{aligned}$$

The supremum is finite since g is uniformly continuous on $\overline{A}$, and hence $f * g$ is uniformly continuous on $\mathbb{R}^d$.

Now allow $g \in L^q(\mathbb{R}^d)$ and recall that the linear space C_c^0 of all continuous functions with compact support in $\mathbb{R}^d$ is dense in $L^q(\mathbb{R}^d)$. Let $(g_n)_{n \in \mathbb{N}} \subset C_c^0$ be a sequence with $\lim_{n \to \infty} \|g_n - g\|_q = 0$. Adding and subtracting $(f * g_n)(s)$ and $(f * g_n)(t)$, we get

$$\begin{aligned} |(f * g)(s) - (f * g)(t)| &\leq |(f * g)(s) - (f * g_n)(s)| + |(f * g_n)(t) - (f * g)(t)| \\ &\quad + |(f * g_n)(s) - (f * g_n)(t)| \\ &\leq 2\|f\|_p \|g - g_n\|_q + |(f * g_n)(s) - (f * g_n)(t)|, \end{aligned}$$

using Hölder's inequality. By construction, $\|f\|_p \|g - g_n\|_q \to 0$ for $n \to \infty$, and the last term is uniformly continuous for all $n \in \mathbb{N}$ due to the inequality shown above. It follows directly that $f * g$ is uniformly continuous on E. □

Theorem 2.2.15 Suppose $f \in L^1(\mathbb{R}^d)$ and $g \in C^k(\mathbb{R}^d)$, and let the αth derivative[2] $\partial^\alpha g / \partial s^\alpha$ of g be bounded for all α with $0 \leq |\alpha| \leq k$. Then it holds

$$f * g \in C^k(\mathbb{R}^d) \quad \textit{and} \quad \frac{\partial^\alpha (f * g)}{\partial s^\alpha} = f * \frac{\partial^\alpha g}{\partial s}.$$

[2]As usual, $\alpha \in \mathbb{N}_0^d$ denotes a multi-index, and we write $|\alpha| = \sum_i \alpha_i$.

Proof: For the αth derivative of a function h, we write $h^{(\alpha)}$. By applying Theorem 2.2.14 with $p = 1$ and $q = \infty$ we see that $f * g^{(\alpha)}$ is continuous for all α with $0 \leq |\alpha| \leq k$. The function $s \mapsto f(p)g(s-p)$ is k times differentiable, and for all $0 \leq |\alpha| \leq k$ we have

$$|f(r)g^{(\alpha)}(s-r)| \leq |f(r)| \cdot \sup_{u \in \mathbb{R}^d} |g^{(\alpha)}(u)|.$$

Since $f \in L^1(\mathbb{R}^d)$, we can integrate under the integral sign (Theorem 2.2.13). Hence

$$(f * g)^{(\alpha)}(s) = \int_{\mathbb{R}^d} f(p)g^{(\alpha)}(s-r)\,\mathrm{d}r = (f * g^{(\alpha)})(s).$$

□

Corollary 2.2.16 (Differentiability of covariance functions) Let w be a k times continuously differentiable translation invariant weight function, and suppose that $\partial^\alpha w/\partial s^\alpha$ is bounded for all $0 \leq |\alpha| \leq k$. Then the induced stationary covariance function $C_w : \mathbb{R}^d \to \mathbb{R}$ is $2k$ times continuously differentiable, and it holds

$$\frac{\partial^\beta C_w}{\partial s^\beta} = \frac{\partial^{\alpha_1} w}{\partial s^{\alpha_1}} * \frac{\partial^{\alpha_2} w}{\partial s^{\alpha_2}},$$

for all $\beta, \alpha_1, \alpha_2 \in \mathbb{N}^d$ with $0 \leq |\beta| \leq 2k$, $0 \leq |\alpha_{1,2}| \leq k$ and $\alpha_1 + \alpha_2 = \beta$.

Proof: Apply Theorem 2.2.15 twice. Note that the square integrability of the weight function is not needed here. □

Remark 2.2.17 The differentiability of a stationary covariogram C is closely related to the "smoothness" of the corresponding process in terms of L^2-differentiability. A second-order process Z on $\mathbb{R}^d$ is said to be *L^2-differentiable* at $s \in \mathbb{R}^d$ if $(Z_{s+h_je_j} - Z_s)/h_j$ converges in L^2 as $h_j \to 0$, $j = 1, \ldots, d$, where $(e_j)_{j=1,\ldots,d}$ is the natural basis of $\mathbb{R}^d$. If $C : \mathbb{R}^d \to \mathbb{R}$ is two times differentiable at 0, then Z is L_2-differentiable at all $s \in \mathbb{R}^d$. (See Cressie (1993, p. 60) for details and references.)

Remark 2.2.18 (Fourier transform of a covariance function) Consider a translation invariant weight function w on $\mathbb{R}^d$ and the induced covariance function $c = w * w$. We can apply results from Fourier Analysis to this class of covariance functions: The Fourier transform $C(\omega)$ of $c(h)$ can be determined using

$$C(\omega) = W(\omega)^2,$$

where $W(\omega)$ denotes the Fourier transform of w. This is a consequence of the Convolution Theorem.

Studying the Fourier transform or spectral density of stationary covariograms allows inference on the local behaviour of the stochastic process (or the assumed model). See Stein (1999) for an application to geostatistics.

2.2.3 Modeling Local Anisotropy: the Elliptical Class of Models

Theorem 2.2.5 shows that the class of covariograms induced by a weight function is sufficiently large as to be useful instruments for covariance modeling. In the following, the construction method will be used for creating a class of semivariograms and covariograms that can adapt to local anisotropies that may be quite irregular, but following a known pattern, e. g. topography or tectonic structures.

In this subsection, we choose $d = 2$ and $E = \mathbb{R}^2$. However, the approach followed here can easily be transferred to processes with higher-dimensional parameter sets.

Definition 2.2.19 (Elliptical semivariograms) Let $w^o_\tau : \mathbb{R} \to \mathbb{R}$, $\tau \in T \neq \emptyset$, be a square integrable function with support $\subset B^2(0,1)$. For $\phi \in [0,\pi[$, $r \geq 0$ and $q \in]0,1]$ we define

$$R(\phi; r, q) = \frac{1}{r}\begin{pmatrix} \cos\phi & \sin\phi \\ -q^{-1}\sin\phi & q^{-1}\cos\phi \end{pmatrix}$$

to be a combined contraction and rotation by $-\phi$ satisfying

$$R(\phi; r, q)\mathrm{Ell}_2(0; r, q, \phi) = B^2(0,1),$$

where $\mathrm{Ell}(0; r, q, \phi)$ denotes the two-dimensional ellipse around 0 with longer radius r in an angle of ϕ with the x_1-axis and axis ratio q.

Now consider a function $\theta = (\sigma^2, \phi, a, q, \tau) : D \to \mathbb{R}^+_0 \times [0,\pi[\times \mathbb{R}^+ \times]0,1] \times T$. Then we define

$$w^*_{(a,q,\tau)} : D \times \mathbb{R}^d \to \mathbb{R}, \qquad w^*_\theta(s,p) = w^o_{\tau(s)}(\|R(\phi(s); \tfrac{a(s)}{2}, q(s))(p-s)\|),$$
$$w^{\mathrm{ell}}_\theta = \sigma^2 \mathcal{N}(w^*_\theta),$$

where $\mathcal{N}$ is the normalizing functional from Remark and Definition 2.2.10. Then the weight function w^{ell}_θ is called an *elliptical weight function* on D. Covariograms and semivariograms induced by elliptical weight functions are also called *elliptical.* w_0 will be referred to as a *kernel function*.

A component of θ is called a *parameter*, if it is a constant function, otherwise a *covariable*.

Remark 2.2.20 In this work, the elliptical semivariograms considered have parameters σ^2, a, q and τ, and one single covariable ϕ, unless specified otherwise.

Remark 2.2.21 i) Kernel functions for elliptical semivariograms are for example

$$w^{\mathrm{ind}}(h) = \mathbf{1}_{[-1,1]}(h), \qquad \text{(“\textit{simple kernel function}”)}$$

$$w^{\mathrm{lin}}(h) = \begin{cases} 1-|h| & \text{if } |h|<1, \\ 0 & \text{otherwise,} \end{cases} \qquad \text{(“\textit{linear kernel fn.}”)}$$

$$w^{\mathrm{pwl}}_b(h) = \begin{cases} 1 & \text{if } |h| \leq b, \\ 1-(|h|-b)/(1-b) & \text{if } b<|h|<1, \\ 0 & \text{otherwise,} \end{cases} \qquad \text{(“\textit{piecewise linear kernel fn.}”)}$$

$$w^{\mathrm{bez}}_\nu(h) = \begin{cases} (h+1)^\nu (h-1)^\nu & \text{if } |h|<1, \\ 0 & \text{otherwise,} \end{cases} \qquad \text{(“\textit{Bezier kernel fn.}”)}$$

Simple elliptical semivariograms (induced by the simple kernel function) with $q = 1$ are isotropic and coincide with the semivariogram $\gamma_{\widetilde{w}}$ in Example 2.2.9 ii).

The covariograms and semivariograms induced by the Bezier kernel function (see figure 2.2) are at least $\lfloor 2\nu \rfloor$ times continuously differentiable because w^{bez}_0 is (exactly) $\lfloor \nu \rfloor$ times continuously differentiable (Corollary 2.2.16). The other kernel functions presented are not differentiable on $\partial B^2(0,1)$.

ii) For $\Theta =]0,\infty[\times]0,\infty[\times]0,1[\times T$ and an arbitrary kernel function w^o_τ, the family of all induced semivariograms $\gamma_\theta = \gamma^{\mathrm{ell},w^o_\tau}_{(\sigma^2,a,q)}$, $\theta = (\sigma^2, a, q, \tau) \in \Theta$, form a semivariogram model $(\gamma^{\mathrm{ell}}_\theta)_{\theta\in\Theta}$.

iii) Simple elliptical covariograms (i. e. for $w_0 = w^{\mathrm{ind}}_0$) can be written as

$$C_\theta(s,t;\phi) = \frac{\sigma^2}{\lambda(\mathrm{Ell}(0; a/2, q))}\lambda\left(\mathrm{Ell}(s; a/2, q, \phi(s)) \cap \mathrm{Ell}(t; a/2, q, \phi(t))\right),$$

i. e. they represent a standardized measure for the overlapping of ellipses.

Even the simple area of dissection of these ellipses is difficult to determine analytically. Therefore quasi-Monte Carlo integration methods will be applied in this work in order to approximate elliptical semivariograms (see Sections 3.2.3 and 3.3.5).

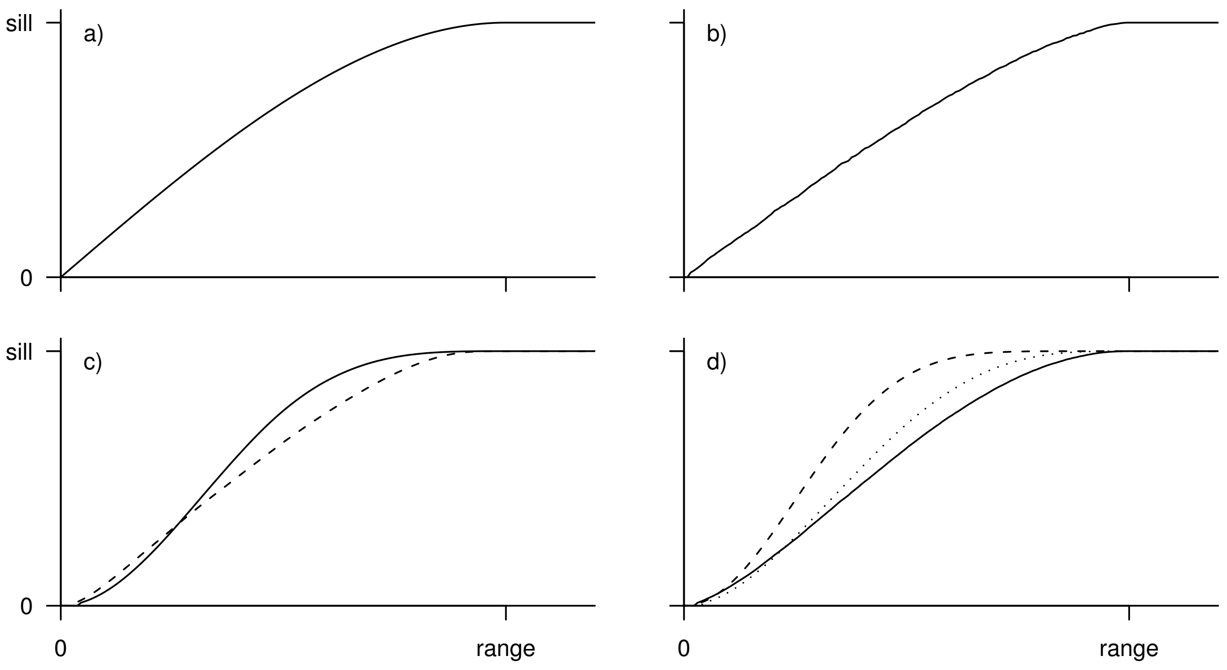


Figure 2.2: Semivariogram plot of the form $\gamma(s, s + h\nu)$, $h \in \mathbb{R}$, $\|\nu\| = 1$, for:
a) Spherical semivariogram (Example 2.2.9 i)).
b) Simple elliptical semivariogram (Remark 2.2.21, Example 2.2.9 ii)).
c) Elliptical semivariograms corresponding to the linear (solid line) and piecewise linear (break point $b = 0.8$; dashed) kernel function.
d) Elliptical semivariograms corresponding to the Bezier kernel function with exponents $\nu = 0.3$ (solid line), 1 (dotted) and 3 (dashed).

Remark 2.2.22 The following example shows how the class of elliptical semivariograms can be extended, and how parameters and covariables can be chosen in order to represent qualitative knowledge of the processes to model.

Example 2.2.23 (Modeling soil loss by water run-off) Soil erosion by water run-off is a geomorphological process that basically depends on topography, vegetation, land use, soil properties and precipitation regime. At a small scale, however, the size of the catchment area and the slope's inclination are the most important factors that influence soil erosion.

We wish to construct weight functions and semivariograms that are consistent with our knowledge of the processes governing soil erosion. In particular we know that the soil loss at one point depends on the soil loss uphill in the same catchment area, and that there is little correlation with soil loss on the other side of a ridge or on the opposite side of a valley. We want to represent this knowledge of correlation being restricted to a catchment area. (In addition, large scale dependencies may be modeled with a different semivariogram.)

For a point $s \in D$, let $A(s) \subset \mathbb{R}^2$ denote its catchment area, i. e. the area of land that drains to s. Then a weight function w on $D \times \mathbb{R}^d$ with $\operatorname{supp} w(s, \cdot) = A(s)$ respects our knowledge of the relation between soil erosion and topography.

As an example, we define a weight function by

$$w_{\tilde{\theta}}(s, p) = w_\theta^{\text{ell}}(s, p)\mathbf{1}_{A(s)}(p), \qquad \tilde{\theta} = (\theta, \mathbf{1}_{A(\cdot)}),$$

where w_θ^{ell} is an arbitrary elliptical weight function with covariable $a : s \mapsto 2\sup_{p\in A(s)} \|p - s\|$ and parameters σ^2 and τ, the axis ratio $q = 1$ being constant and hence ϕ without any effect.

The catchment area of a point can be determined by analyzing Digital Elevation Models (DEM), and a great number of oracle calls $\mathbf{1}_{A(s)}(p)$ is necessary in order to approximate the induced semivariogram by numerical integration.

A computationally less demanding method is the following, which approximates $w_{\tilde{\theta}}$ quite well in not too irregular relief. Let $R(s)$ be the shortest distance to the ridge that lies uphill from s, and let $\phi(s) \in [0, 2\pi[$ denote the gradient of topography at s expressed as an angle, and $\delta(s) \in]0, \pi]$ an opening angle. Then for an arbitrary elliptical weight function w_θ^{ell}, with $q = 1$ and covariables $a = 2R$ and ϕ, we define

$$w'_{\tilde{\theta}}(s,p) = w_\theta^{\text{ell}}(s,p)\mathbf{1}_{\text{Sec}(s;\phi(s),\delta(s),R(s)/2)}(p),$$

where $\text{Sec}(s;\phi(s),\delta(s),a(s)/2) \subset B^2(s,a(s)/2)$ denotes the sector with opening angle $\delta(s)$ and radius $a(s)/2$ oriented according to $\phi(s)$. The opening angle $\delta(s)$ could for example be constant or a function of local curvature at s.

2.2.4 Modeling Boundaries between Subprocesses

In many geostatistical applications we find the following situation: The parameter set D of the process Z of interest decomposes into η disjoint subsets $D_1, \dots, D_\eta \subset D$ such that the subprocesses $Z_1 := Z|_{D_1}, \dots, Z_\eta := Z|_{D_\eta}$ have little or no correlation between each other.

Instead of studying each subprocess Z_i separately, one might wish to study the process Z as a whole.

Question 2.2.24 How can boundaries between subprocesses with little or no correlation be modeled using weight functions?

In this subsection we follow a constructivist approach, studying transformations of weight functions and their effects on the induced covariogram model rather than asking for necessary conditions on the weight functions given certain properties of subprocesses. Consequently, the definitions presented here are mainly intended to be useful for practical purposes.

Definition 2.2.25 (Ordinary and strong boundaries) Let $w_1^o, \dots, w_\eta^o$ be weight function on $D \times E$, $E = \mathbb{R}^d$. Then we define:

i) The weight functions w_i on $D \times E$,

$$w_i(s,p) := w_i^o(s,p)\mathbf{1}_{D_i}(s), \qquad i = 1, \dots, \eta,$$

and the induced covariograms $C_1, \dots, C_\eta$ are said to have *ordinary boundaries* (between $D_1, \dots, D_\eta$).

ii) The weight functions w_i^* on $D \times E$,

$$w_i^*(s,p) := w_i^o(s,p)\mathbf{1}_{D_i}(s)\mathbf{1}_{D_i}(p), \qquad i = 1, \dots, \eta,$$

and the induced covariograms $C_1^*, \dots, C_\eta^*$ are said to have *strong boundaries* (between $D_1, \dots D_\eta$).

If the covariograms $C_1, \dots, C_\eta$ have ordinary (strong) boundaries between $D_1, \dots, D_\eta$, then a process Z on D with covariogram

$$C = \sum_{i=1}^{\eta} C_i$$

is also said to have ordinary (strong) boundaries between $D_1, \dots, D_\eta$.

In a colloquial way, let us agree on speaking of ordinary boundaries only if they are not strong.

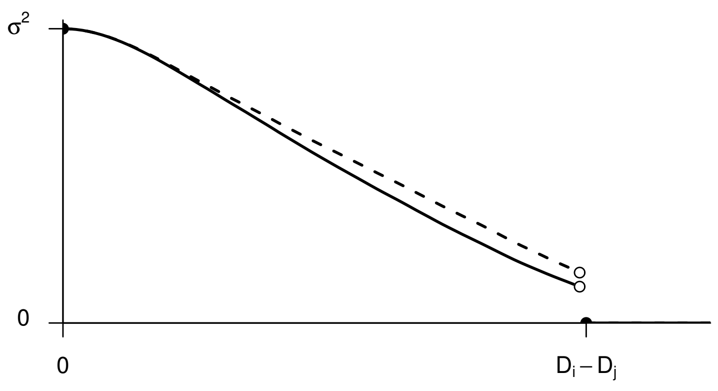


Figure 2.3: Covariograms with ordinary and strong boundaries near the boundaries.

Remark 2.2.26 If a stochastic process Z on D has ordinary or strong boundaries between $D_1, \dots, D_\eta$, then the subprocesses $Z|_{D_1}, \dots, Z|_{D_\eta}$ are pairwise uncorrelated.

Proof: For $i \neq j$, consider arbitrary $s \in D_i$, $t \in D_j$. We have to show that $C(s,t) = 0$. For $k = 1, \dots, \eta$ and all $p \in E$, we have

$$w_k(s,p)w_k(t,p) = w_k^o(s,p)w_k(t,p)\mathbf{1}_{D_k}(s)\mathbf{1}_{D_k}(t) = 0,$$

since at least one of $\mathbf{1}_{D_k}(s)$ and $\mathbf{1}_{D_k}(t)$ is zero. Hence $C(s,t) = 0$. □

Remark 2.2.27 Suppose that the process Z with covariogram $C = \sum_{i=1}^{\eta} C_i$ has ordinary or strong boundaries between $D_1, \dots, D_\eta$. Then its semivariogram writes

$$\gamma = \sum_{i=1}^{\eta} \gamma_i, \qquad 2\gamma_i(s,t) = \begin{cases} 0, & \text{if } s,t \notin D_i, \\ C_i(s,s), & \text{if } s \in D_i,\ t \notin D_i, \\ C_i(t,t), & \text{if } s \notin D_i,\ t \in D_i, \\ C_i(s,s) + C_i(t,t) - 2C_i(s,t), & \text{if } s,t \in D_i. \end{cases}$$

Remark 2.2.28 Let Z be a process with ordinary boundaries and covariogram $C = \sum_{i=1}^{\eta} C_i$. Suppose that for $i = 1, \dots, \eta$, C_i is induced by the weight function $(s,p) \mapsto w_i^o(s,p)\mathbf{1}_{D_i}(s)$, where w_i^o induces a covariogram on D_i that is generically stationary with respect to $g_i : D_i \to \mathbb{R}^k$. Define $g : D \to \mathbb{R}^{k+\eta}$ by $g|_{D_i} := (g_i, \mathbf{1}_{D_1}, \dots, \mathbf{1}_{D_\eta})^T$.

Then Z is generically stationary on D with respect to g.

Remark 2.2.29 i) Near the boundaries, covariograms with ordinary and strong boundaries are greater than their analogues with ordinary boundaries (figure 2.3). This is due to the normalizing factor that decreases rapidly as the integration domain is being cut by $\mathbf{1}_{D_i}$.

ii) When approximating covariograms with strong boundaries using numerical integration, the functions $\mathbf{1}_{D_i}$, $i = 1, \dots, \eta$, have to be evaluated at each node $p \in E$. In practice, $D_1, \dots, D_\eta$ generally are polygons stored within a Geographical Information System, and evaluations of $\mathbf{1}_{D_i}$ have to be considered as expensive "oracle calls" (see Section 3.2.3).

Now we will consider a way of constructing covariograms of correlated subprocesses. Instead of adding covariograms of subprocesses, we will add weight functions.

Definition 2.2.30 (Soft boundaries) Let $w_1^o, \dots, w_\eta^o$ be arbitrary weight functions on $D \times E$, $E = \mathbb{R}^d$. Then the weight function $w : D \times E \to \mathbb{R}$, defined by

$$w(s,p) = \sum_{i=1}^{\eta} w_i(s,p)\mathbf{1}_{D_i}(s), \tag{2.8}$$

and the induced covariogram C are said to have *soft boundaries* between $D_1, \dots, D_\eta$.

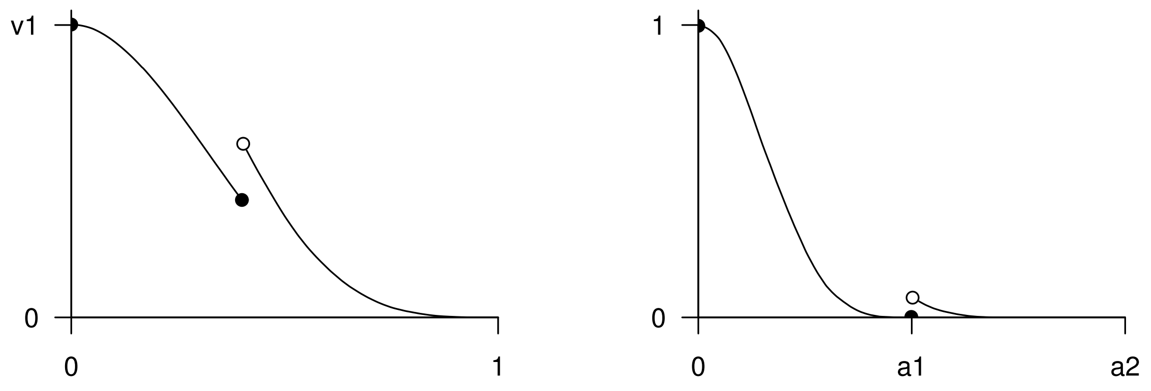


Figure 2.4: Covariograms $C(s, s+h\nu)$ (left) and $C'(s, s+h\nu)$ from Example 2.2.32 ($s \in A_1$, $\|\nu\| = 1$).

Remark 2.2.31 Suppose that Z is a process with soft boundaries, its covariogram being induced by a weight function $(s,p) \mapsto \sum_{i=1}^{\eta} w_i(s,p)\mathbf{1}_{D_i}(s)$. Let Z_i be a process on D_i with covariogram induced by $w_i|_{D_i \times E}$, $i = 1, \ldots, \eta$. If for all $i = 1, \ldots, \eta$ the process Z_i is generically stationary with respect to $g_i : D_i \to \mathbb{R}^k$, then Z is generically stationary with respect to $g : D \to \mathbb{R}^{k+\eta}$, $g|_{D_i} = (g_i, \mathbf{1}_{D_1}, \ldots, \mathbf{1}_{D_\eta})^T$.

Example 2.2.32 Suppose $D_1 \cup D_2 = D$, $D_1 \cap D_2 = \emptyset$, and let C, C' denote covariograms with soft boundaries between D_1 and D_2 induced by weight functions

$$(s,p) \mapsto w^{\text{lin}}_{(\sigma_1^2,1,1)}(s,p)\mathbf{1}_{D_1}(s) + w^{\text{lin}}_{(\sigma_2^2,1,1)}(s,p)\mathbf{1}_{D_2}(s)$$

and

$$(s,p) \mapsto w^{\text{lin}}_{(1,a_1,1)}(s,p)\mathbf{1}_{D_1}(s) + w^{\text{lin}}_{(1,a_2,1)}(s,p)\mathbf{1}_{D_2}(s),$$

respectively (cf. Definition 2.2.19, Remark 2.2.21), where $\sigma_1^2 < \sigma_2^2$ determine different sills on D_1 and D_2, and $a_1 < a_2$ are different ranges. This may produce covariograms as shown in figure 2.4.

The discontinuity of C can be accepted if we take into account that the correlogram $\bar{C}(s,t) = C(s,t)/\sqrt{C(s,s)C(t,t)}$ does not depend on σ_1^2, σ_2^2 but remains continuous.

However, it will generally not be desirable to introduce a positive correlation $\bar{C}'(s,t)$ for $s \in A_1$ and $\|t - s\| > a_1$. This effect will however be negligeable if the ranges a_1 and a_2 do not defer too much and the weight functions are small near the support's boundary.

Remark 2.2.33 (Smooth transitions) The problem encountered in Example 2.2.32 with soft boundaries may be overcome by allowing the weight function's parameter $\theta \in \Theta$ to vary smoothly in space, i. e. taking $\widetilde{w}(s,p) = w_{f(s)}(s,p)$ for a smooth function $f : D \to \Theta$. In such a situation, the induced covariogram is said to have *smooth transitions*.

This approach follows the same principle that was used when turning from geometric anisotropies to those modeled by elliptical covariograms with smoothly varying direction of anisotropy $\phi(s)$ (see Remark 2.1.19 and Definition 2.2.19).

Summary 2.2.34 When modeling independent subprocesses, prefer ordinary boundaries to strong ones, and simply add covariograms of subprocesses.

When modeling smooth transitions within a process or between subprocesses, use semivariogram parameters smoothly varying in space, rather than soft boundaries.

2.3 Generic Stationarity: towards Stationarizing Instationarity

In Section 2.2.3 the class of elliptical semivariograms was presented, which were found to be in general non-geometrically anisotropic. However, by construction their anisotropy can be considered as rather regular because it is merely determined by the direction $\phi(s)$, $s \in D$, of local anisotropy, which is assumed to be known.

Now we want to introduce a concept that is more general than that of stationarity and includes the elliptical case and other situations with "understandable" anisotropies.

Example 2.3.1 Let Z be a process on $E = \mathbb{R}^d$ with elliptical semivariogram γ with parameter $\theta \in \Theta$ and direction of local anisotropy $\phi : \mathbb{R}^d \to [0, \pi[$. Then for all $s, s+h, t, t+h \in D$ with $\phi(s) = \phi(t)$ and $\phi(s+h) = \phi(t+h)$ it holds

$$\gamma(s, s+h) = \gamma(t, t+h) \tag{2.9}$$

(see the proof given below). That is, elliptical semivariograms have "something like" a stationarity property conditional on the direction of local anisotropy ϕ.

However, there will probably exist points $u, u+h \in D$ with $\phi(s) \neq \phi(u)$ or $\phi(s+h) \neq \phi(u+h)$. If we go beyond the semivariogram itself and study the way how the generation of our semivariogram depends on ϕ, we will find out that for all $s, t \in D$, we have

$$\gamma(s,t) = \gamma_g(t - s \,|\, \phi(s), \phi(t)),$$

where the function $\gamma_g : \mathbb{R}^d \times [0,\pi[\times [0,\pi[\to \mathbb{R}$, is defined by

$$\gamma_g(h \,|\, \phi_1, \phi_2) = \sigma^2 \int_{\mathbb{R}^d} \mathcal{N} w^o_\theta(\|R(\phi_1;\theta)(p)\|) \mathcal{N} w^o_\theta(\|R(\phi_2;\theta)(p-h)\| \, \mathrm{d}p. \tag{2.10}$$

Thus, γ depends on $t - s$, $\phi(s)$ and $\phi(t)$ only.

The last two equations suggest that "if we *had* $\phi(s) = \phi(u)$ and $\phi(s+h) = \phi(u+h)$, then we would get $\gamma(s, s+h) = \gamma(t, t+h)$" — a hypothetic version of (2.9), based on our belief in the validity of the law expressed in (2.10).

The following definition reflects this concept in a precise way.

Proof: Let $w^o : \mathbb{R} \to \mathbb{R}$ be a kernel function such that γ is equal to the induced elliptical semivariogram with suitable parameter $\theta = (\sigma^2, a, q)$. Consider arbitrary $s, s+h, t, t+h \in D$ with $\phi(s) = \phi(t)$ and $\phi(s+h) = \phi(t+h)$ as above, and let ν be the normalizing function from Remark and Definition 2.2.10. Then it holds

$$R(\phi(s); \tfrac{a}{2}, q) = R(\phi(t); \tfrac{a}{2}, q), \qquad R(\phi(s+h); \tfrac{a}{2}, q) = R(\phi(t+h); \tfrac{a}{2}, q).$$

Because of this and $p - \tilde{s} = (p + \tilde{t} - \tilde{s}) - \tilde{t}$, $\tilde{s}, \tilde{t} \in D$, we get for the weight function w_θ corresponding to γ

$$w_\theta(s,p) = w_\theta(t, p+t-s), \qquad w_\theta(s+h,p) = w_\theta(t+h, p+t-s).$$

The translation invariance of the integral over $E = \mathbb{R}^d$ then yields (using the translation $p \mapsto p' = p + t - s$)

$$\int_E w_\theta(s,p) w_\theta(s+h,p) \, \mathrm{d}p = \int_E w_\theta(t,p') w_\theta(t+h,p') \, \mathrm{d}p'.$$

and furthermore

$$\nu(w_\theta, s) = \nu(w_\theta, t), \qquad \nu(w_\theta, s+h) = \nu(w_\theta, t+h),$$

The last three equations together with (2.1) prove (2.9). □

The basic idea of the following definition is due to van den Boogaart (1999).

Definition 2.3.2 (Generic stationarity) Consider a stochastic process Z on D with covariance function C, semivariogram γ and mean m, and let $g : D \to T$ be a function onto an arbitrary set T. Then we define:

i) The process Z is *strongly generically stationary with respect to* g, if there exists a function $P_g : \mathcal{B}^d \times T \to \mathbb{R}$ such that

$$\forall\, s, t \in D \;\; \forall\, B \in \mathcal{B}^d : \quad P(Z_s \in B) = P_g(Z_s \in B \,|\, g(s)) = P_g(Z_t \in B \,|\, g(s)).$$

ii) $\Gamma \in \{C, \gamma\}$ is *generically stationary with respect to* g, if there exists a function $\Gamma_g : \mathbb{R}^d \times T \to \mathbb{R}$ such that

$$\forall\, s, t \in D : \quad \Gamma(s,t) = \Gamma_g(t - s \,|\, g(s), g(t)).$$

iii) If there exists a function $E_g : D \times T \to \mathbb{R}$ such that

$$\forall\, s, t \in D : \quad m(s) = E_g(Z_s \,|\, g(s)) = E_g(Z_t \,|\, g(s)),$$

then m is *generically stationary with respect to* g, and Z is *first-order generically stationary with respect to* g.

iv) Z is (*second-order* or *weakly*) *generically stationary*, if m and C are, and the process is *intrinsically generically stationary*, if m and γ are generically stationary.

v) The functions P_g, Γ_g and E_g are called *influence laws of generic stationarity*, and g an *influence function*.

Remark 2.3.3 Consider a process that is generically stationary with respect to local geology, and let $s, t \in D$ be two arbitrary points. Then generic stationarity says that the distribution laws of Z_s and Z_t are or become the same, if local geology around s and t are the same or are "forced" to be the same. That is, generic stationarity assumes that there is something like a law of nature that determines the distribution of a random variable given the local geology g. Depending on how we choose the function g, generic stationarity becomes a triviality or an instrument that describes how a distribution law is determined by the environment.

Theorem 2.3.4 (Stationarity and generic stationarity) For a stochastic process Z on D, the following conditions are equivalent:

i) Z is stationary.

ii) Z is generically stationary with respect to a constant mapping on D.

iii) Z is generically stationary with respect to every mapping on D.

Proof: iii) $\Rightarrow$ ii) is trivial.

ii) $\Rightarrow$ i): If Z is generically stationary with respect to a constant mapping $g : D \to \mathbb{R}$, then C_g in Definition 2.3.2 actually does not depend on $g(s)$ and $g(s+h)$, and thus C_g and C depend on h only.

i) $\Rightarrow$ iii) is trivial. $\square$

Remark 2.3.5 Consider a generically stationary stochastic process Z with influence function g. g can take two extreme cases:

i) g is constant. Then Z is stationary in the usual sense.

ii) $g = \mathrm{id}_D : s \mapsto s$. Then Z is an arbitrary process, there is no condition on its mean or covariance function.

One could say that a constant influence function gives no information on local geology, while identity contains complete knowledge of local geology and hence explains arbitrary spatial variation of the process' first- and second-order structures.

"Between" these two extremes, there exists a broad variety of meaningful influence functions and laws, as we saw in Example 2.3.1.

2.4 Kriging

In this section, only a brief review of the most important kriging techniques is given. We refer to Cressie (1993) for further details and proofs.

Suppose that the process Z on D can be modeled as

$$Z_s = Y_s + \beta^T f(s) \qquad \text{for all } s \in D, \tag{2.11}$$

where $Y = (Y_s)_{s\in D}$ is a zero-mean stationary random field on D, $f : D \to \mathbb{R}^k$, $k \geq 1$, is a deterministic function and $\beta \in \mathbb{R}^k$ a parameter vector. In this model, $\beta^T f(s)$ represents a deterministic trend, to which randomness is included by adding Y.

Note that (2.11) defines a generalized linear model with residuals Y correlated according to a stationary covariogram.

For simplification, we assume $f_1(s) = 1$ for all $s \in D$, i. e. a constant overall mean β_1 (or in other words, an intercept) is incorporated into the model.

Suppose that we observe $Z_{s_1}, \ldots, Z_{s_n}$, $s_1, \ldots, s_n \in D$. Write $Z_{(n)} = (Z_{s_1}, \ldots, Z_{s_n})^T$. We wish to predict Z_{s_0}, $s_0 \in D$, by a *linear predictor*

$$\widehat{Z}_{s_0} = \lambda^T Z_{(n)}, \qquad \lambda \in \mathbb{R}^n.$$

The *mean squared error* of this predictor is defined by

$$\sigma^2(s_0) = \mathrm{E}(Z_{s_0} - \widehat{Z}_{s_0})^2.$$

If there exists a linear predictor that minimizes the mean squared error among all linear predictors, it is called a *best linear predictor.* A predictor is said to be *unbiased*, if

$$\mathrm{E}(\widehat{Z}_{s_0}) = \mathrm{E}(Z_{s_0}).$$

Now let γ denote the semivariogram of Z. Write $\gamma_0 = (\gamma(s_0, s_1), \ldots, \gamma(s_0, s_n))^T$.

Theorem 2.4.1 Suppose that the semivariogram matrix $\Gamma = (\gamma(s_i, s_j))_{i,j=1,\ldots,n}$ is regular, $F = (f(s_i)^T)_{i=1,\ldots,n} \in \mathbb{R}^{n\times k}$ is of full rank, and $f(s) \in \mathrm{Im}(F^T)$.

Then a best linear unbiased predictor for Z_{s_0} is $Z^*_{s_0} = \lambda^T Z_{(n)}$, where $\lambda \in \mathbb{R}^n$ is given by

$$\begin{pmatrix} \Gamma & F \\ F^T & 0_{n\times n} \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \end{pmatrix} = \begin{pmatrix} \gamma_0 \\ f(s_0) \end{pmatrix}. \tag{2.12}$$

Furthermore, the mean squared error of $Z^*_{s_0}$ is

$$\sigma^2(s_0) = (\lambda^T, \mu^T) \cdot (\gamma_0^T, f(s_0)^T)^T. \tag{2.13}$$

Proof: Cf. Cressie (1993).

Remark 2.4.2 i) In geostatistics, best linear unbiased prediction is called *kriging*.
ii) If the mean of Z is an unknown constant, we can put $k = 1$ and $f(s) = 1$ for all $s \in D$. Best linear unbiased prediction is then called *ordinary kriging*.
iii) For $f \not\equiv 1$, best linear unbiased prediction is called *universal kriging*.
iv) If the mean of Z is a known constant, best linear unbiased prediction is called *simple kriging*.
v) Kriging for a process $(\mathbf{1}_{]-\infty,b]} \circ Z_s)_{s\in D}$ is called *indicator kriging*.

Remark 2.4.3 A process Z of the form (2.11) is generically stationary with respect to $g = \beta^T f$.

2.5 Estimation of Semivariogram Parameters

Suppose that we observe realizations $z_1 = Z_{s_1}(\omega), \ldots, z_n = Z_{s_n}(\omega)$, $\omega \in \Omega$, of a stochastic process Z on $D \subset \mathbb{R}^d$ at n locations $s_1, \ldots, s_n \in D$. We estimate the semivariogram γ of Z at (s_i, s_j) by

$$\widetilde{\gamma}_{ij} = (z_i - z_j)^2, \qquad i, j \in \{1, \ldots, n\}.$$

The matrix $\Gamma = (\widetilde{\gamma}_{ij})_{i,j=1,\ldots,n}$ is called an *empirical semivariogram* based on the data $z = (z_1, \ldots, z_n)$. We want to find a *semivariogram estimator* γ^* for γ that fits the empirical semivariogram Γ “best”.

Definition 2.5.1 (Mean squared error) Let $\widehat{\gamma}$ be an estimator for the semivariogram γ of Z. The *mean squared error* of $\widehat{\gamma}$ is defined to be

$$\operatorname{mse}(\widehat{\gamma}) = \sum_{i=1}^{n} \sum_{i=1}^{n} (\widehat{\gamma}(s_i, s_j) - \widetilde{\gamma}_{ij})^2. \tag{2.14}$$

Remark 2.5.2 We consider a semivariogram model $(\gamma_\theta)_{\theta\in\Theta}$ and want to minimize the mean squared error $\operatorname{mse}(\gamma_\theta)$ by choosing an appropriate parameter vector $\theta \in \Theta$. If such an optimal parameter $\theta^* \in \Theta$ exists, it is called a *best parameter estimator* in Θ and γ_{θ^*} a *best semivariogram* in $(\gamma_\theta)_{\theta\in\Theta}$. Searching for such an estimator is referred to as *fitting semivariogram models* or just *fitting semivariograms*. In this situation, the function

$$\operatorname{mse} : \Theta \to \mathbb{R}, \qquad \operatorname{mse}(\theta) = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{i=1}^{n} \big(\gamma_\theta(s_i, s_j) - (z_i - z_j)^2\big)^2 \tag{2.15}$$

is called the *mean squared error function* of the semivariogram model $(\gamma_\theta)_{\theta\in\Theta}$ with respect to the data $z \in \mathbb{R}^n$.

Remark 2.5.3 (Covariogram fitting) For a non-geo-statistician, a more straightforward way of modeling the second-order structure of a stochastic process could consist of fitting a covariogram model $(C_\theta)_{\theta\in\Theta}$. The “canonical” mean squared error corresponding to this problem is

$$\sum_{i=1}^{n} \sum_{j=1}^{n} (C_\theta(s_i, s_j) - (z_i - m(s_i))(z_j - m(s_j)))^2, \qquad \theta \in \Theta.$$

Unfortunately, this requires knowledge of the expected value m of Z, which in general cannot be assumed even if m is constant. This problem does not arise in the case of semivariogram fitting using (2.15).

Remark 2.5.4 i) Suppose that γ_θ is linear in $\theta \in \Theta$. Then

$$\text{minimize } \operatorname{mse}(\theta) \tag{2.16}$$

is a quadratic optimization problem. A comprehensive theory and efficient algorithms for this kind of problems exist. See e. g. Krekó (1974) and Nocedal and Wright (1999).

ii) In general, (2.16) defines a non-linear optimization problem. The target function mse may possess local minima. (See Section 3.2.5 for a numerical example.) Computational aspects are discussed in Section 3.2.5.

2.6 Modelling in the Presence of Global Trend

2.6.1 Introduction

We return to the situation of Section 2.4, considering a process Z that can be modeled as

$$Z_s = Y_s + \beta^T f(s) \qquad \text{for all } s \in D, \tag{2.17}$$

where $Y = (Y_s)_{s \in D}$ is a zero-mean stationary process on D, $f : D \to \mathbb{R}^k$, $k \geq 1$, a deterministic function and $\beta \in \mathbb{R}^k$ a parameter vector.

Again we assume $f_1(s) = 1$ for all $s \in D$, i. e. a constant overall mean β_1 is incorporated into the model.

Let γ_Z denote the semivariogram of Z, γ_Y the semivariogram of Y. There are two ways of predicting Z_{s_0}, $s_0 \in D$, based on observations $Z_{s_1}, \ldots, Z_{s_n}$, $s_1, \ldots, s_n \in D$ using kriging techniques:

i) If $\gamma_Z(s_0, s_i)$, $i = 1, \ldots, n$, is known, we can apply universal kriging.

ii) If $\gamma_Y(s_0, s_i)$, $i = 1, \ldots, n$, and β are known, then $Y_{s_1}, \ldots, Y_{s_n}$ are known, too, we can predict Y_{s_0} through simple kriging, and the trend value at s_0 is given by $\beta^T f(s_0)$. Adding the predictor for Y_{s_0} to $\beta^T f(s_0)$, we get an unbiased predictor for Z_{s_0}.

In practice, γ_Z or (γ_Y, β) have to be estimated. In situation ii), we can proceed in the following way (Goovaerts 1997):

(1) Assume that $Y_{s_0}, \ldots, Y_{s_n}$ are independent and identically distributed. Estimate β by $\widehat{\beta}$ by fitting the linear model (2.17) to the observed data.

(2) Assume that the residuals of the linear model are (spatially) correlated and stationary. Fit a semivariogram model to the residuals of the linear model.

(3) Predict the trend component of Z_{s_0} as $\widehat{\beta}^T f(s_0)$, and predict $\widehat{Y}_{s_0}$ using the semivariogram fitted in step (1) for simple kriging of the linear model's residuals. Estimate Z_{s_0} by the sum of both predictors.

Writing down explicitely the assumptions made in steps (1) and (2), it becomes obvious that there is a contradiction. A work-around could consist of repeating step (1), this time using a covariogram corresponding to the estimated semivariogram[3] for fitting a generalized linear model with correlated errors, and proceeding with steps (2) and (3) in the same way as before.

However, there is no guarantee that subsequent repetitions of this procedure lead to converging estimates for β and the semivariogram parameters, which should be a minimum requirement for "believing" in this method. Furthermore, there remains the problem that the residuals of the linear model may be instationary, which enters in conflict with the assumption made in step (3) (van den Boogaart 2000).

[3]When using semivariograms and covariograms induced by weight functions, there is a canonical correspondence.

A more promising approach follows alternative i) and consists of estimating γ_Z taking into account the presence of global trend, and doing universal kriging with the obtained estimator. This strategy was studied by van den Boogaart (2000) and will be presented, implemented (Section 3.3.4) and applied (Section 4.3) in the present work.

2.6.2 Estimation of Semivariogram Parameters

The following definition presents an analogue of the classical empirical semivariogram estimator for the model with trend.

Definition 2.6.1 (Empirical semivariogram model) Let $0 = h_0 < \ldots < h_p = \infty$, $\Theta = [0, \infty[^p$. Then the function γ_θ, $\theta \in \Theta$, defined by

$$\gamma_\theta(s,t) = \begin{cases} 0, & \text{if } s = t, \\ \theta_i, & \text{if } h_{i-1} < \|t-s\| \leq h_i \end{cases} \tag{2.18}$$

is called an *empirical semivariogram* with breaks at $h_1, \ldots, h_{p-1} \in \mathbb{R}^+$, and the collection $(\gamma_\theta)_{\theta\in\Theta}$ an *empirical semivariogram model.*

Remark 2.6.2 Define

$$P = I_n - F(F^TF)^{-1}F^T$$

and write $Z_{(n)} = (Z_{s_1}, \ldots, Z_{s_n})^T$. Then P is symmetric and idempotent, and it holds

$$\mathrm{E}(PZ_{(n)}Z_{(n)}^TP^T) = -P\Gamma P^T = PCP^T. \tag{2.19}$$

Proof: Symmetry and idempotence of P are easily verified. Furthermore, it holds $PF = 0_{n\times n}$. Hence,

$$PZ_{(n)} = P(Y_{(n)} + F\beta) = PY_{(n)}$$

and

$$\mathrm{E}(PZ_{(n)}) = P\,\mathrm{E}(Y_{(n)}) = P0_n = 0_n. \tag{2.20}$$

Furthermore, $\mathbb{1}_n = (1, \ldots, 1)^T \in \mathrm{Im}(F)$ because $f_1 \equiv 1$, and hence $P\mathbb{1}_{n\times n} = P\mathbb{1}_n\mathbb{1}_n^T = 0_{n\times n}$. Then, putting $c_0 = C(s,s)$ independently of $s \in D$ and using $\Gamma = c_0 - C$, it follows

$$P\Gamma P = P(c_0\mathbb{1}_{n\times n} - C)P = -PCP.$$

This implies

$$\mathrm{Cov}(PZ_{(n)}, PZ_{(n)}) = P\,\mathrm{Cov}(Y_{(n)}, Y_{(n)})\,P = PCP = -P\Gamma P,$$

and finally the hypothesis follows by applying (2.20):

$$\begin{aligned} \mathrm{E}(PZ_{(n)}Z_{(n)}^TP^T) &= \mathrm{Cov}(PZ_{(n)}, PZ_{(n)}) + \underbrace{(\mathrm{E}PZ_{(n)})(\mathrm{E}PZ_{(n)})^T}_{=0_{n\times n}} \\ &= PCP \;=\; -P\Gamma P. \end{aligned}$$

□

Remark 2.6.3 We observe a realization $z = Z_{(n)}(\omega)$, $\omega \in \Omega$, and know the matrix P, and we are looking for a semivariogram γ_θ, $\theta \in \Theta$, that "fits" the true semivariogram matrix Γ best. Remark 2.6.2 motivates the following concept of mean squared error for the model with linear trend; it was proposed by van den Boogaart (2000).

Definition 2.6.4 (Restricted mean squared error) Let $\widehat{\gamma}$ be an estimator for the semivariogram of Z, Let z denote a realization of $Z_{(n)}$, and use the notation introduced above. Then by

$$\operatorname{rmse}(\widehat{\gamma}) = \|Pzz^TP + P\Gamma P\|^2/n^2 \qquad \text{with } \|(a_{ij})_{i,j=1,\dots,n}\|^2 = \sum_{i,j=1}^{n} a_{ij}^2. \tag{2.21}$$

we define a mean squared error in a projected linear space, which we propose to call the *restricted mean squared error* of $\widehat{\gamma}$.

Remark 2.6.5 i) Remark 2.5.2 applies analogously.
ii) In contrast to the concept of mean squared error introduced in 2.5, the restricted mean squared error can be used for fitting covariogram models even without knowing the mean of Z; we just have to replace $P\Gamma P$ by $-PCP$, which in fact is identical (see Remark 2.6.2).

Remark 2.6.6 In general, the mean squared error (2.14) and the restricted mean squared error (2.21) in presence of constant trend are not identical.

Proof: We study two linear mappings $\widetilde{P}, \widetilde{\Phi} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$, defined by

$$\widetilde{P}(A) = PAP, \qquad \widetilde{\Phi}(A) = (\tfrac{1}{2}a_{ii} + \tfrac{1}{2}a_{jj} - a_{ij})_{i,j=1,\dots,n}$$

for $A = (a_{ij})_{i,j} \in \mathbb{R}^{n\times n}$. Using the notation introduced earlier in this section, we can express both concepts of means squared error in terms of $\widetilde{P}$ and $\widetilde{\Phi}$:

$$\operatorname{mse}(\gamma) = \|\widetilde{\Phi}(zz^T - C)\|^2/n^2, \qquad \operatorname{rmse}(\gamma) = \|\widetilde{P}(zz^T - C)\|^2/n^2.$$

Assuming $f \equiv 1$, we have

$$P = I_n - \tfrac{1}{n}\mathbb{1}_{n\times n}.$$

Then, if A is symmetric, we get $\mathbb{1}_{n\times n}A = A\mathbb{1}_{n\times n}$, and

$$\begin{aligned}\widetilde{P}(A) &= (I_n - \tfrac{1}{n}\mathbb{1}_{n\times n})\,A\,(I_n - \tfrac{1}{n}\mathbb{1}_{n\times n}) \\ &= A - \tfrac{2}{n}\mathbb{1}_{n\times n}A + \tfrac{1}{n^2}\underbrace{\mathbb{1}_{n\times n}\mathbb{1}_{n\times n}}_{n\mathbb{1}_{n\times n}}A \\ &= A - \tfrac{1}{n}\mathbb{1}_{n\times n}A.\end{aligned}$$

We choose the tridiagonal matrix

$$T = \begin{pmatrix} 1 & 1 & & 0 \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ 0 & & 1 & 1 \end{pmatrix}$$

and compare $\operatorname{mse}(T)$ with $\operatorname{rmse}(T)$.[4] It can easily be seen that

$$\widetilde{P}(T) = \frac{1}{n}\begin{pmatrix} n-2 & n-3 & -3 & \cdots & -3 & -2 \\ n-2 & n-3 & \ddots & \ddots & \vdots & \vdots \\ -2 & n-3 & \ddots & \ddots & -3 & \vdots \\ \vdots & -3 & \ddots & \ddots & n-3 & -2 \\ \vdots & \vdots & \ddots & \ddots & n-3 & n-2 \\ -2 & -3 & \cdots & -3 & n-3 & n-2 \end{pmatrix},$$

[4]The reader might wonder why the simpler matrix I_n is not used here. The reason is that then the result suggests a very simple relationship between mse and rmse, which would be misleading.

which yields

$$\|\widetilde{P}(T)\|^2 = 3 - \frac{11}{n} + \frac{10}{n^2}.$$

Turning to the model without trend, first we observe that for any matrix A with constant diagonal elements $a_{11} = \ldots = a_{nn} = a$, it holds

$$\widetilde{\Phi}(A) = \widetilde{\Phi}(A - aI_n) + \widetilde{\Phi}(aI_n) = (aI_n - A) + a\widetilde{\Phi}(I_n),$$

and using $\widetilde{\Phi}(\mathbb{1}_{n\times n}) = 0_{n\times n}$, we obtain

$$\widetilde{\Phi}(I_n) = \widetilde{\Phi}(I_n - \mathbb{1}_{n\times n}) = \mathbb{1}_{n\times n} - I_n.$$

Thus, we see that

$$\widetilde{\Phi}(A) = aI_n - A + a\mathbb{1}_{n\times n} - aI_n = a\mathbb{1}_{n\times n} - A.$$

Hence, the tridiagonal matrix T yields

$$\widetilde{\Phi}(T) = \mathbb{1}_{n\times n} - T,$$

and, counting the non-zero elements of $\widetilde{\Phi}(T)$,

$$\|\widetilde{\Phi}(T)\|^2 = n^2 - n - 2(n-1) = n^2 - 3n + 2.$$

We observe that $\|\widetilde{\Phi}(T)\|^2 \neq \|\widetilde{P}(T)\|^2$ (and that there is apparently no simple relation between both terms). $\square$

Chapter 3

Geostatistics and GIS

The present chapter treats issues concerning the implementation of geostatistical techniques and of a GIS interface. First, an introductory part (Section 3.1) leads to the selection of software platforms to be used in this work. Section 3.2 then consists of a presentation of mathematical methods that form a theoretical background for implementation (Section 3.3).

Sections 3.3 and 3.4 deal with the implementation of geostatistical objects and functions and of a GIS interface.

3.1 Introduction

3.1.1 Integration of GIS and Data Analysis Tools

Basically we can distinguish three stages of integration of tools for geostatistical data analysis or other purposes and Geographical Information Systems (GIS), as shown in figure 3.1.

A very common practice today is the loose coupling scenario, which consists in exporting data from the GIS, performing data analysis tasks independently within the tool, and finally importing results into the GIS. However, this may cause problems due to incompatibility of file formats, and meta-information such as information on the semantics of data and data types have to be handled by the user and may get lost during multiple analyses and export-import processes.

In a second scenario, GIS and tool still are independent software, but both possess communication interfaces that allow the user to pass data to and execute analyses within the data analysis tool while he is working in a GIS environment (or vice versa). This linkage may consist in running the tool within the GIS, in executing single tool commands from within the GIS, or, potentially, in Object Linking and Embedding (OLE). The data analysis environment R, for example, can be run within the GRASS GIS shell, and GRASS' data base then can be accessed after dynamically loading compiled GIS library functions into the R environment (Bivand 1999). Furthermore, R offers the possibility of executing commands by calling or being called by a Dynamic Link Library (DLL) (R Development Core Team 2000), and the GIS software ArcView, for example, allows such calls to be done (ESRI 1996). However, communication by means of DLL calls always requires a certain amount of interface programming at a lower level (usually in C).

The highest degree of integration is achieved when the tool is fully integrated within a GIS. This is, without any doubt, the most comfortable situation for users who want to apply statistical standard methods within a GIS without having to familiarize themselves with a specialized environment. However, a GIS will not be able to offer the broad spectrum of methods available within R or S-PLUS, for example. Probably the first GIS that fully integrated a rather comprehensive set of geostatistical methods is ESRI's ArcGIS 8.1 Geostatistical Analyst released in May 2001 (ESRI 2001).

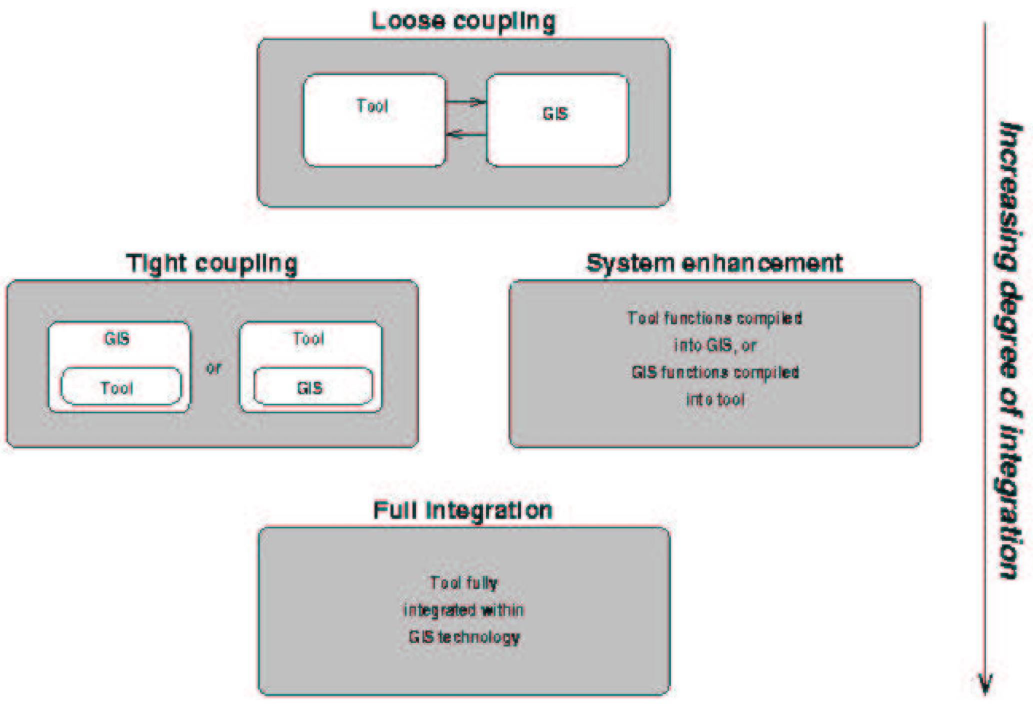

Loose coupling
Tool
GIS
Tight coupling
GIS
Tool
or
Tool
GIS
System enhancement
Tool functions compiled into GIS, or GIS functions compiled into tool
Full integration
Tool fully integrated within GIS technology
Increasing degree of integration

Figure 3.1: Levels of integration of data analysis tools and GIS (taken from Bivand and Neteler (2000)).

Now, which level of integration should be aimed at and can be reached in this work?

What we need is a flexible programming environment that allows us to implement, test and apply modeling techniques for generically stationary geostatistics. Efficient mathematical algorithms and a variety of data analytical methods should be available, and mathematical objects should be easy to handle. Furthermore, we do not want to restrict the applicability of the code to a specific GIS.

Therefore, the goal will be to implement geostatistical functions and objects fully within a general-purpose data analysis environment, and to provide direct access to this functionality from an exemplarily chosen GIS in a tight-coupling scenario.

3.1.2 Choice of the Platform: R 1.2.2 and ArcView 3.1

There exists a great number of mathematical and statistical programming environments that could be used for implementing geostatistical methods. For this work, R 1.2.2 was chosen, a "language and environment for statistical computing and graphics" as it defines itself. R offers a simple and effective programming language that includes conditionals, loops and user defined functions, and there exists a simple kind of object-oriented design. There are many operators and functions for handling mathematical objects such as matrices, and a great variety of statistical models and methods is also available, among them linear models, clustering and time-series analysis. Furthermore, so-called *packages*, i. e. bundles of functions dealing with a special problem, are available for free and extend the functionality of R.

The R environment is an open-source software freely available for download at `http://www.r-project.org`. Open-source software is not only cheaper than commercial alternatives; the main advantage of using it is related to the efficiency of problem-solving within a community ("bazaar") of users. Among them, at least one participant will already have met and solved one's problem, if the community is large enough. Furthermore, full access to source code makes it possible to find out details of the implementation that are not documented in the manuals. (Bivand 1999)

R differs very little from the language S and its derivative S-PLUS. The main differences concern the placement of objects in memory and scoping rules. However, only small changes may be necessary when transferring code between both systems.

As GIS platform, the commercial software ArcView 3.1 was chosen, because it is being used at many universities (including Freiberg University), in environmental agencies and consultancies. Furthermore, the author was already quite familiar with ArcView when starting this work. Unfortunately, it is a commercial software. However, anticipating what will be seen later in Section 3.4, a relatively small amount of code had to be developed in order to create a basic user interface within ArcView's object-oriented programming language AVENUE, so it should be quite easy to write similar interfaces for other GIS.[1] While it is rather easy to implement windows-based applications within AVENUE, it is not suited for handling mathematical objects.

In total, about 100 KB of R code, 30 KB of C code and 30 KB of AVENUE source code were produced. The code was developed in R 1.2.2, Microsoft Visual C++ 5.0 and ArcView 3.1 within a Microsoft Windows 2000 5.0 / NT 4.0 client–server environment at the Department of Mathematics and Computer Science, Freiberg University for Mining and Technology. The server has two Pentium II processors and 1 GB of memory, and the client that usually was used has a Pentium II family processor and 64 MB of memory.

Source code and binaries can be obtained from the author (e-mail: `ali@proforma.de`).

3.2 Mathematical Background

3.2.1 Preliminaries

First we recall some definitions and remarks from numerical linear algebra that will be needed in this chapter.

Definition and Remark 3.2.1 (Matrix norm) Let $\|\cdot\|$ be an arbitrary norm on $\mathbb{R}^n$. On the linear space $\mathbb{R}^{n\times n}$ we define the mapping

$$\|\cdot\| : \mathbb{R}^{n\times n} \to \mathbb{R}, \qquad \|A\| := \sup_{x\neq 0} \frac{\|Ax\|}{\|x\|}$$

and call it the *matrix norm induced by* the vector norm $\|\cdot\|$.

The induced matrix norm is a norm, and it holds

$$\|Ax\| \leq \|A\|\|x\|, \qquad \|AB\| \leq \|A\|\|B\|$$

for all $A, B \in \mathbb{R}^{n\times n}$ and $x \in \mathbb{R}^n$.

Definition 3.2.2 (Spectral radius) Let $\lambda_1, \ldots, \lambda_n \in \mathbb{C}$ be the eigenvalues of $A \in \mathbb{R}^{n\times n}$. Then the *spectral radius* of A is defined to be

$$\rho(A) := \max_{i=1,\ldots,n} |\lambda_i|.$$

Remark 3.2.3 (Spectral norm) The matrix norm $\|\cdot\|_2$ induced by the Euclidian norm on $\mathbb{R}^n$ is called the *spectral norm*. For all $A \in \mathbb{R}^{n\times n}$ it holds

$$\|A\|_2 = \rho(A^T A)^{1/2}.$$

[1] In fact GRASS GIS should be an R programmer's favourite GIS, for two reasons: First, R can be run within GRASS, and there exists a contributed R package that allows addressing GRASS objects with the R language; and second, GRASS is open source software, just as R. See Bivand (1999) and Bivand and Neteler (2000) for more details on integrating GRASS and R.

If A is symmetric, then $\|A\|_2 = \rho(A)$.

(Proof: See Werner (1992).)

Question 3.2.4 Consider a system of linear equations $Ax = b$, where $A \in \mathrm{GL}(n)$ is a regular matrix and $b \in \mathbb{R}^n$. In how far does an error in the data A and b affect the solution?

Definition 3.2.5 (Condition number) The *condition number* of a regular matrix $A \in \mathrm{GL}(n)$ with respect to the matrix norm $\|\cdot\|$ is defined by

$$\mathrm{cond}A := \|A\|\|A^{-1}\|.$$

Remark 3.2.6 Suppose $A \in \mathrm{GL}(n)$. Then it holds $\mathrm{cond}A \geq \mathrm{cond}I_n = 1$. If A is symmetric, the condition number of A with respect to the spectral norm $\|\cdot\| = \rho(\cdot)$ is

$$\mathrm{cond}_2A = \rho(A)\rho(A^{-1}) = |\lambda_{\max}|/|\lambda_{\min}|,$$

where $\lambda_{\max}$ and $\lambda_{\min}$ denote eigenvalues of A with greatest and smallest absolute value, respectively.

Theorem 3.2.7 (Singular-value decomposition) For an arbitrary matrix $A \in \mathbb{R}^{n\times p}$ with $n \geq p$ and rank r, there exist matrices $U \in \mathbb{R}^{n\times p}$ and $V \in \mathbb{R}^{p\times p}$ with $U^TU = I_p$ and $V^TV = I_n$ and real numbers $\sigma_1 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_n = 0$ such that

$$A = UDV^T, \tag{3.1}$$

where $D = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$. This representation is called the *singular-value decomposition* of A.

(Proof: See Werner (1992).)

Remark 3.2.8 (Singular values) In the situation of Theorem 3.2.7, $\sigma_1^2, \ldots, \sigma_r^2$ are the (positive) eigenvalues of A^TA (and of AA^T, too). $\sigma_1, \ldots, \sigma_r$ are called the *singular values* of A. If A is symmetric and regular, then $\sigma_1, \ldots, \sigma_n$ are the eigenvalues of A, and we have

$$\mathrm{cond}_2A = \sigma_1/\sigma_n.$$

3.2.2 Generating Random Data

It is common to use random data with certain well-known properties in order to test statistical techniques (see Chapter 4). These simulated datasets can be generated quickly on demand and possess exactly the desired mean and covariance structure unlike real-world data gathered at high cost. In this section, we present some background for creating random datasets corresponding to a Gaussian process. We refer to Gentle (1998) and Cressie (1993) for further details.

We wish to generate numbers that can "practically" not be distinguished from realizations of normal random variables $Z_1, \ldots, Z_n$ with mean $m \in \mathbb{R}^n$ and covariance matrix $C \in \mathbb{R}^{n\times n}$. These numbers are called a *simulation* of $Z_1, \ldots, Z_n$. (See Remark 3.2.15 and Gentle (1998) for more precise quality criteria.)

Remark 3.2.9 Let $S_1, \ldots, S_n$ be independent standard normal random variables, i. e. $S = (S_1, \ldots, S_n)^T \equiv N(0, I_n)$. Then for arbitrary $m \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n\times n}$, $Z = m + AS$ is normally distributed with mean m and covariance matrix AA^T. Therefore we wish to decompose a covariance matrix C as $C = AA^T$, simulate S, and derive a simulation of Z.

Definition and Remark 3.2.10 A symmetric matrix $C \in \mathbb{R}^{n\times n}$ is *positive definite* if for all $x \in \mathbb{R}^n \setminus \{0\}$, $x^TCx > 0$. It is *positive semidefinite*, if for all $x \in \mathbb{R}^n$, $x^TCx \geq 0$.

A symmetric matrix is positive definite (positive semidefinite) if and only if all its eigenvalues are positive (non-negative). The covariance matrix of a finite collection $(Z_1, \ldots, Z_n)^T$ of random variables is symmetric and positive semidefinite.

Decompositions of Covariance Matrices

We recall the following results (Werner 1992):

Corollary 3.2.11 (Singular-value decomposition) The singular-value decomposition of an arbitrary symmetric matrix $C \in \mathbb{R}^{n\times n}$ of rank r is of the form

$$C = UDU^T = (U\sqrt{D})(U\sqrt{D})^T,$$

where $D = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$, $\sigma_1 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_n = 0$, $U \in \mathbb{R}^{n\times n}$ is orthogonal and $\sqrt{D} = \operatorname{diag}(\sqrt{\sigma_1}, \ldots, \sqrt{\sigma_n})$.

Theorem 3.2.12 (Cholesky decomposition) Let $C \in \mathbb{R}^{n\times n}$ be symmetric and positive definite. Then there exists a unique lower triangular matrix $L \in \mathbb{R}^{n\times n}$ with positive diagonal elements and $C = LL^T$. This decomposition is called the *Cholesky decomposition* of C.

Remark 3.2.13 i) A standard algorithm for computing the Cholesky decomposition as presented by Werner (1992) needs $\sim \frac{1}{3}n^3$ floating point operations. For the diagonal elements of L and C, it holds $l_{kk} \leq \sqrt{c_{kk}}$, $k = 1, \ldots, n$. This has a stabilizing effect on the algorithm (Werner 1992).

ii) When possible, the Cholesky decomposition should be preferred to the singular-value decomposition of a positive definite matrix. However, it will cause an error if the matrix is not "numerically positive definite". In this case, the singular-value decomposition can be used.

Random Numbers

For the discussion of random numbers and their generation, we refer to Gentle (1998). Here we just give a few remarks of general character and concerning generators available in R.

Remark 3.2.14 i) Random numbers are computed deterministically from a finite number k of predecessors or, at the beginning of the sequence, from k starting values, called *seed*. These numbers belong to the finite set of numbers that can be represented within a computer, so the sequence will inevitably repeat as soon as the same k values appear for a second time. The number of random numbers generated before the sequence starts to repeat, is called *cycle length* or *period* of the sequence.

ii) Most currently used generators of uniformly distributed random numbers are based on modulo operations with repsect to a huge prime number. Common periods are around 10^{18}, for example.

Remark 3.2.15 (Quality criteria) i) A large period is desirable.

ii) There shall be no geometric regularities. (This is in particular a problem of linear congruential generators, which produce subsequences $(x_i, \ldots, x_{i+k-1})$ forming a lattice in $\mathbb{R}^k$.)

iii) Random numbers should pass common statistical tests, among them goodness-of-fit tests of (possibly transformed) random numbers, and autocorrelation tests of different order.

Remark 3.2.16 (Random numbers in R) R uses by default a multiply-with-carry algorithm recommended by Marsaglia for generating uniform random numbers, and the Kinderman–Ramage generator for normal random numbers. R also offers a variety of other generators for the uniform and normal case; see the R help topics `RNG` and `RNGkind` for details and references.

3.2.3 Approximating Semivariograms with Quasi-Monte Carlo Integration

When we fit a semivariogram model or do kriging, a great number of semivariogram evaluations is needed (cf. equations (2.12), (2.15), (2.21)), which are generally computed by approximating the integral in (2.3) (or (2.4)), if a model of the elliptical class is used (cf. Definition 2.2.19). Quasi-Monte Carlo techniques, which use quasi-random nodes, possess good convergence rates and are often preferred to numerical methods, especially in higher-dimensional integration.

A brief introduction to quasi-Monte Carlo integration is presented here following Niederreiter (1992), and it is applied to the approximation of induced covariograms. For further details on quasi-Monte Carlo Methods and the approximation of integrals in general, we refer to Niederreiter (1992), Evans and Swartz (2000) and Press et al. (1992).

Quasi-Monte Carlo Integration

Let $f : I^d \to \mathbb{R}$ be an arbitrary Riemann-integrable function on the d-dimensional unit cube $I^d = [0,1]^d$, and consider the integration problem

$$I(f) := \int_{I^d} f(p)\,\mathrm{d}p. \tag{3.2}$$

(More general integration domains are considered in Remark 3.2.24.) We use the *quasi-Monte Carlo approximation*

$$\hat{I}_K(f) := \frac{1}{K}\sum_{i=1}^{K} f(p_i) \approx I(f)$$

with $p_1, \ldots, p_K \in I^d$. We want a sequence of nodes $(p_i)_{i\in\mathbb{N}} \subset I^d$ with

$$\lim_{K\to\infty} \hat{I}_K(f) \to I(f).$$

A sufficient condition for this to hold is that the sequence[2] $(p_i)_{i\in\mathbb{N}}$ is *uniformly distributed* in I^d in the sense of

$$\lim_{K\to\infty} \hat{I}_K(\mathbf{1}_J) = \lambda^d(J) \qquad \text{for all subintervals } J \text{ of } I^d.$$

This suggests that the nodes $p_1, \ldots, p_K$ should be "evenly distributed" over I^d. The irregularity of distribution or deviation from uniform distribution is measured using various notions of discrepancy. Here we only introduce the star discrepancy, because it appears in the error bound given later in Theorem 3.2.18.

Definition 3.2.17 The *star discrepancy* of the sequence of nodes $p_1, \ldots, p_K \in I^d$ is

$$D_K^*(p_1, \ldots, p_K) = \sup_{J\in\mathcal{I}^*} \left| \frac{1}{K}\sum_{i=1}^{K} \mathbf{1}_J(p_i) - \lambda^d(J) \right|,$$

where $\mathcal{I}^*$ denotes the collection of all subintervals of I^d of the form $\prod_{i=1}^{d}[0, u_i[$.

The following error bound is due to Proinov (1988); we cite it from Niederreiter (1992).

Theorem 3.2.18 If f is continuous on I^d, then for a finite sequence of nodes $p_1, \ldots, p_K \in I^d$, we have

$$|\hat{I}_K(f) - I(f)| \le 4\omega(f; D_K^*(p_1, \ldots, p_K)^{1/d}),$$

[2] In a more general setting, sequences of finite sequences $p^{(K)} = (p_i^{(K)})_{i=1,\ldots,K}$ can be considered, if the first K elements of $p^{(K+1)}$ are not equal to $p^{(K)}$.

where ω denotes the *modulus of continuity* of f,

$$\omega(f;t) = \sup_{\substack{u,v\in I^d \\ \|u-v\|_\infty \leq t}} |f(u) - f(v)|,$$

for $t \geq 0$ and $\|u\|_\infty = \max_{1\leq i\leq d} |u_i|$.

Remark 3.2.19 i) The more the integrand varies at small distance, the greater will be its modulus of continuity and hence the approximation error.

ii) The greater the discrepancy of a sequence of nodes is, the greater will be the approximation error. There exist error bounds that are linear in the star discrepancy (Niederreiter 1992). These are "best possible" error bounds in certain sense, which motivates the search for low-discrepancy sequences of quasi-random numbers.

Low-Discrepancy Sequences

Remark 3.2.20 Now we will take a look at the sequence of nodes used. In contrast to Monte Carlo methods, which are based on (pseudo-) random numbers that aim to reach a maximum degree of "randomness", quasi-Monte Carlo techniques use more evenly spaced quasi-random numbers that in fact have nothing to do with randomness but aim at minimizing discrepancy (figure 3.2.

Definition 3.2.21 (Low-discrepancy sequence) A sequence $(p_n)_{n\in\mathbb{N}}$ in $[0,1]^d$ is a *low-discrepancy sequence* if for all $N > 1$,

$$D^*_K(p_1, \ldots, p_K) \leq c(d)\frac{(\log K)^d}{K}, \tag{3.3}$$

where $c(d)$ depends only on the dimension d.

Remark 3.2.22 The Koksma-Hlawka inequality, which gives an error bound that is linear in the star discrepancy, shows that quasi-Monte Carlo integration with low-discrepancy sequences of nodes has an error bound of the order of $\mathcal{O}((\log N)^d N^{-1})$. Monte Carlo integration, using random nodes, has error bounds of the order of $\mathcal{O}(N^{-1/2})$ (Niederreiter 1992).

Definition 3.2.23 (Sobol sequence) We give a brief description of the generation of Sobol sequences following Cheng and Druzdzel (2000). For details on this and other low-discrepancy sequences and their implementation, we refer to Antonov and Saleev (1979), Niederreiter (1992), Bratley and Fox (1988), Press et al. (1992) and Evans and Swartz (2000).

The *Sobol sequence* $(X_n)_{n\in\mathbb{N}}$, $X_n = (x_n^1, \ldots x_n^w)^T$ in w dimensions consists of numbers between zero and one that are generated as binary fractions of length w bits from a sequence $(V_1, \ldots, V_w)$ of special binary fractions $V_i = (v_i^1, \ldots, v_i^w)^T \in \{0,1\}^w$ of length w bits. The numbers V_i are called *direction numbers.*

For the generation of direction numbers, we start with a primitive polynomial over the field with elements $\{0,1\}$. If the primitive polynomial of degree q in dimension j is

$$p_j(x) = x^q + a_1x^{q-1} + \ldots + a_{q-1}x + 1,$$

then the direction numbers in dimension j are generated using the q-term recurrence relation

$$v_i^j = a_1v_{i-1}^j \oplus a_2v_{i-2}^j \oplus \ldots \oplus a_{q-1}v_{i-q+1}^j \oplus (v_{i-q}^j/2^q),$$

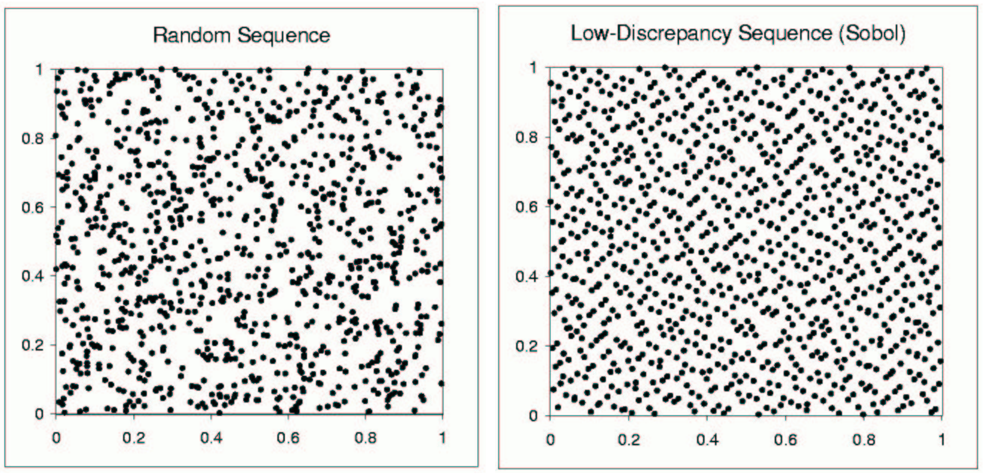


Figure 3.2: 1024 random and quasi-random points in two dimensions (from Cheng and Druzdzel (2000)).

where $i > q$. Here $\oplus$ denotes the bitwise XOR operation. The starting values $v_1^j \cdot 2^w, \ldots, v_q^j \cdot 2^w$ can be arbitrary odd integers smaller than $2, 2^2, \ldots,$ and 2^q, respectively. Then, writing $n = \sum_{i=0}^{w} b_i 2^i$, $b_i \in \{0,1\}$, the nth element of the Sobol sequence in dimension j is generated by

$$x_n^j = b_1 v_1^j \oplus b_2 v_2^j \oplus \ldots \oplus b_w v_w^j.$$

For each dimension, a different primitive polynomial should be used.

Antonov and Saleev (1979) proposed to use the bits of the Gray code of n to select direction numbers. This version in the implementation of Press et al. (1992) will be used in this work.

Approximation of Covariograms and Semivariograms

Remark 3.2.24 We want to approximate the integral of a continuous function $\tilde{f} := f \circ T^{-1}$ on a domain $\tilde{I}^d := TI^d$, where $T : x \mapsto a + Bx$, $a \in \mathbb{R}^d$, $B \in \mathrm{GL}(d)$, is an affine-linear transformation. The approximation problem

$$\hat{I}_K(\tilde{f}) = \frac{\lambda^d(TI)}{K} \sum_{i=1}^{K} f(T^{-1}p_i) \approx \int_{\tilde{I}^d} f(T^{-1}p)\,\mathrm{d}q = I(\tilde{f})$$

on $\tilde{I}^d$, $p_1, \ldots p_K \in \tilde{I}^d$, has the error bound

$$|\hat{I}_K(\tilde{f}) - I(\tilde{f})| \leq \lambda(\tilde{I})^d \cdot 4\omega(f; D_K^*(p_1 \ldots, p_K)^{1/d})$$

as a consequence of Theorem 3.2.18.

Corollary 3.2.25 (Approximation of covariograms) i) Consider a covariance function C induced by a weight function w with bounded support. We want to approximate (non-zero values of) $C(s,t)$, $s, t \in D$, $E = \mathbb{R}^d$, i. e. the integral of $f_{s,t}(p) = w(s,p)w(t,p)$ over a suitable interval $A \supset \operatorname{supp} f_{s,t} = \operatorname{supp} w(s,\cdot) \cap \operatorname{supp} w(t,\cdot)$. If we define an *overall modulus of continuity* of f to be

$$\omega(f;t) = \sup_{s,t \in D} \sup_{\substack{u,v \in A \\ \|u-v\|_\infty \leq t}} |f_{s,t}(u) - f_{s,t}(v)|, \qquad t \geq 0$$

and write $\widehat{C}_K(s,t) = \hat{I}_K(f_{s,t})$, we get the error bound

$$|\widehat{C}_K(s,t) - C(s,t)| \leq \lambda^d(A) \cdot 4\omega(f; D_K^*(p_1, \ldots, p_K)^{1/d}). \tag{3.4}$$

Proof: This is a consequence of Theorem 3.2.18 and Remark 3.2.24. □

Example 3.2.26 (Moduli of continuity: the elliptical case) Let w_θ^{ell} be an elliptical weight function on $D = E = \mathbb{R}^2$ with parameters $\theta = (\sigma^2, a, q, \tau)$, $\tau \in T$, corresponding to a kernel function w_τ^o (see Definition 2.2.19). We give the moduli of continuity of $f_{s,t}(p) = w_\theta^{\text{ell}}(s,p)w_\theta^{\text{ell}}(t,p)$ in some special cases. Let us assume $\sigma^2 = a = q = 1$. Write $\nu(w_\theta) = \int_{\mathbb{R}^2} w_\theta^{\text{ell}}(s,p)^2 \, \mathrm{d}p$, which is independent of s when $q = 1$.

i) *Simple weight function* $w_{(1,1,1)}^{\text{ind}}$: It holds

$$\omega_{(1,1,1)}^{ind}(t) = 1/\pi,$$

because of the discontinuity of w_θ^{ind} at $\|p\|_\infty = 1$ and $\nu(w_{(1,1,1)}^{\text{ind}}) = \pi$.

ii) *Linear weight function* w_θ^{lin}: Suppose $\|p\|_\infty = t$, $p \in \mathbb{R}^2$. Then:

$$\omega_{(1,1,1)}^{lin}(t) = (1 - f_{0,0}(p)) = (1-(1-t)^2)/\nu(w_{(1,1,1)}^{lin}) = (2t - t^2)/\nu(w_{(1,1,1)}^{lin}).$$

iii) *Piecewise linear weight function* $w_{(1,1,1,b)}^{\text{pwl}}$:

$$\omega_{(1,1,1,b)}^{pwl}(t) \le \min\left(1, \frac{2t}{1-b} - \frac{t^2}{(1-b)^2}\right)/\nu(w_{(1,1,1,b)}^{pwl}),$$

taking into account that, prior to normalization, $w_\theta^{\text{pwl}}(\mathbb{R}^2) = [0,1]$ and hence $f_{s,t}(\mathbb{R}^2) \subset [0,1]$, i. e. w_θ^{pwl} and $f_{s,t}$ do not vary by more than 1, and that the greatest variation at distances $\le (1-b)$ can be deduced by generalizing the formula for the linear weight function in i) with $a := 1 - b$.

iv) Now let w_θ be an isotropic elliptical weight function that is continuously differentiable on $\mathbb{R}^2$. Furthermore, suppose that $w_\theta(p) = w_\theta(0,p)$ decreases monotonically as $\|p\|_\infty$ grows. Then it holds

$$\omega(f_\theta; t) \le 2t \sup_{p\in\mathbb{R}^2} |w_\theta(hv)| \sup_{p\in\mathbb{R}^2} \|\operatorname{grad} w_\theta(p)\|_\infty.$$

Proof (draft): Maximum variation of $f_{\theta;s,t}$ is achieved when $s = t$, so, writing $f = f_{\theta;0,0}$ we have

$$\sup_{s,t\in\mathbb{R}^2} \sup_{\substack{u,v\in\mathbb{R}^2\\ \|u-v\|_\infty\le t}} |f_{\theta;s,t}(u) - f_{\theta;s,t}(v)| = \sup_{\substack{u,v\in\mathbb{R}^2\\ \|u-v\|_\infty\le t}} |f(u) - f(v)|.$$

Furthermore, due to the mean value theorem we get

$$|f(u) - f(v)| \le t \sup_{p\in\mathbb{R}^2} \|\operatorname{grad} f(p)\|_\infty \qquad \text{for } \|u-v\|_\infty \le t.$$

Now,

$$\operatorname{grad} f(p) = 2w_\theta(p)\operatorname{grad} w_\theta(p)$$

leads us to

$$t \sup_{p\in\mathbb{R}^2} \|\operatorname{grad} f(p)\|_\infty \le 2t \sup_{p\in\mathbb{R}^2} |w_\theta(p)| \sup_{p\in\mathbb{R}^2} \|\operatorname{grad} w_\theta(p)\|_\infty.$$

□

Remark 3.2.27 When approximating covariograms of elliptical weight functions, the following operations produce high cost, and the number of computations needed by different algorithms should hence be compared:

i) *Oracle calls*: When using a covariogram with strong boundaries around $A \subset D$, then for each node p we have to evaluate the term $\mathbf{1}_A(p)$. This may result in an oracle call, i. e. a call to an "expensive" function, such as a `SelectByPolygon` request from R to ArcView.

ii) *Weight function evaluations*: It should be our aim to avoid redundant evaluations that yield zero anyway.

iii) *Evaluations of* $\mathbf{1}_{\text{Ell}(s,\phi,a,q)}$ and transformations between polar and Euclidian coordinates: These evaluations are almost as expensive as evaluations of relatively simple weight functions, and they are in many cases avoidable.

We wish to know how the cost grows as the number of measurement points, N, and the number of nodes, K, increase. We will consider *infill asymptotics*, where a fixed domain is subsequently filled with measurement points, and *increasing-domain asymptotics*, where, roughly speaking, the density of points is preserved while spreading new measurement points out over an increasing area. Note that in both cases we will require the accuracy to remain unchanged, i. e. the number of nodes will have to increase.

We will now present two algorithms that compute the quasi-Monte Carlo approximation of covariograms induced by elliptical weight functions. Both algorithms support ordinary and strong boundaries (see Definition 2.2.25).

The first algorithm generates nodes when needed, minimizing the number of weight function evaluations by generating nodes in polar coordinates within one of the weight function supports. The second algorithm uses a different strategy: Nodes are created a priori, so the number of "oracle calls" will be minimized.

For simplicity, the first algorithm is only presented for the approximation of integrals on circles instead of ellipses.

Algorithm 3.2.28 (Integration on dissections of circles)

Input: $s_1, s_2 \in A \subset \mathbb{R}^2$, $a_1, a_2 > 0$,
$w : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ weight function with $\operatorname{supp} w(s_i, \cdot) \subset B^2(s_i, a_i/2)$, $i = 1, 2$,
K: number of nodes to create.

Output: $\hat{I} \approx \int_{\mathbb{R}^2} w(s_1,p)w(s_2,p)\mathbf{1}_A(p)\,\mathrm{d}p$.

```
01  if $\|s_2 - s_1\| \geq (a_1 + a_2)/2$ then return 0
02  $\hat{I} := 0$
03  generate quasi-random nodes $\widetilde{p}_1, \ldots, \widetilde{p}_K \in B^2(s_1, a_1/2)$   $(\widetilde{p}_i = (r_i, \varphi_i))$
04          in polar coordinates relative to $s_1$
05  $p_i := \widetilde{p}_i$ transformed to euclidian coordinates $(i = 1, \ldots, K)$
06  for $i := 1$ to $K$ do
07      if $p_i \in B^2(s_2, a_2/2)$ then                    // check for the integrand's support
08          if $p_i \in A$ then                                          // oracle call
09              $\hat{I} := \hat{I} + r \cdot \widetilde{w}(s_1, \widetilde{p}_i) w(s_2, p_i)$
10          end if
11      end if
12  end for
13  $\hat{I} := \hat{I} \cdot \lambda^2(B^2(s_1, a_1/2))/K$
14  return $\hat{I}$
```

For a sequence $s_1, \ldots, s_N$ of measurement points, call the algorithm for each pair (s_i, s_j) with $i < j$ and use the symmetry of the integral.

Remark 3.2.29 (Algorithm 3.2.28) i) If $A = \mathbb{R}^2$, the algorithm uses ordinary boundaries, otherwise strong boundaries.

ii) The algorithm performs $\mathcal{O}(N^2)$ oracle calls "$p_i \in A$", $\mathcal{O}(N^2)$ weight function evaluations and $\mathcal{O}(N^2)$ "ellipse tests" and coordinate transformations both with infill and fixed-domain asymptotics.

iii) In line 9, the factor r in the integrand is due to integration in polar coordinates. It causes difficulties when integrating on ellipses, because the radius then depends on φ, too. With respect to normalization of the integral in order to get an elliptical covariogram, see Remark 3.2.33 below.

iv) An important improvement can be made in Algorithm 3.2.28 by using adaptive sampling: After a first integration step with few nodes, find a sector of $B^2(s_1, a_1)$ that fully contains the dissection of both circles, and then integrate over this sector only.

v) Instead of using a fixed number of nodes, go on creating them until the desired error bound is reached ("sampling until").

vi) Integrate over the smaller circle, swapping s_1 and s_2 (and a_1, a_2) in line 03. The dissection of both circles forms a greater part of the smaller circle than of the greater one, so a higher percentage of nodes will fall into the dissection when generating them within the smaller circle.

Remark 3.2.30 (Reduction of oracle calls) We wish to reduce the number of oracle calls necessary for quasi-Monte Carlo integration. More precisely, from a computational point of view, we can distinguish between two concepts of oracle call, because instead of $\mathbf{1}_A(p)$, we will compute $\mathbf{1}_A(\{p_1, \dots, p_K\})$. So on one side, we have to execute *calls to the oracle* (routine, software,...), for example a GIS or R function, and on the other, *evaluate the oracle* function $\mathbf{1}_A(p_i)$ (in a mathematical sense) for each $i = 1, \dots, K$. Now we can describe the aim expressed in the beginning of this Remark in the following way: We wish to reduce both the number of calls to the oracle and the number of evaluations of the oracle function. Now we present a strategy that provides an important reduction of the number of calls to the oracle.

Definition 3.2.31 (A priori nodes) Consider integration problems $(I(f;i))_{i=1,\dots,N}$ for some function f. If we use the same nodes $p_1, \dots, p_K$ for each quasi-Monte Carlo approximation $\hat{I}_K(f;i)$, $i = 1, \dots, N$, then we call these nodes *a priori nodes.*

Algorithm 3.2.32 (Integration on dissections of ellipses using a priori nodes)

Input: (i) $(s_i, g_i, a_i, q_i)^T_{i=1,\dots,N} \subset A \times [0, \pi[\times]0, R] \times]0, 1]$ $(A \subset \mathbb{R}^2)$
(ii) F, pairwise disjoint sets $F_1, \dots, F_L$ with $\bigcup_i F_i = F \supset B(s_1, \dots, s_N; R)$ and $R := \max_i a_i/2$
(iii) quasi-random nodes $p_1, \dots, p_K \in B(s_1, \dots, s_N; R)$ and $0 = K_0 \leq \dots \leq K_L = K$ such that $p_{K_{i-1}+1}, \dots, p_{K_i} \in F_i$ for $i = 1, \dots, L$
(iv) $\lambda^2(F_1), \dots, \lambda^2(F_L)$
(v) a weight function w on $\mathbb{R}^2$ with $\operatorname{supp} w(s_i, \cdot) \subset \operatorname{Ell}(s_i, g_i, a_i/2, q_i)$, $i = 1, \dots, N$

Output: $(\hat{I}_{ij})_{i,j=1,\dots,N}$,
$\hat{I} \approx \int_{\mathbb{R}^2} w(s_i, p) w(s_j, p) \mathbf{1}_A(p)\, \mathrm{d}p$

```
01  $\hat{I} := (0)_{i,j=1,\dots,N}$
02  $B := (b_{ij})_{i=1,\dots,N\,j=1,\dots,K}$, $b_{ij} = \mathbf{1}_{\mathrm{Ell}(s_i,g_i,a_i/2,q_i)}(p_j)$
03  $\chi := (\mathbf{1}_A(p_i))_{i=1,\dots,K}$                    // oracle call
04  for $1 \leq i \leq j \leq N$ do
05      if $\|s_j - s_i\| < (a_i + a_j)/2$ then
06          for each $\lambda \in \Lambda := \{l :\ F_l \cap \mathrm{Ell}(s_i, g_i, a_i/2, q_i) \cap \mathrm{Ell}(s_j, g_j, a_j/2, q_j) \neq \emptyset\}$ do
07              for $k := K_{\lambda-1} + 1$ to $K_\lambda$ do
08                  if $\chi_k = 1$ and $b_{ik} = b_{jk} = 1$ then
09                      $\hat{I}_{ij} := \hat{I}_{ij} + w(s_i, p_k) w(s_j, p_k)$
10                  end if
11              end for
12          end for
```

	infill asymptotics		increasing-domain as.	
	Alg. 3.2.28	Alg. 3.2.32	Alg. 3.2.28	Alg. 3.2.32
oracle calls	N^2	1	N^2	N
function evaluations	N^2	N^2	N^2	N^2
test if node is contained within ellipse, or transformation of polar coordinates	N^2	N	N^2	N^2

Table 3.1: Comparison of the asymptotic cost of Algorithms 3.2.28 and 3.2.32.

```
13            end if
14            $\hat{I}_{ij} := \hat{I}_{ij} \cdot \sum_{l\in\Lambda} l\lambda^2(F_l)/\sum_{l\in\Lambda}(K_l - K_{l-1})$
15            $\hat{I}_{ji} := \hat{I}_{ij}$
16        end if
17        return $(\hat{I}_{ij})_{i,j}$
```

Remark 3.2.33 (Algorithm 3.2.32) We discuss implementational issues and possible improvements.

i) In practice, a regular decomposition of F is chosen, so that the set Λ in line 06 is rather easy to determine. For example, let F and $F_1,\dots,F_L$ be rectangles with constant widths $\lambda_1^2(F_1) = \dots = \lambda_1^2(F_L) \geq R$ and heights $\lambda_2^2(F_1) = \dots = \lambda_2^2(F_L) \geq R$, then Λ has ≤ 9 elements.

ii) Similarly not all elements of B have to be computed explicitly: For all l, i with $F_l \cap \mathrm{Ell}(s_i;\cdot) = \emptyset$, put $b_{i,1} = \dots = b_{i,K} = 0$.

iii) A normalization of the integral can be introduced into the algorithm in one of the following ways: (a) If $I(s) = \int_{\mathbb{R}^2} w(s,p)^2 \mathbf{1}_A(p)\,\mathrm{d}p$ can be determined analytically, divide $\hat{I}_{i,j}$ by $(I(s_i)I(s_j))^{1/2}$ in line 14. (b) In the general case, add a few lines in the innermost "for" loop in order to approximate $I(s_\nu)$ by $\hat{I}(s_\nu)$, $\nu = i,j$, on $\mathrm{Ell}(s_\nu;\cdot)$ (not just on the dissection of two ellipses!) and divide $\hat{I}_{ij}$ by $(\hat{I}(s_i)\hat{I}(s_j))^{1/2}$ in line 14.

iv) When approximating a semivariogram γ, $\gamma(s_i,s_i) = 0$ is known, so we do not need to perform the corresponding integral approximation. If furthermore the normalization can be done analytically (cf. iii) (a) above), it is sufficient to generate nodes in $B^*(s_1,\dots,s_N;R)$, i. e. on the dissection of circles around $s_1,\dots,s_N$, not in their union (input (iii) to the algorithm).

Remark 3.2.34 (Cost of Algorithm 3.2.32) With strong boundaries ($A \subsetneq D$), the algorithm for quasi-Monte Carlo integration using a priori nodes needs $\mathcal{O}(N)$ evaluations of the oracle function $\mathbf{1}_A$ when considering increasing-domain asymptotics, and only $\mathcal{O}(1)$ evaluations for infill asymptotics, because in this case, no new nodes have to be created. Moreover, with increasing-domain asymptotics, $\mathcal{O}(N^2)$ times has to be checked if a node is within an ellipse (line 02), but with infill asymptotics only $\mathcal{O}(N)$ times, because then the number of nodes is constant. Finally, $\mathcal{O}(N^2)$ function evaluations are needed with both asymptotics.

See Table 3.1 for a comparison with Algorithm 3.2.28, and remember that in this setting, oracle calls are only needed when using strong boundaries.

Algorithm 3.2.35 (Generation of a priori nodes)

Input: $s_1,\dots,s_N \in \mathbb{R}^2$
$K \in \mathbb{N}$, the number of nodes to generate
$R > 0$, the maximum radius of circles to be considered
a rectangle F, pairwise disjoint $F_1,\dots,F_L$ with $\bigcup_i F_i = F \supset B(s_1,\dots,s_N;R)$

Output: K nodes in $B(s_1, \ldots, s_K; R)$ from a two-dimensional Sobol sequence
c, the total number of nodes that had to be generated in F
$K_1, \ldots, K_L$, the number of nodes in $F_1, \ldots, F_L$

```
    $i := 0, c := 0$
    while $i < K$ do
        $q :=$ next element of a quasi-random sequence in $F$
        if $q \in B^2(s_1, \ldots, s_k; R)$ then
            $p_i := q$
            $i := i + 1$
        end if
        $c := c + 1$
    end while
    sort $(p_j)_{j=1,\ldots,K}$ so that $p_{K_{i-1}+1}, \ldots, p_{K_i} \in F_i$, $i = 1, \ldots, L$, $0 = K_0 \leq K_1 \leq \ldots \leq K_L$
    return $(p_1, \ldots, p_K)$, $c$, $(K_1, \ldots, K_L)$
```

Remark 3.2.36 (Algorithm 3.2.35) i) From the output of Algorithm 3.2.35, N/c is the quasi-Monte Carlo approximation of $\lambda^2(B(s_1, \ldots, s_N; R))$.

ii) The algorithm can be modified in order to generate a higher number of nodes in "more important" areas F_i. Furthermore we might generate quasi-random points in F_i, not globally, in F. However systematic errors near the boundaries between neighbouring F_is have to be avoided.

iii) When approximating semivariograms with ordinary boundaries with a normalization term that is known analytically, we can replace $B(s_1, \ldots, s_N; R)$ by $B^*(s_1, \ldots, s_N; R)$, generating nodes only on pairwise dissections of circles.

3.2.4 Kriging with Approximated Semivariograms

In Section 2.4 we already studied the variability of kriging predictors in terms of the kriging variance (2.13). This inherent randomness of predictions is due to the fact that they are based on data that is considered random itself. However, we also have to be aware of prediction errors caused by errors in approximated semivariogram values used for kriging.

We apply a general result on disturbed systems of linear equations to the kriging equations (2.12).

Theorem 3.2.37 Let $A \in \mathrm{GL}(n)$ and $\Delta A \in \mathbb{R}^{n\times}$ be matrices with $\|A^{-1}\|\|\Delta A\| < 1$. Then $A + \Delta A$ is regular. For given $b \in \mathbb{R}^n \setminus \{0\}$ and $\Delta b \in \mathbb{R}^n$, let $x \in \mathbb{R}^n$ and $\Delta x \in \mathbb{R}^n$ be determined by the systems of linear equations

$$Ax = b \quad \text{and} \quad (A + \Delta A)(x + \Delta x) = b + \Delta b.$$

Then it holds

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\mathrm{cond}A}{1 - \mathrm{cond}A\frac{\|\Delta A\|}{\|A\|}} \left[\frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta b\|}{\|\Delta b\|}\right].$$

(Proof: See Werner (1992).)

Remark 3.2.38 Theorem 3.2.37 gives an upper bound for the error in the solution of a disturbed system of linear equations. The greater the condition number of A is, the greater will be the error bound. Therefore the matrix A is said to be *badly conditioned* if cond$A$ is big.

An orthogonal matrix $U \in \mathbb{R}^{n\times n}$ is well-conditioned with respect to the spectral norm:

$$\mathrm{cond}_2 U = \|U\|_2\|U^T\|_2 = \rho(U^TU)^{1/2}\rho(UU^T)^{1/2} = \rho(I_n) = 1$$

(see Remark 3.2.3). Furthermore,

$$\mathrm{cond}_2 UA = \mathrm{cond}_2 AU = \mathrm{cond}_2 A$$

for all $A \in \mathrm{GL}(n)$.

Remark 3.2.39 (Kriging with approximated semivariograms) For simplicity, we consider the system of linear equations $\Gamma\theta = \gamma$ instead of the kriging equations (2.12). We have approximations Γ, γ of semivariogram evaluations and we write $\Gamma + \Delta\Gamma$ and $\gamma + \Delta\gamma$ for the exact values. Suppose $\|\Gamma\|_\infty = \|\gamma\|_\infty \le n\sigma^2$, where σ^2 is the sill of the semivariogram. We do not know the errors $\Delta\Gamma$ and $\Delta\gamma$, but we might have some absolute error bound $|\Delta\gamma_{i[j]}| \le \varepsilon = \widetilde{\varepsilon}\sigma^2$, $i, j = 1, \ldots, n$. Then we have

$$\|\Delta\Gamma\|_\infty \le n\varepsilon, \qquad \|\Delta\gamma\|_\infty \le n\varepsilon.$$

Applying Theorem 3.2.37, we get

$$\frac{\|\Delta\theta\|_\infty}{\|\theta\|_\infty} \le \frac{2\widetilde{\varepsilon}\,\mathrm{cond}_\infty\Gamma}{1 - \widetilde{\varepsilon}\,\mathrm{cond}_\infty\Gamma},$$

provided that $\|\Gamma\|_\infty^2 < 1/\widetilde{\varepsilon}$ and $\mathrm{cond}_\infty\Gamma < 1/\widetilde{\varepsilon}$.

This result could not yet be applied to kriging with approximated elliptical semivariograms, because the actual error bounds (or more precisely, the constant $c(d)$ in (3.3)) of quasi-Monte Carlo approximation with Sobol nodes could not be found in the available literature. Using the (too loose) Monte Carlo error bounds that are currently implemented (see Section 3.3.6), the condition $\mathrm{cond}_\infty\Gamma < 1/\widetilde{\varepsilon}$ could generally not be satisfied.

Theorem 3.2.40 For an arbitrary matrix $A \in \mathbb{R}^{n\times p}$, $n \ge p$, there exist an orthogonal matrix $Q \in \mathbb{R}^{n\times n}$ and a matrix

$$R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix} \in \mathbb{R}^{n\times p},$$

where $R_1 \in \mathbb{R}^{p\times p}$ is an upper triangular matrix, such that

$$A = QR.$$

This representation is called a *QR decomposition* of A.

(Proof: See Werner (1992).)

Remark 3.2.41 Typical methods for computing the QR decomposition of a matrix are the Householder, Givens and fast Givens method. Each needs $\mathcal{O}(n^3)$ floating point operations (flops) for quadratic matrices, the Householder method requiring about half the number of flops needed by the Givens method. (Werner 1992)

Remark 3.2.42 (Backsolving) For a regular matrix $A \in \mathbb{R}^{n\times n}$, suppose that a QR decomposition $A = QR$ is given. We wish to solve the system of linear equations $Ax = b \in \mathbb{R}^n$. This is equivalent to

$$Rx = Q^T b =: c, \tag{3.5}$$

which can be solved by *backsolving*,

$$x_i = \frac{1}{r_{ii}}\left(c_i - \sum_{j=i+1}^{n} r_{ij}x_j\right), \qquad i = n, n-1, \ldots, 1,$$

since R is an upper triangular matrix. Note that

$$\mathrm{cond}_2 A = \mathrm{cond}_2 R,$$

i. e. the modified problem (3.5) has the same condition number (with respect to the spectral norm) as the initial problem $Ax = b$ (see Remark 3.2.38).

Remark 3.2.43 (Kriging cost) Consider the situation of Section 2.4. We want to predict realizations of $Z_{p_1}, \ldots, Z_{p_k}$ based on observations $Z_{t_1}, \ldots, Z_{t_n}$. Using kriging predictors $Z^*_{p_i} = (Z_{t_1}, \ldots, Z_{t_n})^T \theta^{(i)}$ for $\mathbb{Z}_{p_i}$ requires solving equation (2.12), which we write

$$\widetilde{\Gamma}\theta^{(i)} = \widetilde{\gamma}^{(i)}, \quad i = 1, \ldots, k.$$

Note that $\widetilde{\Gamma}$ does not depend on i, so we can determine its QR decomposition or its inverse in advance at a cost of $\mathcal{O}(n^3)$ flops. Both backsolving and multiplying $\widetilde{\gamma}$ with a matrix require $\mathcal{O}(n^2)$ flops "only".

3.2.5 Minimizing the Mean Squared Error

Remark 3.2.44 We give a result that is useful for implementation: When evaluating the restricted mean squared error function of the trend model, the matrix

$$P = I_n - F(F^TF)^{-1}F^T$$

has to be calculated, where $F \in \mathbb{R}^{n\times p}$ is assumed to be of full rank. Let

$$F = UDV^T,$$

be the singular-value decomposition of F with $U \in \mathbb{R}^{n\times p}$, $D, V \in \mathbb{R}^{p\times p}$, $U^TU = I_p$, $V^TV = I_n$ and a diagonal matrix D (see Theorem 3.2.7). Then, writing $DD = D^2$ and $D^{-1}D^{-1} = D^{-2}$ we get

$$\begin{aligned} F^TF &= VD^TU^TUDV^T = VD^2V^T, \\ (F^TF)^{-1} &= VD^{-2}V^T, \\ F(F^TF)^{-1}F^T &= UDV^TVD^{-2}V^TVDU^T = UDD^{-1}D^{-1}DU^T = UU^T \end{aligned} \tag{3.6}$$

and hence

$$P = I_n - UU^T.$$

Remark 3.2.45 Within the scope of the present work, it was not possible to implement and compare different optimization techniques for minimizing [restricted] mean squared errors. Only a few remarks are given here.

Remark 3.2.46 (Minimization in R) In R, minimization can be carried out using a Newton-type algorithm available through the function `nlm`. (For details, see the references given in the R help files.) In this work, `nlm` is used for minimizing [restricted] mean squared error functions. `nlm` computes numerical derivatives of the target function and consequently needs a high number of (expensive) function evaluations. Derivative-free techniques such as evolutionary algorithms may therefore run much faster.

There also exist specialized numerical methods for nonlinear least-squares problems (see e. g. Nocedal and Wright (1999)). These make use of the special structure of derivatives of mean squared error functions.

Furthermore, linear parameters —in general only sill parameters— can be estimated more efficiently using linear optimization methods, and note also that a (preliminar) a priori estimation of a linear parameter can be made using just one approximation of the semivariogram matrix.

Observation 3.2.47 At this place some of the experiences made with fitting semivariograms of the elliptical class during this work should be anticipated. All observations were made using the implementation described later on in Section 3.3.

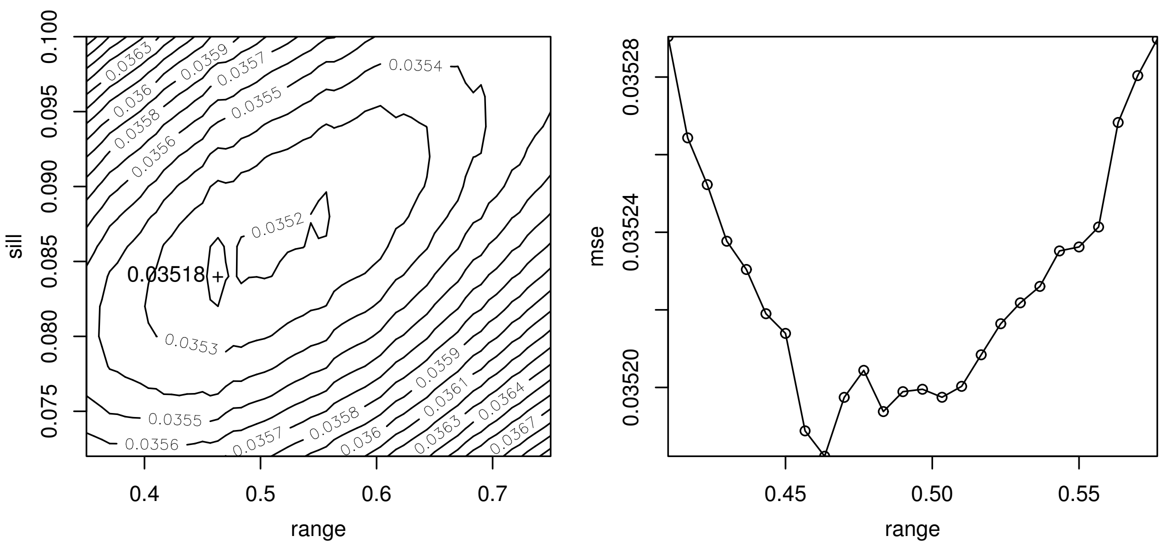


Figure 3.3: A mean squared error function with local minima: contour plot and a cross-section at the global minimum.
The mean squared error function corresponds to the simple elliptical semivariogram (with $q = 1$) and a simulated dataset of 150 observations on $[0, 1]^2$ generated using a spherical semivariogram. For semivariogram approximations, a total of 10 000 a priori nodes was created globally, which corresponds to high precision.

Simple elliptical semivariograms: Mean squared error functions with and without trend (2.21,2.15) corresponding to the simple elliptical semivariogram (Example 2.2.9, Remark 2.2.21) both possess local minima (see figure 3.3). This must be due to the singularity of the semivariogram's first derivative at distances equal to the range. The problem does not vanish when using very high numbers of nodes for quasi-Monte Carlo integration. The piecewise linear elliptical kernel function (Remark 2.2.21) with parameter b close to 1 is practically identical to the simple one, but it is continuous. The minimization of mean squared error performed quite well with the semivariogram corresponding to piecewise elliptical kernel functions with $b = 0.95$, for example.

Integration accuracy: When minimizing mean squared error functions of semivariogram models using quasi-Monte Carlo integration, a relatively small number of nodes and consequently a small accuracy have no effect on the parameter estimates. A "small" number of nodes may be a few dozen within a circle with radius equal to half the semivariogram's estimated range, and hence just a handful of nodes in some of the dissections of circles or ellipses.

Degrees of smoothness: The degree of smoothness of the Bezier semivariogram (cf. Remark 2.2.21) cannot be fitted in a straightforward way. The estimated smoothness parameter ν (≥ 1) is always too small even when the empirical semivariogram reflects the smoothness. Weighted least squares may be a solution (apart from estimating ν by eye).

3.3 Implementation of Geostatistical Methods

While the previous section presented a mathematical framework for an implementation of geostatistical methods and discussed different algorithms, now the actual implementation realized within the data analysis language R 1.2.2 will be described. Although a broad range of object classes and methods was created, of course not all possible alternatives could be realized, and especially exploratory data analysis and visualization techniques had to step behind in order to put a stronger emphasis on itelligent handling of generically stationary semivariogram models and a flexible quasi-Monte Carlo integration algorithm. (In the following sections, some recommendations for further development of the code are made.)

Because of our emphasis on generically stationary processes, the generated R and C code will be called MoGeS, which stands for "**Mo**deling **Ge**neric **S**tationarity".

3.3.1 An Overview

The current implementation deals with the following geostatistical tasks:

Exploratory Data Analysis: Perform a series of operations on and visualization of the data or parts of it in order to recognize spatial and non-spatial regularities, trend, anisotropies etc.

Semivariogram modeling: Select from a variety of stationary and instationary models, determine covariables, assign fixed values to parameters, and add semivariograms. Furthermore, visualize semivariograms, and of course, compute them.

Semivariogram fitting: Estimate semivariogram parameters by minimizing the [restricted] mean squared error function.

Kriging: Compute universal and ordinary kriging predictors and variances.

Interaction with a GIS: Read and write geostatistical datasets and meta-data in an interchangeable file format.

Simulation of datasets: Create random data to a given semivariogram and trend surface.

The specific tasks result in a series of routines, most of them being represented as *methods* of *object classes*. These are entities formed by functions and data, which model the involved mathematical objects and data structures:

Semivariogram models are represented as objects of class `sv`, which is an abstract parent of `svfn`, `svc` and `csv` representing different degrees of "specification" and "aggregation" (see Section 3.3.3). Semivariogram models "know" which parameters they need, if they are stationary etc., and of course they can evaluate "themselves".

Parameters determine semivariograms. They are modeled as an independent object class `param` in order to allow meta-data such as semantics or the range of valid parameter values to be handled consistently together with the parameter vector itself.

Geostatistical data is of course needed for any geostatistical analysis; it consists of locations, measurements and covariables stored in a `svm.data` object.

Nodes are generated a priori for quasi-Monte Carlo integration; they are stored in a `smp.data` object.

Fitted semivariogram models are represented by a `svm` object, which contains information on parameter estimates and other results of mean squared error minimization trials.

See Table 3.2 for a complete list of objects and their methods created for this work.

In the following subsections, a rather "task-oriented" than object-oriented description of the implementation is provided.

Object class	Methods	
`svm.data`	`as.svm.data`	`as.svm.data.svm.data`
	`is.assoc.svm.data`	`is.svm.data`
	`mask.svm.data`	`new.id.svm.data`
	`random.dataset.svm.data`	`read.svm.data`
	`unify.svm.data`	`within.ellipse.svm.data`
	`write.svm.data`	
`sv`	`is.sv`	`plot.sv`
	`random.dataset.sv`	
`svfn`	`as.csv.svfn`	`as.svfn`
	`as.svfn.svfn`	`checkparam.svfn`
	`compute.svfn`	`correctparam.svfn`
	`get.param.prop.svfn`	`getrange.svfn`
	`is.isotropic.svfn`	`is.stationary.svfn`
	`is.svfn`	`penalty.svfn`
	`print.svfn`	`set.param.prop.svfn`
`svc`	`as.csv.svc`	`checkparam.svc`
	`compute.svc`	`correctparam.svc`
	`get.param.prop.svc`	`getrange.svc`
	`is.svc`	`penalty.svc`
	`print.svc`	`set.param.prop.svc`
`csv`	`as.csv`	`as.csv.csv`
	`checkparam.csv`	`compute.csv`
	`getrange.csv`	`is.csv`
	`penalty.csv`	`print.csv`
	`unify.csv`	
`param`	`as.character.param`	`as.param`
	`as.param.param`	`as.vector.param`
	`checkparam.param`	`correctparam.param`
	`fixparam.param`	`getrange.param`
	`is.param`	`mask.param`
	`penalty.param`	`print.param`
	`unify.param`	
`smp.data`	`is.smp.data`	`is.summary.smp.data`
	`plot.smp.data`	`print.smp.data`
	`summary.smp.data`	`within.ellipse.smp.data`
`svm`	`is.svm`	`predict.svm`
	`print.svm`	`summary.svm`
	`svm`	
`summary.svm`	`is.summary.svm`	`print.summary.svm`

Table 3.2: Object classes and their methods, as returned by `methods`.

Generic function	**Methods**	
`as.param`	`as.param.matrix`	`as.param.param`
	`as.param.vector`	
`as.svm.data`	`as.svm.data.list`	`as.svm.data.svm.data`
`compute`	`compute.csv`	`compute.svc`
	`compute.svfn`	
`random.dataset`	`random.dataset.default`	`random.dataset.matrix`
	`random.dataset.sv`	`random.dataset.svm.data`
`svm`	`svm.kriging`	`svm.mse`
`checkparam`	`checkparam.csv`	`checkparam.param`
	`checkparam.svc`	`checkparam.svfn`
`fixparam`	`fixparam.param`	`fixparam.vector`
`unify`	`unify.csv`	`unify.list`
	`unify.matrix`	`unify.param`
	`unify.svm.data`	`unify.vector`
`mask`	`mask.param`	`mask.svm.data`
	`mask.vector`	
`getrange`	`getrange.csv`	`getrange.param`
	`getrange.svc`	`getrange.svfn`
	`getrange.vector`	
`penalty`	`penalty.csv`	`penalty.param`
	`penalty.svc`	`penalty.svfn`
`within.ellipse`	`within.ellipse.matrix`	`within.ellipse.smp.data`
	`within.ellipse.svm.data`	

Table 3.3: Generic functions and their methods, as returned by `methods`.

`calc.mse.data`	`eda.boxplot`
`eda.boxplot.stats`	`eda.filter.pairs`
`eda.pairs`	`eda.pairs2`
`eda.plot.aoi`	`eda.plot.positive.correlations`
`eda.plot.positive.correlations2`	`eda.plot.semivariogram`
`eda.semivariogram.cloud`	`generate.grid`
`generate.random.points`	`needs.smp.data`
`setnames`	`svm.kriging`
`svm.mse`	

Table 3.4: Further (non-generic) functions (incomplete list).

generate_QMC_sampling_points_exp	QMC_int_ellipses_exp
QMC_GetSMPRectangles_exp	within_ellipse_exp
within_ellipse_smp_exp	sobseq_init_exp
sobseq_exp	sobseq2d_exp

Table 3.5: C routines used for quasi-Monte Carlo approximation and Sobol sequence generation (only exported functions are listed).

3.3.2 Handling Geostatistical Data

In our context, geostatistical data consists of

- locations $s_1, \dots, s_N$ of measurement or prediction sites,
- measurements $z_1, \dots, z_N$ or predicted values (if available),
- covariables $(g(s_i)^T)_{i=1,\dots,N}$ of the semivariogram model (if required),

and, in the case of a model with trend,

- covariables $(f(s_i)^T)_{i=1,\dots,N}$ for the linear trend (2.17).

In MoGeS, this data is modeled as an object of class `svm.data`, which is basically a `list` with components named `"xy"`, `"z"`, `"g"` and `"f"`, where `"xy"` is the only one that is required. A `svm.data` object can be created from a `list` object using a `as.svm.data` method, and it can be read from a text file (exported from a GIS) by `read.svm.data`.

The components are matrices of N rows, and the columns of `"g"` must be named so that they can be identified by a semivariogram object (see below, Section 3.3.3). The following code creates a simple `svm.data` object with uniformly distributed points on $]0,1[^2$ and a `"g"` covariable describing a (global) geometric anisotropy:

```
> N <- 100
> d <- as.svm.data( list(xy=cbind(runif(N),runif(N)),
+     indicator=rep(1,N), orientation=rep(pi/4,N)) )
```

If we want to simulate measurements corresponding to an elliptical semivariogram with sill 1, range 0.3 and axis ratio 0.7, we could for example enter:

```
> d <- as.svm.data( list(indicator=rep(1,N), orientation=rep(pi/4,N)) )
> d <- random.dataset( d, svfn.elliptical.pwlinear,
+    param=param(c(1,0.4,0.7)), create.xy="unif", n=N, smp=5000 )
```

(The `param` object and function will be described later on; `smp=5000` specifies that 5000 a priori nodes shall be used for semivariogram approximation.)

Sometimes we want to merge together geostatistical datasets or extract a subsample. The latter is done by `svm.data`'s `mask` method,

```
mask.svm.data(d, m)
```

where `m` is a logical or numeric vector specifying which data rows to return. For example,

```
> d.poly1 <- mask(d, d$g[,"ind.poly1"]==1)
```

assigns to `d.poly1` all measurements and corresponding data that are within a region with indicator covariable stored in the `"ind.poly1"` column of the covariable matrix `d$g`. Merging is done by a `unify` method (see Appendix A for details; similar `mask` and `unify` methods were implemented for other object classes).

Function name	**Description**	**Parameters**	**Covar.**
`svfn.elliptical`	(abstract) elliptical model	σ^2, a, q $(\dots)$	$\mathbf{1}_A$, ϕ
`---.bezier`	elliptical with Bezier kernel	σ^2, a, q, ν	$\mathbf{1}_A$, ϕ
`---.linear`	linear elliptical	σ^2, a, q	$\mathbf{1}_A$, ϕ
`---.pwlinear`	piecewise linear elliptical	σ^2, a, q, b	$\mathbf{1}_A$, ϕ
`---.geometrical`	abstract elliptical, with geometric anisotropy (ϕ constant)	σ^2, a, q $(\dots)$	$\mathbf{1}_A$
`---.vargeometrical`	abstract elliptical, with geometric anisotropy (ϕ as parameter)	σ^2, a, q, ϕ $(\dots)$	$\mathbf{1}_A$
`svfn.empirical`	(global) empirical semivariogram with 10 steps	$\sigma_1^2, \dots, \sigma_{10}^2$	–
`svfn.nugget`	nugget effect	σ^2	$\mathbf{1}_A$
`svfn.nugget.global`	global nugget effect	σ^2	–
`svfn.spherical`	spherical semivariogram	σ^2, a	–

Table 3.6: Implemented semivariogram models. Parameter semantics: σ^2 = sill, a = range, q = axis ratio, b = break point, ν = exponent of Bezier kernel function, ϕ = direction of anisotropy. Covariable semantics: $\mathbf{1}_A$ = characteristic function, ϕ = direction of local anisotropy. "Abstract" means that the kernel function of the semivariogram has to be determined by an argument to `svfn.elliptical`.

3.3.3 Semivariogram Models

A generically stationary semivariogram model is a function depending on site locations, parameters and an influence function (covariables), and in the case of a the class of elliptical semivariograms, we need further arguments in order to choose between several kernel functions. Furthermore, when approximating a semivariogram function, we will need arguments that influence numerical integration. In MoGeS, the class of elliptical semivariogram models is implemented as a function

```
svfn.elliptical(d, ref, param, smp, fun, fun.params, global,
   strong.boundaries, extra.info, cov, arglist, ...)
```

In this list of arguments, `d` is a `svm.data` object that contains at least measurement locations `d$xy` $= (s_1, \dots, s_N)^T$ and the covariable matrix `d$g` $= (g(s_i)^T)_{i=1,\dots,N}$, and `param` must be of class `param`. A semivariogram function expects its parameters and covariables to be identified by certain keywords, in the above case `"range"`, `"sill"` and `"q"` as names of the `parameter` elements, and `"indicator"` and `"orientation"` as column names for the covariable matrix. Moreover, the `fun` and `fun.params` arguments specify the kernel function and the corresponding parameters to be used (see Remark 2.2.21).

`svfn.elliptical` calls the quasi-Monte Carlo integration routine `QMC.int.ellipses` for approximating the induced *co*variogram matrix given by

$$C(s_i, s_j) = \int_E w(s_i, p) w(s_j, p)\, \mathrm{d}p$$

and then determines the semivariogram matrix using the relation

$$\gamma(s, t) = \sigma^2 - C(s, t),$$

where σ^2 = `param["sill"]`. Note that `svfn.elliptical` takes a `cov` argument, we can use it to get a covariogram matrix.

`svfn.elliptical` supports ordinary and strong boundaries (see Definition 2.2.25).

Further semivariogram models are implemented, for example the spherical one or a nugget effect, as well as some models that are derived from the (abstract) elliptical semivariogram model `svfn.elliptical` (see Table 3.6). It is rather easy to implement additional semivariogram models using the existing functions as templates.

In MoGeS, R functions that implement semivariogram models are converted to objects of class `svfn`, which in addition to the function code itself contain information on the set Θ of valid parameters, and on the stationarity of a model, for example. The summarized output produced by `svfn`'s `print` method gives an overview:

```
> svfn.elliptical.pwlinear
[...]
Properties:  instationary, anisotropic; needs a priori nodes

Parameters:
     names semantics minimum maximum
[1,] sill  sill      0       Inf
[2,] range range     0       Inf
[3,] q     q         0       1
[4,] break break     0       1

g-components needed:
[1] "indicator"   "orientation"
```

These attributes that are assigned to a semivariogram model make its handling much safer.

In geostatistics we sometimes want to add different semivariogram models that for example belong to subprocesses (see Section 2.2.4). These composed semivariogram models are represented as objects of class `csv` and created by the function `csv`. These objects are basically lists of “semivariogram component” objects of class `svc`, which themselves are derived from `svfn` objects using the function `svc`. `svc` objects are not just copies of `svfn` objects; they have a higher degree of “specification” in the sense that fixed values may be assigned to single parameters. Furthermore, and this is also an important feature, parameter names may be changed using the `param.alias` argument of `svc`. This is sometimes necessary because two `svfn` objects that are added may expect parameters of the same alias, say `"sill"`, and in most cases we want them to be independent of each other. If, in contrast, they should be identical, we just have to assign the same name to both. This also applies to covariables and the `svc`'s `g.alias` argument. (Later we will see some sample code in which aliasing takes place.)

However, sometimes we do not need a complex semivariogram model but just want to fit a very simple one. In this case we just use a `svfn` object without constructing a `csv` object. This is possible because these three classes all belong to the more abstract class `sv` and hence have essentially the same functionality and are treated indifferently by most functions.

Semivariogram objects can be computed (`compute`), plotted (`plot.sv`) and used for simulating data (`random.dataset.sv`).

A special object class `param` was implemented for handling parameters. A `param` object is a numeric vector with named elements and a `"semantics"` attribute that specifies the “meaning” of each parameter. Names can be arbitrary because they are unaliased before being passed to a semivariogram object. Semantics may take values such as

`"sill"`	– semivariogram sill
`"range"`	– semivariogram range
`"q"`	– ellipse's axis ratio (for elliptical semivariograms)
`"break"`	– breaking point (piecewise linear ell. sv.)
`"degree"`	– exponent (Bezier elliptical semivariogram)

or any other, if a specific semivariogram model (that may be supplied by the user) expects it. The use of parameter semantics also makes it possible to determine the range of a semivariogram. An example:

```
> my.param <- param( c(1,0.3,0.4), sem=c("sill","range","range"),
+    nm=c("MySill","MyRange","AnotherRange") )
> my.fix.param <- param(0.5,"range")
> getrange(my.param)
[1] 0.4
> my.model <- csv( list(ASpherical=svc(svfn.spherical,
+      fix.param=my.fix.param),
+   MySpherical=svc(svfn.spherical,
+      param.alias=setnames(c("MySill","MyRange"),c("sill","range"))) ) )
> getrange(my.model, param=my.param)
[1] 0.5
```

Furthermore, parameter semantics can be used for checking for validity of supplied values (e. g., any `"sill"` must be non-negative) and for avoiding that parameters of different semantics are identified. Finally, `param`eter objects can be `unify`ed and `mask`ed.

The notions of semantics and aliasing are also used by the GIS interface (see Section 3.4).

3.3.4 Fitting Semivariogram Models

Semivariogram parameter estimation is implemented using the mean squared error function (2.15) and its relative in the presence of trend, the restricted mean squared error (2.21), as target functions, and the nonlinear Newton-type minimizer `nlm` included in the R base package. `nlm` uses numerical derivatives (unless analytical derivatives are supplied) and allows to set several options such as step and gradient tolerances. (See `nlm` in the R documentation for more details and Section 3.2.5 for a discussion.)

In MoGeS, semivariogram fitting is performed by the function `svm`:

```
svm(d, sv, param, fix.param, trend, smp, print.level, iterlim,
   steptol, mse.estimate, trials, fun, ...)
```

Required arguments are a `svm.data` object (`d`), a semivariogram model object (`sv`), and of course a starting parameter value (`param`). This may be either a `param` object or a `matrix` (with named columns and a `"semantics"` attribute), the latter having parameter vectors in each row, which will be used for performing a series of minimization trials.

Some arguments are passed to `nlm` in order to control the optimizer, and `smp` and `...` are additional arguments to `sv`'s `compute` method.

`svm` returns a list of class `svm` containing the mean squared error minima obtained in each optimization (`"mses"` component) and the smallest minimum encountered (`"mse"` component), as well as the corresponding parameter estimates (`"est.params"` component for all, `"est.param"` for the best). In addition, the returned `svm` object stores starting parameter values, the `sv` object used and some other information.

A human-readable output of fitting results is generated by `svm`'s `summary` method (for an example see p. 62).

The target function passed to the optimizer is `svm.mse`. It computes the "classical" or restricted mean squared error (equations (2.15), (2.21)), depending on its `trend` argument. In the latter case, `svm.mse` makes use of the singular-value decomposition of the matrix $F =$ `d$f` and applies (3.6).

Note that the projection matrix P used for computing restricted mean squared errors does not depend on the semivariogram parameters; the same applies to zz^T and, in the "classical" case, $((z_i - z_j)^2)_{i,j}$. These matrices are determined in advance by `calc.mse.data` and passed to `svm.mse` in each `nlm` iteration.

`svm.mse` checks for validity of the parameters. Validity regions are in general given by the `svfn` object (and inherited by derived semivariogram models), but the user may define additional restrictions when creating a `svc` object. Outside the validity region Θ, the mean squared error is multiplied with a factor > 1 that linearly increases with distance from Θ (for details, see `penalty` in Appendix A and the source code). However, the current implementation only supports closed intervals (including infinite ones).

The fitting method described above showed an acceptable performance, but speed should be improved. Using an elliptical semivariogram model with three free parameters (sill, range, axis ratio), 200 simulated measurements and 3 000 a priori nodes for integration, it often takes some 10 to 30 minimization steps and one to five minutes of time to estimate parameters. See Section 3.2.5 for a few remarks on improving this performance and some observations.

3.3.5 Semivariogram Approximation

The approximation of semivariograms (and covariograms) of the elliptical class was implemented in C and made accessible from R by an interface function `QMC.int.ellipses`. The C routine `QMC_int_ellipses_exp`,

```
extern void __declspec(dllexport) QMC_int_ellipses_exp
   (double xy1[], double g1[], double a1[], double q1[], double *lpdN1,
    double xy2[], double g2[], double a2[], double q2[], double *lpdN2,
    double *lpdIsSymmetric, double *lpdCalcDiagonal,
    double *lpdGetExtraInfo, double *lpdForceIntUnion,
    double *lpdStrongBoundaries, char **lplpszFunction,
    double *lpdFunctionParameters, char **lplpszNormBy,
    double *smp_xy, double *smp_N, double *smp_N_total,
    double smp_indicator[],
    double *smp_frame,
    double *smp_rectnum, double *smp_rectsize, double *smp_nrect,
    double *smp_rectindexfrom, double *smp_rectindexto,
    double *smp_area,
    double integral[], double int_lo[], double int_hi[],
    double area[], double N_inside[],
    double *err)
```

is an implementation of Algorithm 3.2.32 using a priori nodes, including the modifications suggested in Remark 3.2.33 i) and iii). Some of the arguments are assigned obvious default values by `QMC.int.ellipses` (e. g. `*lpdCalcDiagonal=0`, we know that $\gamma(s, s) = 0$), see Section 3.2.3 for a discussion and Appendix A.3 for details of the functions' arguments.

The C routine `QMC_int_ellipses_exp` is capable of processing ellipses with different radii and axis ratios, a feature that is not needed by `svfn.elliptical` but can be used for implementing the second semivariogram presented in the soil loss example (Example 2.2.23). For this purpose, it is also necessary to write a suitable weight function, which can easily be made by adapting the C function

```
double qmc_user(double Px, double Py, double Xx, double Xy,
   double g, double a, double q, double *lpParams)
```

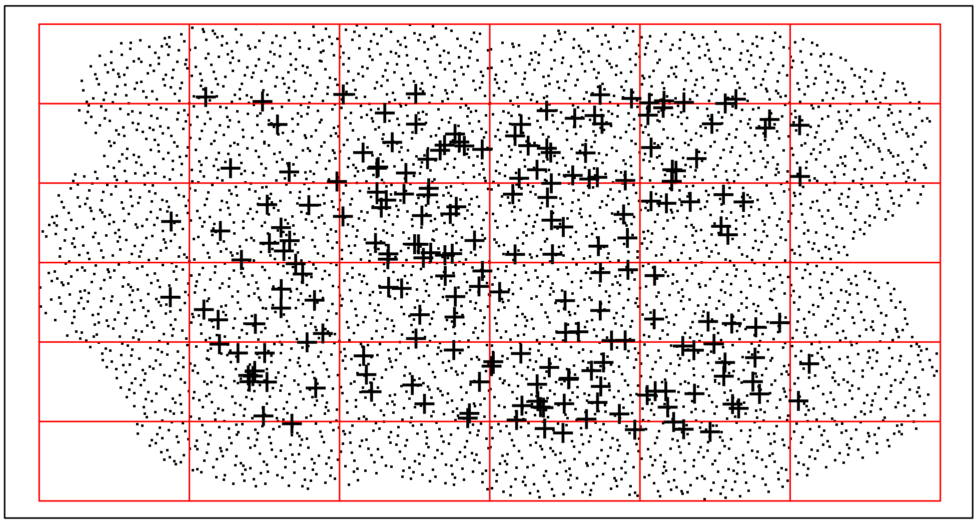


Figure 3.4: 4000 Sobol a priori nodes generated by `smp.data` and the corresponding decomposition of the surrounding rectangle (plotted with `plot.smp.data`).

and using the function name `"user"` with `svfn.elliptical`.

`QMC.int.ellipses` (and its C equivalent) returns a matrix of the approximated integrals

$$\int_{\mathbb{R}^d} w(s_i, p) w(t_j, p)\,\mathrm{d}p \qquad i = 1, \ldots, N_1, j = 1, \ldots, N_2,$$

i. e. a covariogram matrix or vector with ordinary or strong boundaries and, if desired, after normalization. Additionally, an argument of `extra.info=TRUE` yields a 95 % confidence interval based on the assumption of normally distributed errors due to uniformly distributed random (instead of quasi-random) nodes. These admittedly inappropriate error bounds were chosen because only the asymptotic order (3.3) of the star discrepancy of the Sobol sequence could be found in the available literature; a factor that only depends on the problem dimension is unknown. In any case, the implemented (probabilistic) error bounds overestimate the quasi-Monte Carlo error bounds at least asymptotically (see Section 3.2.3).

A few remarks should be made about the ways a priori nodes are created and handled. The `smp.data` function uses Algorithm 3.2.35 and the C routine `generate_QMC_sampling_points_exp` for generating a `smp.data` object. This is basically a `list` containing

- nodes $p_1, \ldots, p_K$ (`"xy"` component),
- if necessary (e. g. in the case of strong boundaries), covariables $\mathbf{1}_A(p)$ evaluated at each node (`"g"` component),
- a surrounding rectangle F determined by its corners, and disjoint subrectangles $F_1, \ldots, F_L$ (`"rect.num"` and `"rect.size"` components),
- indices $K_1, \ldots, K_L$ such that $p_{K_{i-1}+1}, \ldots, p_{K_i} \in F_i$ ($K_0 := 0$; `"rect.index.from"`, `"rect.index.to"`),
- vectors indicating which nodes are within $B(s_1, \ldots, s_N; R)$ and $B^*(s_1, \ldots, s_N; R)$ for $R =$ `Rmin`, `Rmax` (`"within.disj.min"`, `"within.disj.max"`, `"within.conj.min"` components),
- and some other data.

See figure 3.4 for an example plot of a priori nodes generated with `smp.data`.

`smp.data` generates only nodes that are within the union of circles $B(s_1, \ldots, s_N; \texttt{Rmax})$ of radius `Rmax` around the measurement sites `d$xy` $= (s_1, \ldots, s_N)$, so the use of a `smp.data` object is restricted in two ways:

i) It should not be used together with `svm.data` objects that are not subsets of the object that the nodes were generated for.

ii) A `smp.data` object should not be used for approximating semivariograms with ranges greater than 2`Rmax`. This occasionally occurs during minimization when the `nlm` algorithm "tries" large steps in the "wrong" direction. However, these approximations based on inappropriate nodes cause an increase in the mean squared error, so the algorithm is thrown back into the region of ranges ≤2`Rmax`.

The quasi-random nodes used by MoGeS belong to a two-dimensional Sobol sequence and are generated by the C routine `sobseq2d_exp` using an algorithm given by Press et al. (1992) (`sobseq` routine). The generator is initialized by `sobseq_init`. R interface functions are `sobseq.init`, `sobseq2d`.

Further C routines for quasi-Monte Carlo integration include those of the form `within_ellip-se_..._exp`, which return a matrix of zeroes and ones indicating which points are within which ellipse. There also exist R interface functions (see Table 3.3).

The C routines are compiled to the Windows DLL `numint.dll` and accessed through the R function `.C` (see R documentation, help topic `".C"`). Some care must be taken when passing `integer` arguments to C routines; for safety, they are converted to `double` precision (but maybe this is or was just a bug of the R version used at the beginning). The existing C code should also be improved in order to handle R lists directly instead of passing the components as separate numerical vectors and matrices, but this requires studying the incompletely documented "deep R".

3.3.6 Kriging

Indicator and universal kriging are implemented in a rather simple way using equation (2.12) and a QR decomposition of the coefficient matrix (or alternatively, a singular-value decomposition). When predicting the process at several locations based on the same measurements, the coefficient matrix remains unchanged and therefore has to be inverted only once.

In MoGeS, kriging can be performed by a call to the function

```
svm.kriging(data, newdata, sv, param, G, method = "qr", tol = 1e-10,
   trend = FALSE, ...)
```

or by using the method

```
predict.svm(object, data, newdata, ...)
```

belonging to the object class `svm`. This `predict` method is an analogue of `predict` methods existing for other statistical objects in R, such as `lm` or `nlm` for (generalized) linear models.

The kriging functions return an object of class `predict.svm`, i. e. a list containing predicted values (`"z"` component), kriging variances (`"var"` component) and confidence intervals at a level of significance of 95 % (`"z.lo"`, `"z.hi"` components).

3.3.7 Exploratory Data Analysis and Visualization

Some functions were written to help visualize and explore geostatistical data, but the collection is incomplete. An important instrument is the empirical semivariogram given by the moment estimator

$$\widehat{\gamma}(h) = \frac{1}{2|N(h)|} \sum_{s_i, s_j \in N(h)} (z_i - z_j)^2,$$

where $N(h)$ is the set of pairs (s_i, s_j) that have a similar lag $h = s_i - s_j \in \mathbb{R}^2$ (or $h = \|s_i - s_j\| \in \mathbb{R}$). This is implemented as

```
eda.semivariogram.cloud(d, intv = "default", method = "equidistant",
   direc = NULL, tol = (1/18) * pi, extra.info = FALSE, trend = FALSE,
   mse.data)
```

The estimators may be based on intervals of the same length (`method="equidistant"`) or of same number of observations (`method="moment"`). Furthermore, optionally filtering of pairs of points of (almost) the same orientation can be performed in order to assess geometric anisotropies (arguments `direc` and `tol`). In the presence of trend, the empirical semivariogram `svfn.empirical` as given by equation (2.18) has to be fitted.

The R base package functions `boxplot` and `boxplot.stats` were adapted to spatial data analysis: The functions `eda.boxplot` and `eda.boxplot.stats` compute box-and-whisker plots using a projection of two-dimensional coordinates onto a straight line.

Many of the exploratory data analysis functions available within the R distribution may be applied to geostatistical data. There are also visualization routines available such as `contour` and `persp` for two- and three-dimensional surface representations, but they are restricted to gridded data. Since visualization is an important instrument for analyzing geostatistical data, the capabilities of GIS or specialized visualization software should also be used for three-dimensional plots.

3.4 Implementation of a GIS Interface

Due to the great effort put onto the implementation of a useful geostatistical package for modeling generic stationarity within R, the GIS interface remains comparatively basic. In spite of this restriction, it shows how a flexible linkage of a GIS with a data analysis tool can be designed, and hopefully it is the starting point for further development efforts that aim at covering a wider range of geostatistical data analysis tasks and supplying an intelligent interface.

The ArcView/MoGeS interface performs geostatistical modeling in four steps:

Export data: Select a set of points and corresponding data from a *point theme* and its data base or *table* (in ArcView terminology) and convert it to a text file format that can be read by MoGeS.

Specify a semivariogram model: Select semivariogram models, and link covariables with fields in the theme's data base. If desired, assign fixed values to parameters or identify parameters.

Fit the semivariogram model: Choose starting parameter values, and perform the fitting through a call to the R environment.

Perform kriging: Select measurement and prediction locations, export the corresponding data and perform kriging by calling the R environment.

Main scripts	
`Step 1 - Export Data`	`Step 2 - Model Specification`
`Step 3 - Semivariogram Fitting`	`Step 4 - Kriging`
Auxiliar scripts	
`CheckVariables`	`DifferentParameterSemantics`
`DoExportData`	`EnterStartingValues`
`FixVariables`	`GetSemantics`
`IdentifyVariables`	`QueryNewAlias`
`R_AssignAlias`	`R_AssignParam`
`R_AssignSemivariogramComponent`	`R_AssignSemivariogramModel`
`R_ReadSvmData`	`SelectSemivariogramModel`
`shp2ascii`	

Table 3.7: AVENUE scripts forming the ArcView/MoGeS interface.

The ArcView/MoGeS interface is a collection of AVENUE scripts that perform these tasks (see Table 3.7, Appendix A). The code and a sample project can be obtained from the author (e-mail: `ali@proforma.de`). It can be executed using commands added to the `Theme` menu, which is available when a theme is active. The scripts however do not cover the complete MoGeS functionality available within R. Nevertheless, the problems mentioned above can be solved more easily than by hand, since R code is generated and executed automatically, and a user who is familiar with R will be able to add flexibility by modifying this code or doing additional analyses using the whole spectrum of R and MoGeS functions.

The AVENUE scripts model parameter vectors, names, aliases and semantics just as MoGeS does, however as seperated lists rather than object classes or names vectors.

In its current implementation, the ArcView/MoGeS interface strongly depends on the MoGeS implementation (i. e. its function identifiers, argument names etc.), which makes it very sensible to small changes in MoGeS. This could be overcome by using a meta-language for geostatistical modeling that makes for example model specifications independent of the actual implementation that executes it.

3.4.1 Exporting Geostatistical Data

Before executing the script `"Step 1 - Export Data"`, an ArcView point theme has to be active and all the data records to be exported must be selected. The user specifies the names of fields that will be created and filled with projected point coordinates needed by MoGeS for georeferencing. The exported data must contain all the covariables (e. g. $\mathbf{1}_D(s_i)$, $\phi(s_i)$ in the case of an elliptical semivariogram on D) needed for modeling. The data will be stored in a text file of `.csv` ("comma-separated value") format that can also be read by many other applications.

3.4.2 Specifying a Semivariogram Model

The script `"Step 2 - Model Specification"` first initializes a series of global variables that contain, among other data, identifiers of some semivariogram models implemented within MoGeS (e. g. `svfn.elliptical.pwlinear`) and the corresponding parameter and covariable names and semantics. The user is then asked to select one or more of these semivariograms, and he may identify parameters. After that, fixed values can be assigned to parameters, if desired. Note that the supplied scripts only accept valid parameter values (script `"CheckVariables"`); this is possible due to the concept of parameter semantics. Finally, a correspondence between the covariables required by the selected semivariogram model and the field names of the exported data base is established by the user (see figure 4.1 on page 59).

The semivariogram specification is stored in global variables and thus can be used in subsequent executions of the following two steps.

3.4.3 Fitting a Semivariogram Model

Given a semivariogram model, the user can call the `Step 3 - Semivariogram Fitting` script to specify starting parameter values and the field that contains the observed values. Then R code is generated that reads the geostatistical data, specifies the semivariogram model and fits it. If desired by the user, this code will automatically be executed within R through a call to the terminal version of R, `Rterm.exe` (see figure 4.2).[3]

3.4.4 Performing Kriging

The script `Step 4 - Kriging` requires the active theme to have a field `"Observed"`. Non-zero entries indicate that the corresponding record represents a measurement site. For kriging, the selected records from the theme's data base are exported again, parameters for the semivariogram model have to be entered, and if desired, the code is automatically executed by R. In this case, the generated data is written to a file with the extension `.krg` that can be loaded into ArcView using the commands `Add Table` and `Add Event Theme`. The field `"Z"` in the imported data base then contains the predicted values. These results can be visualized in ArcView as a three-dimensional Digital Elevation Model (DEM).

[3]Note that the correct path of `Rterm.exe` must be specified within the script's code. Currently the path is set to `Z:/rw1023/bin`.

Chapter 4

Application

In this Chapter the geostatistical methods without stationarity assumptions that were presented in Chapter 2 will be applied to simulated data using the MoGeS routines. We will generate two datasets, one with linear trend and another one with non-geometric anisotropies, and use the restricted mean squared error and elliptical semivariograms, respectively, for model fitting.

When testing geostatistical methods, real-world datasets have the crucial disadvantage that in most cases, one does not have precise knowledge of the stochastic process and its distribution law. Exploratory Data Analysis and possibly data transformation and selection have to be performed, and different models must be fitted and compared in order to find one that allows us to make good predictions.

When testing statistical methods, however, we need datasets with well-known properties, because we want to see if the analysis techniques yield results that are consistent with our a priori knowledge. Therefore we will generate data based on a semivariogram model and reasonable parameter values and covariables and hope that our fitting methods can reproduce them or perform at least as good as “classical” estimators.

Before analyzing the simulated datasets, an introductory sample session will exemplify the use of our implementation within ArcView and R.

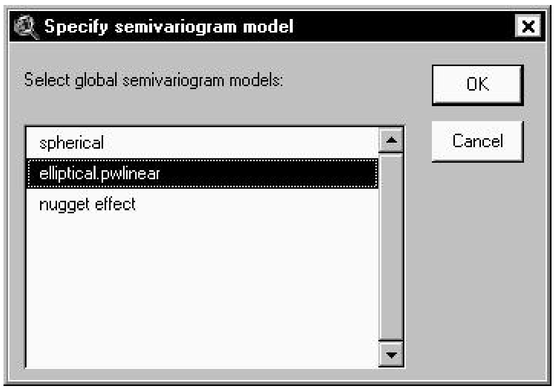
4.1 A Sample Session

This section shows how the ArcView/MoGeS interface integrates the GIS ArcView with the data analysis environment R. It is also intended to be an introduction to MoGeS, since the R code generated automatically by the interface scripts can easily be adapted to other situations and extended using other R and MoGeS functions.

We use a climatologic–ecologic dataset provided by Prof. Dr. Michael Richter (Erlangen), who studies climate gradients and erosion processes in the Andes of Southern Ecuador. The part of the dataset that we will use here basically consists of topographic data and humidity indices at 137 locations within an area of about 120 km × 120 km. These indices represent the approximate average number of humid months per year and are derived from detailed phytosociologic studies, since weather stations are very rare and in many cases not representative in this high-mountain area.

There are climatologic reasons for assuming that there exists a trend that depends on altitude, slope exposition and distance from the continental watershed that crosses the investigated are in North–South direction, and anisotropy can also be expected to exist related to the orientation of valleys.

However, we will not study this challenging dataset in detail. Instead we will limit ourselves


Specify semivariogram model
Select global semivariogram models:
OK
Cancel
spherical
elliptical.pwlinear
nugget effect

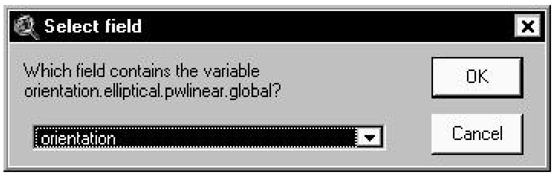

Select field
Which field contains the variable orientation.elliptical.pwlinear.global?
OK
Cancel
orientation

Figure 4.1: Specifying semivariogram models within ArcView using the ArcView/MoGeS interface.

to fitting just one global semivariogram, an elliptical semivariogram with piecewise linear kernel function and geometric anisotropy.

The data needed can be accessed from the ArcView project file `geostat.apr`, where it is included as a point theme called `Humid.sph` and the corresponding table. After selecting all the theme's points and activating the theme, the data can be exported using the command `GS * Export Data` in the Themes menu, which runs the script `Step 0 - Export Data`. Then, the semivariogram is determined by executing the `GS * Model Specification` command (see figure 4.1). The observed values are stored in the table's `Humo` field, and the presumed (constant) direction of anisotropy in `orientation`[1].

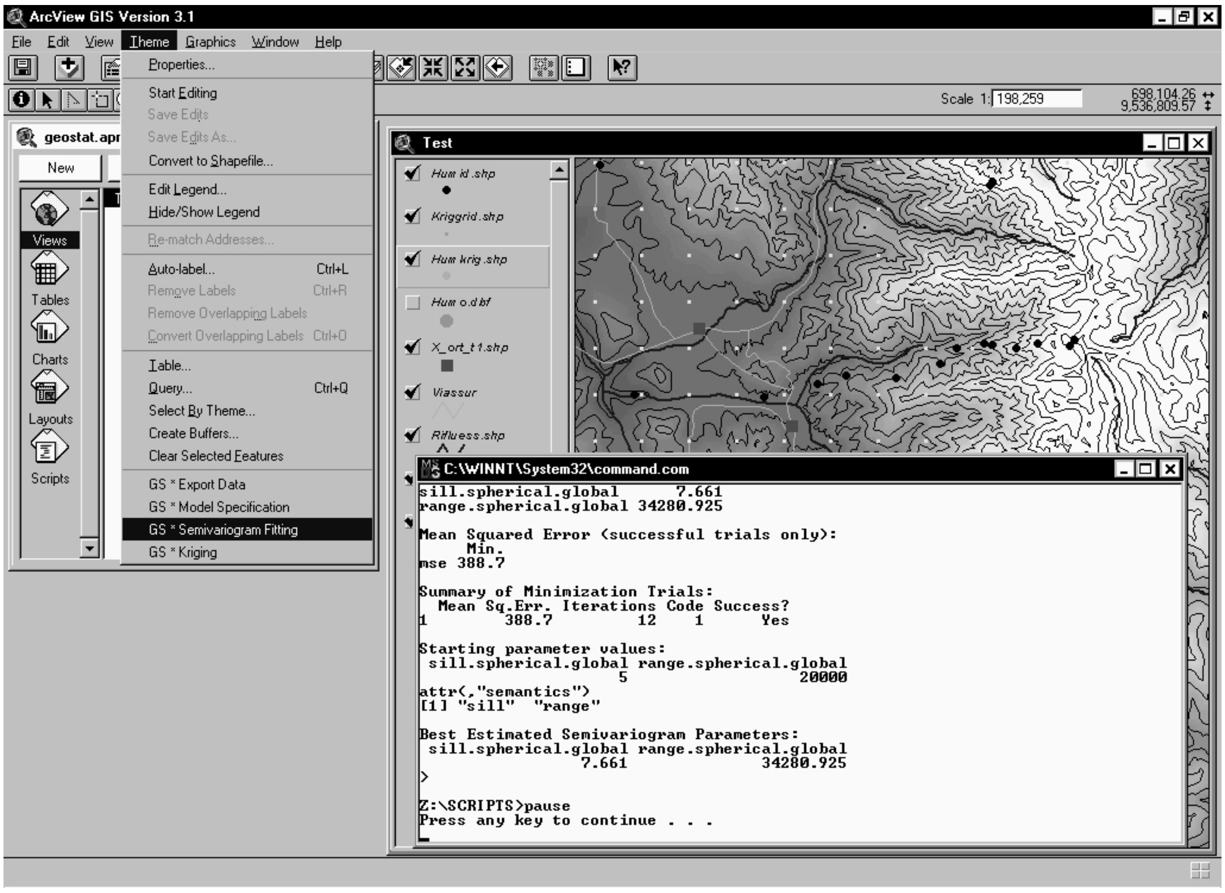
The next step, `GS * Semivariogram Fitting`, generates the following R code stored in a `.R` file (some comments were added by hand in order to make the following lines self-explaining):

```
library("mva")
source("numint.r")
source("fit.r")
source("eda.r")
source("ell.r")

###### read geostatistical data
d <- read.svm.data( file="z:/scripts/humid.csv",
   xnames="x", ynames="y", znames="Humo", gnames=c("orientation") )
n <- nrow(d$xy)

###### specify a semivariogram model
# we have one fixed parameter:
fpa <- param( c(0.9),
   nm = c("break.elliptical.pwlinear.global"),
   sem = c("break") )
# parameter aliases:
pa.al <- setnames( c("sill.elliptical.pwlinear.global",
   "range.elliptical.pwlinear.global","q.elliptical.pwlinear.global",
   "break.elliptical.pwlinear.global"),
   c("sill","range","q","break") )
# covariable aliases:
g.al <- setnames( c("orientation.elliptical.pwlinear.global"),
   c("orientation") )
# we have just one 'svc' semivariogram object:
svc1 <- svc( svfn.elliptical.pwlinear.global, fix.param=fpa,
   param.alias=pa.al, g.alias=g.al )
# and this is our composed 'csv' semivariogram object:
sv <- csv( list( svc1 = svc1 ) )
```

[1]In a more sophisticated model, directions of anisotropy are determined using a Digital Elevation Model.



```
sill.spherical.global       7.661
range.spherical.global 34280.925

Mean Squared Error (successful trials only):
      Min.
mse 388.7

Summary of Minimization Trials:
  Mean Sq.Err. Iterations Code Success?
1       388.7          12    1       Yes

Starting parameter values:
 sill.spherical.global range.spherical.global
                     5                  20000
attr(,"semantics")
[1] "sill"  "range"

Best Estimated Semivariogram Parameters:
 sill.spherical.global range.spherical.global
                 7.661              34280.925
>

Z:\SCRIPTS>pause
Press any key to continue . . .
```

Figure 4.2: Semivariogram fitting within ArcView using the ArcView/MoGeS interface.

```
###### fit the semivariogram model:
# starting ‘param’eter object:
start <- param( c(5,10000,0.7),
   nm = c("sill.elliptical.pwlinear.global","range.elliptical.pwlinear.global",
      "q.elliptical.pwlinear.global"),
   sem = c("sill","range","q") )  # semantics!
# generate nodes for quasi-monte carlo integration (if necessary):
smp <- NULL
if (needs.smp.data(sv))  # ‘sv’ knows if it needs a ‘smp.data’ object with
                         # a priori nodes, even if ‘sv’ is rather complex!
   # the ‘Rmax’ argument must be sufficiently large!
   # 5000 a priori nodes should be sufficient in most cases
   smp <- smp.data(d, Rmax=1.7*max(getrange(sv,start))/2, N=5000)
# finally we can fit the model:
svmfit <- svm(sv,d=d,param=start,smp=smp,trend=FALSE)
# and print a summary of the results:
print(summary(svmfit))
```

The code is interpreted by the terminal version of R, `Rterm.exe`, which is executed within a DOS window (see figure 4.2). For this purpose, the following batch file is generated:

```
z:\rw1023\bin\Rterm --no-restore --no-save <z:\scripts\humid.r
pause
```

Instead of executing `humid.r` automatically, the user might prefer to edit the code or execute it step by step in order to observe the results more in detail.

Kriging within ArcView works in a very similar way as fitting semivariogram models, and in fact great part of the R code generated is the same since it is only concerned with reading the data

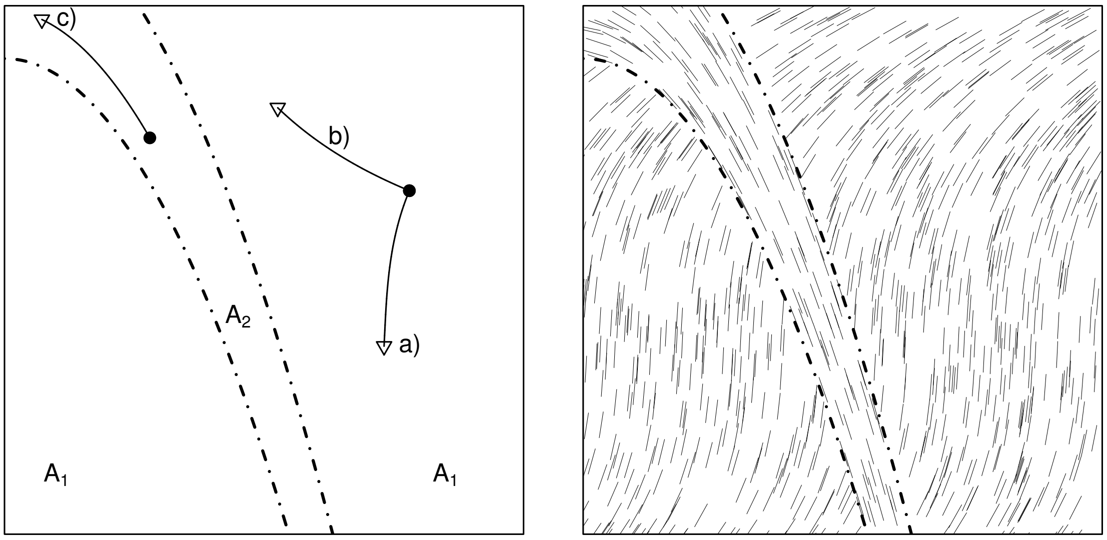


Figure 4.3: The situation of the simulated dataset with local anisotropy: Two independent subprocesses on A_1 and A_2 are considered (left). The curves represent the paths for which semivariograms are shown in figure 4.4. The sketch at the right visualizes the directions of local anisotropy.

or semivariogram specification. This is why this technique is not treated here.

4.2 A Simulated Dataset in Complex Geology

In the first simulated dataset, we consider a Gaussian process Z on $D = [0,1]^2$ made up of two independent subprocesses Z_1 on A_1 and Z_2 on A_2 with location-dependent directions of anisotropy $\phi(s)$ (see Figure 4.3 for illustration). We select a constant mean $m = 5$ and elliptical semivariograms with a piecewise linear kernel function and parameters $\theta_1^T = (\sigma_1^2, a_1, q_1, b) = (1, 0.2, 0.6, 0.6)$ on A_1 and $\theta_2^T = (0.8, 0.1, 0.4, 0.95)$ on A_2, and on A_2, we also add a nugget effect with parameter $\sigma^2_{\text{nug}(2)} = 0.4$. The directions of anisotropy on A_1 and A_2 are defined by two different polynomials and visualized in Figure 4.3.

A geologic setting that hosts such a process could for example be the following: Suppose that Z represents some soil property that essentially depends on the underlying rock. On A_2, geologically young river sediments with strong anisotropy down-stream and high local irregularity host the subprocess Z_2 with analogous properties ($q_2 = 0.4$, small range, nugget effect). On A_1, things are smoother ($a_1 = 2a_2$, no nugget effect), but oblique sediment layers with folding structures originate an anisotropy ($q = 0.6$).

A total of $n = 259$ locations was generated, 170 of which are uniformly distributed over A_1, and the remaining 89 are uniformly distributed over A_2, the density being higher in A_2. Then realizations of Z at these points were simulated using `random.dataset` and a `smp.data` object of 10 000 a priori nodes in order to guarantee a precise semivariogram approximation and hence a simulation that corresponds to the assumed semivariogram.

During semivariogram fitting, a set of 4 000 a priori nodes was used for quasi-Monte Carlo integration. Kriging was performed on a 60×60 grid using 10 000 nodes. The computation of all the kriging predictions presented below took about a quarter of an hour in total, and each minimization trial a few minutes, depending on the number of iterations needed.

For comparison with more sophisticated models, we fit global models with and without anisotropy, first of all the spherical semivariogram model using four different starting values. After about

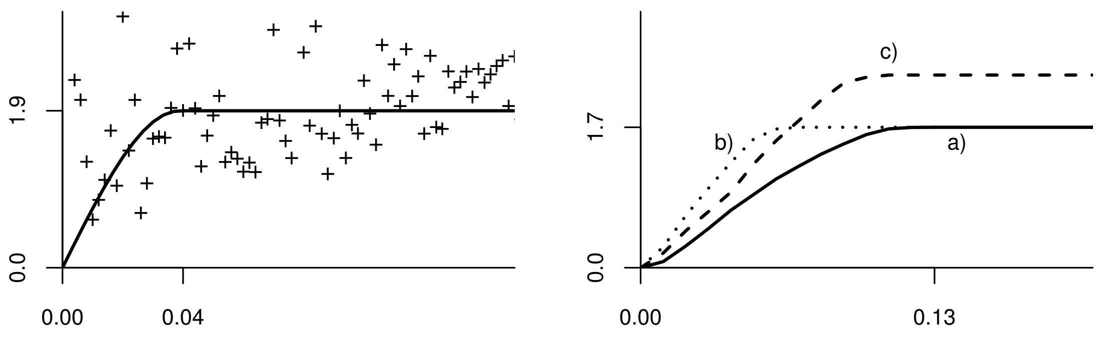


Figure 4.4: Left: Empirical semivariogram and the fitted spherical semivariogram with sill 1.90 and range 0.04.
Right: The fitted “true” semivariogram model, evaluated along the three paths shown in Figure 4.3: a) following the local direction of anisotropy in A_1; b) orthogonal to anisotropy in A_1; c) following anisotropy in A_2.

one minute of computation on a Pentium II processor, we see that all trials succeeded and give us the same estimates (if we consider four digits), namely a sill of 1.90 and a range of 0.39. See Figure 4.4 for a comparison with the empirical semivariogram.

Maybe we can fit a model with geometric anisotropy. Fitting an elliptical semivariogram with a piecewise linear kernel function (with $b = 0.7$) and fixed direction of anisotropy does not yield consistent results for any choice of direction: Sometimes the minimization is not successful, and when it is, then for different starting values and the same assumed direction of anisotropy it converges to different local minima, the mean squared errors however being similar to the one obtained for the spherical semivariogram (32.08).

Turning to semivariogram models with location-dependent anisotropy, we first study a global piecewise linear elliptical semivariogram with breaking point parameter $b = 0.95$ and the known covariable function ϕ. Different starting values all yield the same sill estimate 1.90, which is identical to the one estimated with the isotropic spherical semivariogram. However, mse possesses local minima, the estimate q in most cases being smaller than 0.5 and the range between 0.05 and 0.1, depending on q.

Finally we fit the “true” semivariogram model. Due to the high number of parameters, it has to be expected that the minimization of mse will take several steps, fixing some parameters at each step. Using 10 more or less reasonable random starting values, (the summary method of) `svm` yields the following output:

```
[...]
Semivariogram Parameters (successful trials only):
        BestEst.    Min.     Max
Sill1    1.69628 1.69628 1.69873
Range1   0.12574 0.12574 0.14814
Q1       0.32596 0.32596 0.66534
Break1   0.30757 0.30757 0.50781
Sill2    2.28308 2.28308 2.29611
Range2   0.03163 0.03163 0.05523
Q2       0.47124 0.30095 0.47124
Break2   0.75462 0.45890 0.75462
Nugget2  0.03184 0.02252 0.03184

Mean Squared Error (successful trials only):
     Min.  Mean  Max.
mse 31.91 31.91 31.91
```

```
Summary of Minimization Trials:
   Mean Sq.Err. Rel.MSE Rel.Param.Err. Iterations Code Best? Success?
1         31.91   1e-04          3.809         50    4             No
2         31.91   0e+00          0.000         18    2   the      Yes
3         31.91   1e-04          1.366         50    4             No
4         31.90  -2e-04          0.891         50    4             No
5         31.92   5e-04          7.077         24    3             No
6         31.91   0e+00          1.185         50    4             No
7         31.91   1e-04          1.932         50    4             No
8         31.91   1e-04          1.075         50    4             No
9         31.91   0e+00          2.231         50    4             No
10        31.91   1e-04          1.048         36    2     a      Yes

Starting parameter values:
       Sill1  Range1      Q1  Break1  Sill2  Range2      Q2  Break2  Nugget2
 [1,] 2.3062 0.15460 0.73792 0.87640 1.2719 0.28017 0.99181 0.84202 0.176962
 [2,] 1.2994 0.41529 0.48463 0.31665 1.7898 0.25864 0.58908 0.87567 0.031211
 [3,] 2.5496 0.44579 0.58845 0.74267 1.2101 0.22834 0.97514 0.17047 0.188768
 [4,] 1.3855 0.44847 0.73521 0.15957 1.2863 0.24004 0.78422 0.64345 0.025581
 [5,] 1.6079 0.17843 0.52848 0.79948 1.6496 0.21495 0.52997 0.72069 0.323776
 [6,] 1.9198 0.43815 0.61657 0.65926 1.0294 0.26836 0.61341 0.47269 0.198433
 [7,] 1.6782 0.43172 0.97461 0.48536 1.3657 0.19899 0.60631 0.16365 0.049348
 [8,] 2.9886 0.16557 0.96897 0.33380 1.3533 0.10782 0.80740 0.26245 0.418369
 [9,] 2.8188 0.15083 0.54031 0.57293 2.3126 0.30179 0.50593 0.64969 0.081212
[10,] 2.8444 0.18056 0.97890 0.53258 2.2426 0.19923 0.42710 0.49036 0.387770

Best Estimated Semivariogram Parameters:
  Sill1  Range1      Q1  Break1   Sill2  Range2      Q2  Break2 Nugget2
1.69628 0.12574 0.32596 0.30757 2.28308 0.03163 0.47124 0.75462 0.03184
```

The algorithm is succesful in only two trials, in the others the iteration limit was exceeded. Taking a closer look at the returned `svm` structure, it turns out that mse values are almost identical, so we should take into account all the 10 results:

```
> svmfit$est.params
       Sill1  Range1      Q1  Break1  Sill2     Range2      Q2  Break2  Nugget2
 [1,] 1.7084 0.13339 0.49602 0.92623 2.2264 8.3669e-02 0.53306 0.99993 0.153043
 [2,] 1.6963 0.12574 0.32596 0.30757 2.2831 3.1634e-02 0.47123 0.75461 0.031835
 [3,] 1.7001 0.14427 0.46613 0.72243 2.2808 6.5185e-02 0.72437 0.16815 0.031367
 [4,] 1.6980 0.16443 0.35430 0.15722 2.2853 5.9807e-02 0.57974 0.99876 0.025966
 [5,] 1.7220 0.11499 0.41886 0.73533 2.1539 1.6977e-06 0.34056 0.69226 0.257136
 [6,] 1.6957 0.15124 0.28647 0.66746 2.2881 6.0859e-02 0.77588 0.67157 0.023066
 [7,] 1.7010 0.16101 0.39077 0.43989 2.2744 9.2756e-02 0.47411 0.15869 0.048858
 [8,] 1.6970 0.11949 0.63302 0.32897 2.2891 6.4243e-02 0.66837 0.26322 0.016305
 [9,] 1.6989 0.12505 0.32679 0.57723 2.2641 1.0216e-01 0.31080 0.83476 0.069886
[10,] 1.6987 0.14814 0.66534 0.50781 2.2961 5.5225e-02 0.30095 0.45889 0.022517
```

The estimated sill parameters σ_1^2, σ_2^2 are practically identical, the range parameters are close together, and the axes ratios are “clustered”, but not yet consistent. For the next step, we only fix the sill parameters to $\sigma_1^2 = 1.70$ and $\sigma_2^2 = 2.27$, and the break parameters (which in any case are not of great effect) both to 0.7. We use two interesting starting parameter values that are motivated by the last results, and we obtain:

```
[...]
Semivariogram Parameters (successful trials only):
       BestEst.    Min.     Max
Range1  0.12765 0.12742 0.12765
Q1      0.55030 0.55030 0.56086
Range2  0.12376 0.12376 0.12376
Q2      0.33691 0.33691 0.33691
nugget  0.06222 0.06221 0.06222
```

```
Mean Squared Error (successful trials only):
     Min.  Mean  Max.
mse 31.91 31.91 31.91

Summary of Minimization Trials:
  Mean Sq.Err. Rel.MSE Rel.Param.Err. Iterations Code Best? Success?
1        31.91       0        0.01919         41    2     a      Yes
2        31.91       0        0.00000         37    2   the      Yes

Starting parameter values:
     Range1  Q1 Range2  Q2 nugget
[1,]    0.1 0.5    0.1 0.5   0.03
[2,]    0.2 0.9    0.1 0.5   0.20

Best Estimated Semivariogram Parameters:
 Range1      Q1  Range2      Q2  nugget
0.12765 0.55030 0.12376 0.33691 0.06222
```

Both trials yield practically the same result, we accept the parameter estimate suggested by `svm`.

See figure 4.4 for some sample plots of the fitted semivariogram along the paths shown in figure 4.3. Semivariogram plots along paths are a special feature of the `plot.sv` method; in this example, however, a special function had to be written for tracing the anisotropy.

Note that the plots in figure 4.4 are representative in the sense that any semivariogram evaluation along anisotropy direction within the are A_1 will look like graph a), and any plot orthogonal to anisotropy in A_1 will look like plot b), etc. This is due to the generic stationarity property of elliptical semivariograms.

Comparing fitted and true parameters,

	σ_1^2	a_1	q_1	b_1	σ_2^2	a_2	q_2	b_2	$\sigma_{\text{nug}(2)}^2$
fitted	1.70	0.13	0.55	(0.70)	2.27	0.12	0.34	(0.70)	0.06
true	1.00	0.20	0.60	0.95	0.8	0.10	0.40	0.95	0.40

we observe that the fitted nugget effect almost vanishes, the sill parameters were overestimated, axes ratios were estimated quite well, and the range in the smoother area A_1 was underestimated. Overestimation of the sill may be caused by an additional randomness due to integration errors.

Our next aim is to compare kriging predictions obtained with different fitted models. We use the following semivariograms:

- the true semivariogram,
- the fitted semivariogram consisting of piecewise linear elliptical semivariograms on A_1 and A_2 plus a nugget effect on A_2,
- the fitted spherical semivariogram with sill 1.90 and range 0.39, and
- a spherical semivariogram with sill 1.90 the (more reasonable) range 0.13 taken from the fitted semivariogram with local anisotropy.

Kriging results are shown in Figure 4.5. It can clearly be seen that both predictions based on spherical semivariograms do not reflect the strong anisotropies present in our dataset, whereas the fitted model with anisotropies leads to predictions that are very close to those obtained with the true semivariogram.

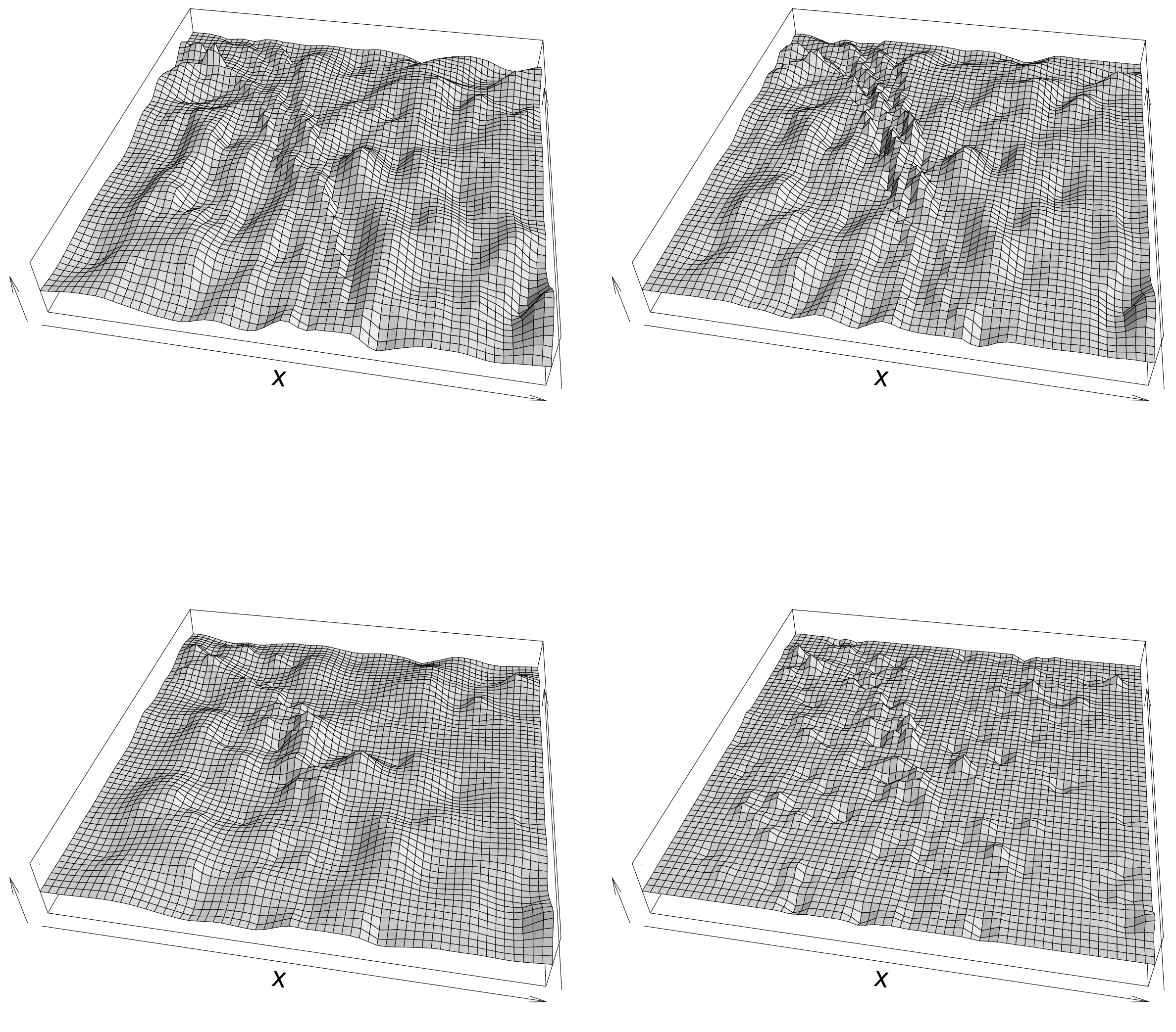


Figure 4.5: Kriging surfaces using the true semivariogram (top left), using the fitted generically stationary model (top right), using a fitted spherical semivariogram (sill 1.90, range 0.039; bottom right), and using a spherical semivariogram with sill 1.90 and range 0.13 (bottom left).

4.3 A Simulated Dataset with Trend

We wish to test if the techniques for modeling in the presence of trend as presented in Section 2.6 perform well. Therefore we will simulate a dataset, compare different semivariogram parameter estimates with the "true" parameters, and compare kriging results with those obtained for the "true" trend surface.

Let Y be a stationary Gaussian process on $D = [0,1]^2$ with a spherical semivariogram with parameter $\theta^T = (\sigma^2, a) = (1, 0.3)$. We define a function $f : D \to \mathbb{R}^4$ by

$$\begin{aligned}
f_1(s) &= 1,\\
f_2(s) &= s^{(1)},\\
f_3(s) &= 1 \text{ if } s^{(2)} > s^{(1)}, \text{otherwise } 0,\\
f_4(s) &= \mathbf{1}_{B^2((0.5,0.5),0.2)}(s),
\end{aligned}$$

where $s = (s^{(1)}, s^{(2)})^T \in D$. We put $\beta = (0,4,1,2)^T$ and define a process Z on D by

$$Z_s = \beta^T f(s) + Y_s$$

for all $s \in D$.

We consider $n = 100$ uniformly distributed points $t_1, \ldots, t_n \in D$ and simulate a realization $z = (z_1, \ldots, z_n)^T$ of $Z_{(n)} = (Z_{t_1}, \ldots, Z_{t_n})^T$ using the `random.dataset` function.

We will apply the following methods for fitting the spherical semivariogram model and for kriging:

i) We know the "true" parameters β and θ. Use them as reference values and for universal kriging.

ii) Estimate θ by $\widehat{\theta}$ using the restricted mean squared error, and use $\widehat{\theta}$ for universal kriging. Estimate β by $\widehat{\beta}$ with a generalized linear model.

iii) Apply the method presented (but not recommended) in Section 2.6.1: Estimate β by $\widetilde{\beta}$ in a linear model with uncorrelated errors. Fit the spherical semivariogram model to the residuals and get a parameter estimator $\widetilde{\theta}$. Do ordinary kriging with the residuals and the fitted semivariogram, and add predictions of the linear model.

Let Z_p^*, $\widehat{Z}_p$ and $\widetilde{Z}_p$, respectively, denote the kriging predictors for Z_p corresponding to these three cases.

In how far will parameter estimates for θ and β differ from the true values and from each other? And are there relevant differences between the kriging predictions that we will obtain, especially with respect to the first method?

Using the MoGeS package, we obtain the following results:

	estimate for β^T	**estimate for θ^T**
(1) "True" parameters	$(0.00, 4.00, 1.00, 2.00)$	$(1.00, 0.30)$
(2) Fitting with the rmse	$(-0.04, 4.17, 0.85, 1.72)$	$(1.22, 0.21)$
(3) Fitting the residuals	$(-0.85, 4.97, 1.50, 1.57)$	(0.98,0.19)

First of all, we notice that methods (2) and (3) yield significantly different trend parameters and also different sill parameters. These estimates differ from the true parameters β, θ.

At a first glance, there seems to be little difference between the kriging predictions plotted in figure 4.6. The plots of differences of predictors, $\widehat{Z}_p - Z_p^*$, $\widetilde{Z}_p - Z_p^*$ (figure 4.7), show however that the errors in the estimation of trend parameters have a very clear effect on prediction errors.

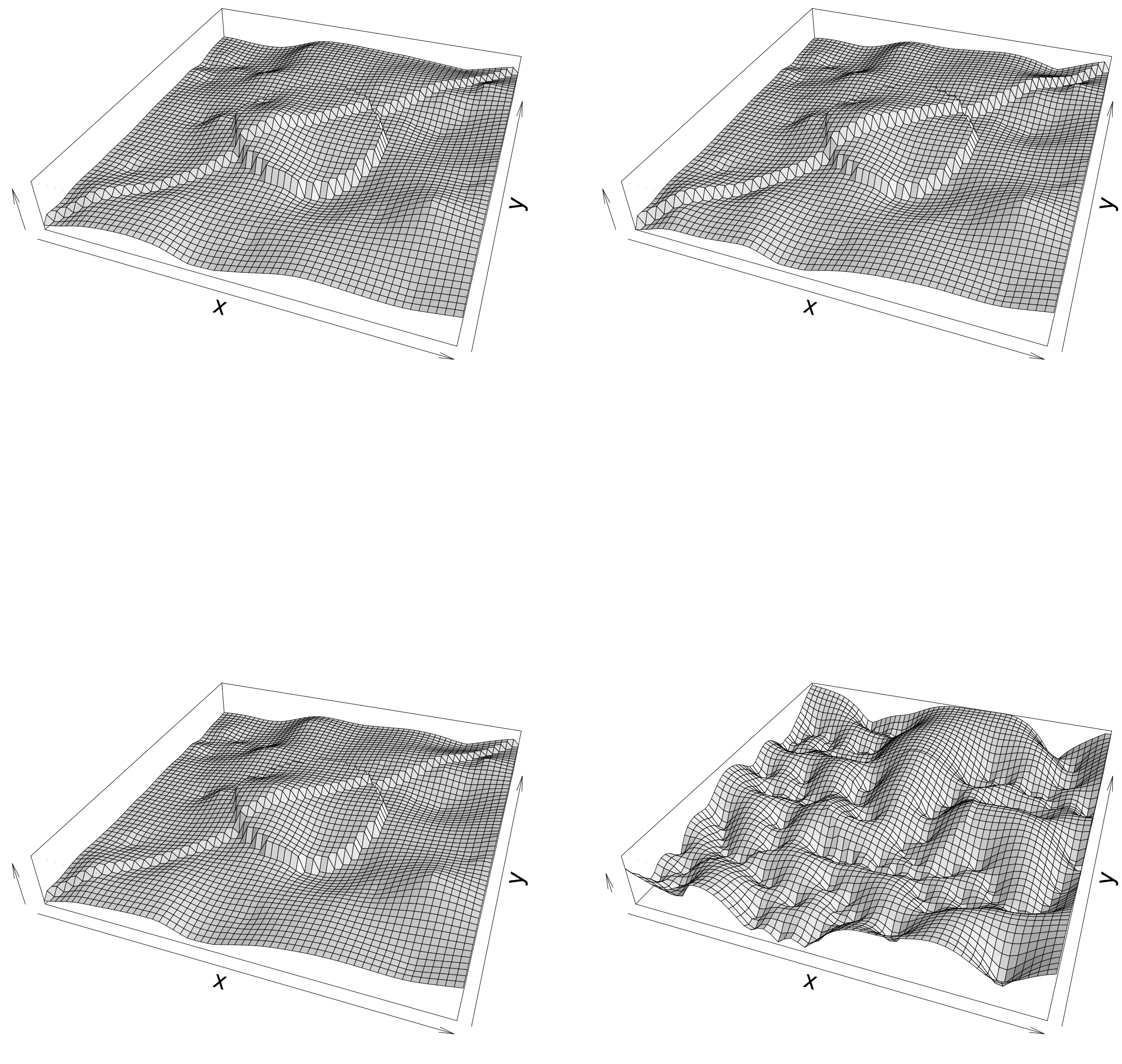


Figure 4.6: Kriging surfaces for the predictors Z_p^* (top left), $\widetilde{Z}_p$ (top right) and $\widehat{Z}_p$ (left bottom), and the kriging variance surface corresponding to $\widehat{Z}_p$.

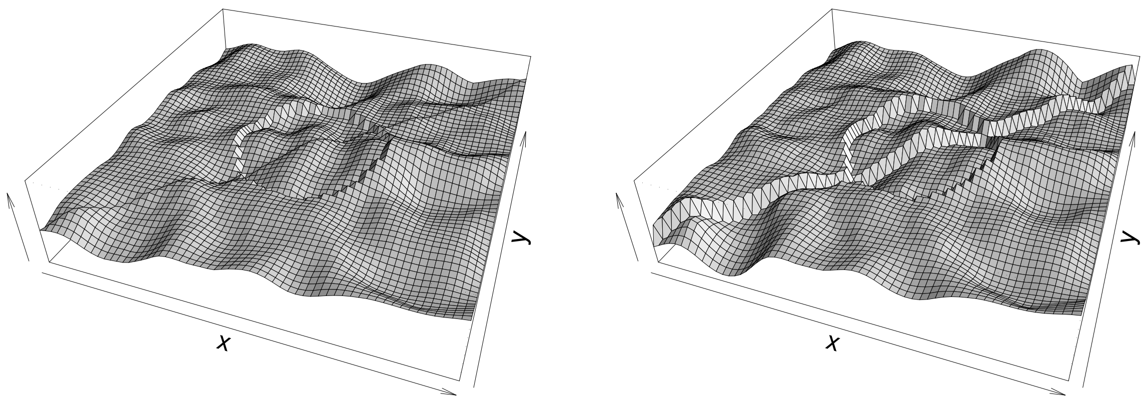


Figure 4.7: Differences between kriging predictors: $\widehat{Z}_p - Z_p^*$ (left) and $\widetilde{Z}_p - Z_p^*$ (right).

In this example the sill estimated with the restricted mean squared error is higher than the true sill and the one estimated based upon the residuals. In contrast, the author often observed that rmse yields smaller sills than mse when dealing with simulated processes with constant trend.

In this example, the results obtained with the semivariogram fitted in presence of trend using the restricted mean squared error function are slightly closer to the predictions Z_p^* that are based upon complete knowledge of the parameters. We cannot generalize this observation, but it makes us confident that estimations based on the restricted mean squared error work at least as good as semivariogram fitting to residuals with all its contradictory assumptions.

Chapter 5

Conclusions

The construction method presented in this work has shown to be a powerful instrument for incorporating knowledge of local geology as stored in a GIS into semivariogram models. Many situations of local anisotropy can be modeled using the class of elliptical semivariograms, which was studied in detail. Using the code provided in this work, such models were successfully fitted, and in an example situation it could be seen that the corresponding kriging results are also consistent with our knowledge of local anisotropy of the process, in contrast to isotropic or geometrically anisotropic semivariograms.

In the stationary case, the constructed covariograms are convolutions of weight functions. This facilitates the application of methods from Fourier analysis in future studies of induced covariograms.

Moreover, a method of semivariogram fitting in the presence of trend was presented, which is based on mean squared errors in a projected linear space. The practical results obtained make us confident that this method deserves further study.

The combination of R and AVENUE code generated for this work forms a flexible framework for geostatistical modeling using elliptical and other semivariograms within tightly coupled ArcView GIS and R data analysis environments.

Our study of the class of elliptical semivariogram models and of induced semivariogram models in general motivated the introduction of the concept of generic stationarity. This concept reflects our belief in the existence of natural laws that determine a process' distribution law depending on local geology. The less knowledge of local geology is necessary to determine the distribution law, "the more stationary" is the process. Thus, generic stationarity becomes a means for stationarizing instationarity conditional on local geology.

Appendix A

Documentation of Source Code

This appendix contains a documentation of R, C and AVENUE source code generated during this work. The description of R functions and objects follows the format that is commonly used in the R online help files, including information about usage, arguments, returned value and details. Sometimes various methods of the same object class are described in a single entry, or equivalent methods of different object classes are bundled.

All code described below was developed independently by the author, with the following exceptions: The C routine `sobseq` for the generation of Sobolev sequences was taken from Press et al. (1992), and the R function `eda.boxplot` for directional boxplots is based on the code of the standard R function `boxplot.default`.

Source code and binaries can be obtained from the author (e-mail: `ali@proforma.de`).

See also Section 3.3 for an overview of all object classes and functions.

A.1 Geostatistical Data: the `svm.data` Object Class

A.1.1 Basic Methods

```
#
# svm.data - Basic functions.
#
# Description:
#
#    Function to construct, coerce and check for ‘svm.data’ objects.
#
# Usage:
#
#    as.svm.data(x,...)
#    as.svm.data.list(d,name="noname",descr=NULL,id=NULL,
#       xyname=DEFAULT.XYNAME, xnames=DEFAULT.XNAMES, ynames=DEFAULT.YNAMES,
#       znames=DEFAULT.ZNAMES, fnames=DEFAULT.FNAMES, gnames=DEFAULT.GNAMES)
#    as.svm.data.svm.data(x)
#    is.svm.data(x)
#
# Arguments:
#
# x, d:          an object to be coerced or checked
#
# name, descr:   A name and description to be given to the ‘svm.data’ object.
#
# xyname etc.:   Names of columns or list components corresponding to the
#                ‘xy’, ‘z’, ‘f’ and ‘g’ components of an ‘svm.data’ object.
#
```

```
# Details:
#
#    'svm.data' objects are lists with the following components:
#
#    'xy':       An (n,2)-matrix: point coordinates of the measurement sites.
#    'z':        An (n,1)-matrix or n-vector: corresponding measured values.
#    'f':        An (n,.)-matrix giving variable values for the linear trend model.
#    'g':        An (n,.)-matrix giving local information needed by the semivariogram
#                function. (This may be, e.g., 0/1-values indicating if a
#                point is contained within a polygon, or orientations of the
#                reflief. (See, e.g., 'svfn.elliptical'.)
#    others:     Any other kind of data, such as names of the measurement sites etc.
#
#    'svm.data' objects have the following attributes:
#
#    "name":     A character string (max. one line) describing the 'svm.data'
#                object briefly.
#    "description": A more detailed description, possibly including formatting
#                characters.
#    "ID"        A (hopefully unique) numeric identifier of the 'svm.data' object
#                used before integrating for checking if an 'smp.data' object
#                fits to the 'svm.data'. (See 'QMC.int.ellipses'.)
#
#    The components 'xy', 'z', 'f' and 'g' must have the same length (if vectors)
#    or number of rows (if matrices).
#



#
# Read and write svm.data objects.
#
# Description:
#
#    Read and write svm.data objects from/to a file of "csv" or other format.
#
# Usage:
#
#    read.svm.data(file, file.format="csv",...)
#    write.svm.data(d,file)
#
# Arguments:
#
# d:              an 'svm.data' object (or a data.frame) to be written
#
# file:           file name
#
# file.format:    string specifying the output file format;
#                 either "csv", "csv2" or "table"
#
# ...:            further parameters to read.csv, read.csv2 or read.table
#
# Details:
#
#    These functions were created for reading and writing data exported from
#    a GIS or to be imported into a GIS.
#
#    In the current implementation, read.svm.data is not the inverse function
#    of write.svm.data in the sense that successive reading and writing
#    operations will not reproduce the original *variable names*. This is due
#    to the fact that files are read and written as data.frame structures,
#    but 'svm.data' objects are lists; its components 'f' and 'g' ---the
#    covariables for trend and generic stationarity specification--- may
#    have columns of the same name but different data, so when writing the
#    object, unique column names have to be created for the data.frame.
#
```

```
# See also:
#
#    read.csv, read.csv2, read.table, save, load
#
```

A.1.2 Simulation

```
#
# generate.grid
#
# Description:
#
#    generate a regularly spaced grid of points on a rectangle
#
# Arguments:
#
#    kx, ky   number of grid points in x and y direction, respectively
#    xmin, xmax, ymin, ymax   define a rectangle in R^2
#
# Usage:
#
#    generate.grid(kx,ky,xmin=0,xmax=1,ymin=0,ymax=1)
#
# Value:
#
#    A (kx*ky,2)-matrix containing kx*ky points regularly distributed over the
#    given rectangle (default rectangle: [0,1]^2).
#
```

```
#
# generate.random.points
#
# Description:
#
#    generate random points uniformly distributed over a rectangle
#
# Usage:
#
#    generate.random.points(n,xmin=0,xmax=1,ymin=0,ymax=1)
#
# Arguments:
#
#    n      number of points to be generated
#    xmin, xmax, ymin, ymax   a rectangle in R^2 (default: [0,1]^2)
#
# Value:
#
#    An (n,2)-matrix containing n points uniformly distributed over the
#    (interior of the) given rectangle.
#
# Details:
#
#    Uses the R (pseudo-) random number generator. By default, this is
#    the Marsaglia multiply-with-carry generator. See ‘RNGkind’.
#
```

```
#
# Generate Random Datasets
#
# Description:
#
```

```
#    Generate (pseudo-) random datasets corresponding to random fields
#    with a given covariance structure and mean or trend.
#
# Usage:
#
#    ramdom.dataset(x,...)
#    random.dataset.matrix(x,d,type="semivariogram",method="cholesky",
#         mean=0,trend=FALSE)
#    ramdom.dataset.sv(x,d,method="cholesky",mean=0,trend=FALSE,smp=NULL,...)
#    random.dataset.svm.data(x,sv,...)
#    random.dataset.default(x,sv,smp=NULL,
#         create.xy=(ifelse("xy" %in% names(d),"no","unif")),...)
#
# Arguments:
#
#    x:           A semivariogram object, ‘svm.data’ object, list or NULL,
#                 depending on the method to be used. (See Details.)
#
#    d:           A ‘svm.data’ object, list or NULL. (See details.)
#
#    type:        Character string indicating the kind of matrix passed to
#                 ‘random.dataset.matrix’. Either "semivariogram", "variogram",
#                 "covariance" or "covariogram".
#
#    method:      Determines the method of matrix decomposition for transfor-
#                 ming random numbers. Either "chol" or "svd". (See ‘chol’
#                 and ‘svd’).
#
#    mean:        If ‘trend=FALSE’, a stationary trend to be added to the
#                 random data. If ‘trend=TRUE’, coefficients of a determi-
#                 nistic linear trend to be added to the random data.
#
#    trend:       If TRUE, a linear trend ‘mean %*% d$f’ is added to the
#                 random data. If FALSE, a stationary mean value ‘mean’ is
#                 added.
#
#    smp:         A ‘smp.data’ object, an integer or NULL. If ‘smp’ is an
#                 integer, an appropriate ‘smp.data’ object with ‘smp’ samp-
#                 ling points is created for evaluating the co-/semivariogram
#                 function.
#
#    create.xy:   Indicates whether to generate measurement sites (‘xy’
#                 component) and, if yes, the method for doing this. Either
#                 "no", "unif" or "grid". The default value is "no", if ‘x’
#                 already has an ‘xy’ component, and ‘unif’ otherwise.
#
# Value:
#
#    A ‘svm.data’ object with randomly generated measurements (‘z’ component)
#    and possibly generated randomly or regularly distributed measurement
#    sites (‘xy’ component).
#
# Details:
#
#    Uses a Cholesky or singular-value decomposition of the covariance matrix
#    in order to transform independent N(0,1)-distributed random numbers,
#    and adds a linear trend or constant mean.
#
```

A.2 Semivariogram and Parameter Object Classes

A.2.1 param Object Class

```
#
# ‘param’eter objects - Basic functions.
#
# Description:
#
#    Creating, coercing to and checking for ‘param’eter objects,
#    and some other basic functions for ‘param’eter objects.
#
# Usage:
#
#    as.param(x,...)
#    as.param.param(x)
#    param(x,sem=NULL,nm=sem)
#    as.param.vector(x)
#    as.param.matrix(x)
#    is.param(x)
#    print.param(x,...)
#    as.vector.param(x,mode="any")
#    as.character.param(x)
#    alias(x,al)
#    unalias(x,al)
#    setnames(x,nm)
#
# Arguments:
#
# sem:      Parameter semantics. If NULL, the first ‘length(x)’ elements
#           of ‘DEFAULT.PARAM.SEMANTICS’ are used.
#
# nm:       Parameter names. (Defaults to ‘sem’.)
#
# Details:
#
#    ‘param’ creates a ‘param’ object with values from a vector ‘x’, semantics
#    ‘sem’ and parameter names ‘nm’, ignoring a possibly existing names
#    attribute of ‘x’.
#    ‘as.param.vector’, in contrast, interprets
#    the "names" attribute of the vector ‘x’ as parameter names and semantics.
#
#    ‘as.character.param’ creates a character vector of the form
#    ‘‘c("sill=1","range=2.7")’’.
#
#




#
# fixparam
#
# Description:
#
#    Overwrite parameter values with fixed parameter values.
#
# Usage:
#
#    fixparam(x,...)
#    fixparam.param(param,fix=NULL,na.ok=FALSE)
#    fixparam.vector(param,fix,na.ok=FALSE)
#
# Arguments:
#
# param:       a ‘param’eter object or (hopefully named) vector
```

```
#
# fix:          a ‘param’eter object or vector that will override elements
#               of ‘param’
#
# na.ok:        if TRUE, NA values in the result are ignored, otherwise a
#               warning will be displayed
#
# Value:
#
#    A ‘param’eter object or vector. See Details.
#
# Details:
#
#    If ‘param’ is a ‘param’eter object or named vector, ‘fixparam’ unifies
#    ‘param’ and ‘fix’ overwriting values in ‘param’ by those corresponding to
#    the same component name in ‘fix’.
#
#    If ‘param’ is a vector without names attribute, ‘fix’ must be of the same
#    length, and non-NA values in ‘fix’ overwrite the value at the
#    corresponding place in ‘param’.
#



#
# checkparam - Check for parameter validity.
#
# Usage:
#
#    checkparam(x,...)
#    checkparam.param(x,p.min,p.max)
#    checkparam.svfn(x,param)
#    checkparam.svc(x,param)
#    checkparam.csv(x,param)
#
# Arguments:
#
# p.min,
# p.max:        ‘param’ objects or named vectors specifying lower and
#               upper boundary of the (closed) intervals of valid parameters.
# param:        a ‘param’ object
#
# Value:
#
#    A named logical vector of length ‘length(x)’. Components are ‘TRUE’ if
#    they correspond to valid parameters, otherwise ‘FALSE’.
#
# Details:
#
#    Note that $[p.min,p.max]$ is considered a *closed* interval, i.e.
#    both ‘param’ = ‘p.min’ and ‘param’ = ‘p.max’ are valid parameter values.
#
#    ‘p.min’ and ‘p.max’ may contain infinite values (Inf,-Inf).
#



#
# Correct parameters
#
# Description:
#
#    Correct parameters according to given intervals.
#
# Usage:
#
```

```
#    correctparam <- function(x,...) UseMethod("correctparam")
#    correctparam.svc(x,param,...)
#    correctparam.svfn <- function(x,param,...)
#    correctparam.param(param,param.min=attr(x,"param.min"),param.max=attr(x,"param.max"),...)
#    correctparam.param <- function(param,param.min=rep(0,length(param)),
#       param.max=rep(Inf,length(param)),warn=TRUE,...)
#    svc.actualparam(svc, param, correction=TRUE, fix.param=NULL,
#       check.length=TRUE, ...)
#
# Value:
#
#    A 'param' object that is valid with respect to the restrictions
#    due to a semivariogram object's definition or 'param.min/.max'
#    arguments.
#
```

A.2.2 sv and Related Object Classes

```
#
# Semivariogram objects
#
# Description:
#
#    Methods for creating, coercing and printing 'svfn' objects.
#
# Usage:
#
#    svfn(x,name,descr=NULL, p.min=rep(0,p.num),p.max=rep(Inf,p.num),p.sem,
#       p.names=p.sem, g.names=NULL, needs.smp.data=FALSE, sg.names=NULL,
#       stationary=FALSE, isotropic=FALSE,...)
#    as.svfn(x,...) {
#    as.svfn.function(x,...)
#    print.svfn(x,partial=FALSE,short=FALSE,...)
#
# Arguments:
#
# x:            a function or a 'svfn' object
#
# name, descr: character strings giving a natural-language characterization
#               of the represented semivariogram model
#
# p.min, p.max: vectors that specify the valid parameter range; may be named
#
# p.sem:        a character vector specifying the semantics of each parameter
#
# p.names:      a character vector specifying the name of each parameter
#
# g.names:      a character vector specifying the names of 'g' components
#               (covariables)
#
# needs.smp.data: TRUE, if the function 'x' needs a priori nodes (an 'smp'
#               argument) for evaluation
#
# sg.names:     the names of covariables that have to be evaluated at the
#               a priori nodes ('g' components of an 'smp.data' object)
#
# stationary,
# isotropic:    specify whether the semivariogram model is stationary or
#               isotropic (for any choice of parameters)
#
# partial:      if TRUE, information on covariables is not printed
#
# short:        if TRUE, the output is more compact
#
```

```
# Value:
#
#    A ‘svfn’ object.
#
# Details:
#
#    ‘sv’ objects inherits from the class ‘sv’.
#
#    Typical parameter semantics are ‘‘sill’’, ‘‘range’’, ‘‘q’’ (axis ratio),
#    and ‘‘degree’’ (degree of smoothness). Others should only be introduced
#    if they have nothing to do with the mentioned semantics. (For example,
#    ‘svfn.elliptical.pwlinear’ specifies a ‘break’ semantics for the breaking
#    point of the integrand.)
#
# See also:
#
#    sv, svc, csv, param
#



#
# ‘svc’ objects
#
# Description:
#
#    Create, print and compute semivariogram models.
#
# Usage:
#
#    is.svc(x)
#    svc(svfn,nm=name(svfn),descr=description(svfn),
#       param.alias,fix.param,param.min,param.max,g.alias,arglist)
#    print.svc(x,short=FALSE,...)
#    compute.svc(x,d,ref=NULL,param,...)
#
# Arguments:
#
# svfn:        a ‘svfn’ object
#
# nm, descr:   string describing the object to be created
#
# param.alias: a named character vector specifying the parameter aliases
#              (i.e. substitutes for the original names) to be used
#              by the resulting ‘svc’ object; the ‘names’ attribute
#              establishes a correspondance to the original parameter names.
#
# fix.param:   a ‘param’eter object specifying the parameter values that
#              shall be constant
#
# param.min,
# param.max:   maximum and minimum parameter values for restricting the
#              minimization domain (usually not necessary)
#
# g.alias:     named character vector specifying the aliases for ‘g’
#              components (covariables)
#
# arglist:     list of arguments to be passed to the semivariogram function
#              in every call
#
# Value:
#
#    ‘svc’ returns an object of class ‘svc’.
#    ‘compute.svc’ returns a covariance or semivariogram matrix
#    (see ‘compute for details)
#
```

```
# Details:
#
# See also:
#
#    ‘svfn’, ‘csv’, ‘compute’, ‘param’
#



#
# ‘csv’ complex semivariogram object
#
# Description:
#
#    Create, coerce to and print ‘csv’ objects
#
# Usage:
#
#    csv(svclist,name=NULL,descr=NULL)
#    is.csv(x)
#    print.csv(x,short=TRUE,...)
#    as.csv(x,...)
#    as.csv.svc(x,...)
#    as.csv.svfn(x,...)
#    csv.insert(x,svc,newname,newdescr)
#    csv.delete(x,which,newname,newdescr)
#    compute.csv(x,d,ref=NULL,...)
#
# Value:
#
# A ‘csv’ object.
# ‘compute.csv’: a semivariogram or covariogram matrix.
#
# Details:
#
#    A ‘csv’ object is a list of ‘svc’ objects and has a name and
#    (natural-language) description.
#
# See also:
#
#    ‘svfn’, ‘svc’, ‘sv’, ‘compute’
#



#
# Evaluate (semivariogram) objects.
#
# Descriptions:
#
#    Functions for evaluating all kinds of semivariogram objects.
#
# Usage:
#
#    compute(x,...)
#    compute.svfn(x,param,fix.param,...)
#    compute.svc(x,d,ref=NULL,param,...)
#    compute.csv(x,d,ref=NULL,...)
#
# Arguments:
#
# x:           A semivariogram object.
# param:       An appropriate ‘param’eter object.
# fix.param:   A ‘param’eter object for fixing parameter.
# d:           An ‘svm.data’ object.
# ref:         An optional ‘svm.data’ for forming pairs of points.
# ...:         Further arguments to the semivariogram function.
#
# Value:
#
#    The semivariogram ‘x’ with parameters (param,fix.param) evaluated
#    at the pairs of points (d$xy[i,], ref$xy[j,]).
#
# Details:
#
#    Note that some of the components of ‘...’ may not be optional.
#    (E.g. ‘param’ in the case of ‘compute.csv’ or ‘smp’ if
#    ‘needs.smp.data(x)’ is TRUE.)
#



#
# penalty - Penalize parameter values out of range.
#
# Description:
#
#    Penalize parameter values that are out of range for a given parameter
#    range or semivariogram object.
#
# Usage:
#
#    penalty(x,...)
#    penalty.svfn(x,param,...)
#    penalty.svc(x,param,...)
#    penalty.csv(x,param,...)
#    penalty.param(x,p.min,p.max,incl=NULL,fac=1)
#
# Arguments:
#
#    x:       a semivariogram object giving an interval of valid parameters
#             or a ‘param’eter object to be checked.
#
#    param:   a ‘param’eter object to be checked.
#
#    incl, fac: specify how strong shall be penalized parameters out range;
#             don’t change them yet; for details, see source code
#
#    ...:     optional ‘incl’ and ‘fac’ arguments to be passed through to
#             ‘penalty.param’
#
# Value:
#
#    1, if ‘param’ is a valid set of paramters, otherwise a factor
#    greater than 1 increasing monotonically as the distance from
#    the valid area increases.
#


#
# Description:
#
#    Extract attributes from semivariogram objects
#
# Usage:
#
#    is.isotropic(x,...)
#    is.stationary(x,...)
#    semantics(x)
#    description(x)
#    name(x)
#
```

```
# See also:
#
#    svm.data, getrange
#
```

A.2.3 Special svfn Objects

```
#
# Elliptical semivariograms
#
# Description:
#
#    Functions and objects representing the elliptical semivariogram class
#
# Usage:
#
#    svfn.elliptical(d, ref=NULL, param, fix.param=NULL, smp, fun, fun.params,
#       global, strong.boundaries, extra.info, cov,arglist=NULL,...)
#    svfn.elliptical.bezier(param,...)
#    svfn.elliptical.linear(param,...)
#    svfn.elliptical.pwlinear(param,...)
#    svfn.elliptical.pwlinear.global(param,...)
#    svfn.elliptical.vargeometric(d,ref=NULL,param,...)
#    svfn.elliptical.geometric(param,orientation,arglist=NULL,...)
#
# Value:
#
#    If 'extra.info = FALSE', an approximated semivariogram (or covariance)
#    matrix.
#    Otherwise, a list with the components G, G.lo and G.hi representing
#    the approximate semivariogram (or covariance) values and error bounds
#    intervals (see 'QMC.int.ellipses' for details).
#
# Details:
#
# 'svfn.[var]geometric' are semivariogram models with geometrical anisotropy,
# the direction of anisotropy being modeled as parameter (svfn.var...) or as
# constant.
#
# See also:
#
#    QMC.int.ellipses, smp.data, param, svm.data, svfn
#


#
# Nugget effect
#
# Description:
#
#    Global or local nugget effect semivariogram function and objects.
#
# Usage:
#
#    svfn.nugget.template(d, ref=NULL, param, global, cov, extra.info, arglist, ...)
#    svfn.nugget.global(...)
#    svfn.nugget(...)
#
# Details:
#
# The local version 'svfn.nugget' requires the 'svm.data' object 'd' to contain
# an ``indicator'' covariable within 'd$g'.
#
# 'param' must be a 'param' object with one element with name ``nugget'' and
# semantics ``sill''.
```

```
#
# ‘svfn.nugget.template’ is not a ‘svfn’ object.
#

#
# Spherical semivariogram model.
#
# Usage:
#
#    svfn.spherical(d, param, ref=NULL,cov=FALSE,extra.info=FALSE,arglist,...)
#    svfn.spherical.iso.internal(h,param,data=NULL)
#

#
# Empirical semivariogram as a step function.
#
#    svfn.empirical(d, ref=NULL, param, arglist, cov=FALSE, breaks,
#       extra.info=FALSE, ...)
#
```

A.3 Quasi-Monte Carlo Integration and the Nodes Object Class

A.3.1 `smp.data` Object Class

R Code

```
#
# Generate quasi-random numbers
#
# Description:
#
#    Generate a (one- or two-dimensional) finite sequence
#    of Sobolev quasi-random numbers.
#
# Usage:
#
#    sobseq.init()
#    sobseq2d(n,fr=rbind(c(0,0),c(1,1)))
#
# Arguments:
#
# n:     number of points to be generated
#
# fr:    a frame rectangle (1st row=bottom left)
#
# Details:
#
#    C routines are called, including the code provided by
#    Press et al. (1992).
#
# References:
#
#    Press et al. (1992): Numerical Recipes in C. Cambridge Univ. Press.
#



#
# ‘smp.data’ - Using sampling point objects
#
# Description:
#
#    Methods for creating, checking for, printing and
#    plotting ‘smp.data’ objects.
#
# Usage:
#
#    smp.data(pt,Rmin=NULL,Rmax,N,fr=NULL)
#    is.smp.data(x)
#    print.smp.data(x,...)
#    plot.smp.data(smp,pts,add=F,rect.col="red",pts.col="blue",...)
#
# Arguments:
#
# x, smp:      ‘smp.data’ objects
#
# pt:          (n,2)-matrix of point coordinates or ‘svm.data’ object
#              (See Details.)
#
# Rmin,Rmax:   minimum and maximum radius of circles to be considered
#
# N:           number of sampling points to be created
#
# fr:          frame rectangle that shall contain all the sampling
#              points to be created
#
# pts:         additional points to be drawn (optional)
#
# add:         if TRUE (default), the plot will be added to an
#              existing one, otherwise a new plot will be startet
#
# rect.col:    color for plotting the subrectangles of the frame rectangle
#              (See Details.)
#
# pts.col:     color for drawing ‘pts’
#
# ...          additional arguments to ‘plot’ / ‘points’
#
# Details:
#
#    ‘smp.data’ creates ‘N’ Sobol a priori nodes and returns a ‘smp.data’
#    object.
#
#    ‘print.smp.data’ currently just prints the ‘smp.data’ summary.
#
# See also:
#
#    summary.smp.data, QMC.int.ellipses
#



#
# ‘summary.smp.data’ - Summarize sampling point objects
#
# Usage:
#
#    is.summary.smp.data(x)
#    print.summary.smp.data(x,...)
#    summary.smp.data(object)
#
# See also:
#
#    smp.data
```

C Code

```
/*
```

```
'generate_QMC_sampling_points_ext' generates a set of Sobol quasi-random
points within a given rectangle 'frame' AND within (open) circles
of radius R[1] around points given by 'pts'. Some additional information
about the created points is also passed as a result.

This 'not so obvious' approach has the aim of minimizing the number of
oracle calls that have to be made for each sampling point before
numerical integration.

pts             Centres of circles, stored as (x1,...,xn,y1,...,yn)
Npts            Number of points (xi,yi) stored in 'pts'
R[2]            Minimum and maximum radius of circles to be considered;
Nsmp            Number of sampling points to be generated
frame[4]        A reasonably large rectangle (ALL the circles of maximum
                radius should be ENTIRELY inside this rectangle!)
                The coordinates have to be stored as
                (xmin,xmax,ymin,ymax).
RectNum[2],     The 'frame' rectangle will be divided into RectNum[1] x
RectSize[2]     RectNum[0] rectangles of hight 'RectSize[0]' and width
                'RectSize[1]', each >= 2*R[1].
res_xy          The generated sampling points, stored as
                (x1,y1,x2,y2,...,xn,yn) [!!].
res_WithinConjMin,  These fields of length 'Nsmp' indicate whether a
res_WithinDisjMin,  sampling point is (a) inside the union of circles of
res_WithinDisjMax   minimum radium, (b) inside the dissection of circles
                of minimum radius, or (c) inside the dissection of circles
                of maximum radius, respectively. Especially the latter
                information is helpful for optimizing the numerical
                integration.
res_Area        The (estimated) total area occupied by the union of all
                the circles of maximum radius.
res_RectIndex   The starting index of sampling points inside a given
                rectangle (cf. 'RectNum' parameter).
                Note that these indices are 1-based, i.e. the first sampling
                point is referred to as 1, not as 0 as usual in C.
                This is a (RectNum[0] x RectNum[1])-matrix stored by rows.
res_dFrameArea
res_Ntotal


Note:

The minimum radius R[0] has merely 'statistical' relevance, i.e.,
it will be used for counting how many sampling points are within a
circle of minimum radius and thus knowing a 'worst case' precision
for extremely small radius.
*/


extern void __declspec(dllexport) sobseq_init_exp();
extern void __declspec(dllexport) sobseq_exp(double *m, double x[]);
extern void __declspec(dllexport) sobseq2d_exp(double *m, double frame[], double x[]);
```

A.3.2 `QMC.int.ellipses` and Related Functions

R Code

```
#
# QMC.int.ellipses - Quasi Monte Carlo integration of weight functions
#                    on dissections of ellipses
#
# Usage:
#
#    QMC.int.ellipses <- function(x,gx,ax,qx,
#       y=NULL,gy=NULL,ay=NULL,qy=NULL,
#       symmetric=is.null(y), diagonal=FALSE, fun="one", norm="none",
#       smp, so.names=NULL, extra.info=FALSE, fun.params=NULL,
#       strong.boundaries=FALSE, force.intunion=FALSE, ...)
#
# Arguments:
#
# x               centers of ellipses (2-vector or n-by-2-matrix)
# gx              orientation of the longer axis of ellipses (in [0,pi[)
# ax              longer axis of ellipses around 'x'
# qx              axis ratio
#
# y,gy,ay,qy      same as x etc. - ellipses to dissect with
#                 see also 'symmetric'
#
# symmetric       if TRUE, 'y','gy','ay','qy' will be set to be equal to
#                 'x' etc.
#
# diagonal        only applies when 'symmetric' is TRUE:
#                 if 'diagonal' is FALSE (default), do not integrate on the
#                 dissection of an ellipse with itself (the diagonal of
#                 a semivariogram is 0, these computations would be redundant)
#
# fun             weight function to be integrated in $\int fun(x,p)fun(y,p)dp$
#                 possible values are
#                 "one"       fun(x,.) = 1 on the ellipse around x, 0 otherwise
#                 "linear"    fun(x,.) decreases linearly towards the boundary
#                 "pwlinear"  fun(x,.) = 1 up to a distance of f.params*radius
#                             from the center, then decreases linearly
#                 "bezier"    Bezier function
#
# norm            normalization to be applied:
#                 "none"      no normalization
#                 "ellipse"   divide by the square roots of the ellipses' areas
#                 "integral"  divide by the square roots of $\int fun^2(s,p)dp$
#                             (s=`x[i,]',`y[j,]')
#
# smp             smp.data object providing an APPROPRIATE set of sampling
#                 points. The MC.int.ellipses does NOT check if these sampling
#                 points cover the ellipses (or their dissections) completely!
#
# so.names        if "strong boundaries" are used, specifies which of the
#                 columns of smp$g indicates if a sampling point lies inside
#                 the area the semivariogram is defined on
#
# extra.info      if TRUE, confidence intervals for the estimated integral
#                 are provided, based on the assumption that the sampling
#                 points are RANDOMLY uniformly distributed; i.e. the "true"
#                 confidence intervals can be expected to be much smaller
#
# fun.params      additional parameters to the passed to the function 'fun'
#
# strong.boundaries   if TRUE, the integrand is multiplied with,
#                 (smp$g[,so.names]==1) i.e. integrated only INSIDE the
#                 parameter set of the process.
#                 This forces 'filter.dissection' to be FALSE.
#
# force.intunion  (see code)
#
# Details:
#
#    Calls the C function `QMC_int_ellipses_exp'.
#
# See also:
#
#
```

```
#    smp.data, sobseq2d, .C
#
```

C Code

```
extern void __declspec(dllexport) QMC_int_ellipses_exp
   (double xy1[], double g1[], double a1[], double q1[], double *lpdN1,
    double xy2[], double g2[], double a2[], double q2[], double *lpdN2,
    double *lpdIsSymmetric, double *lpdCalcDiagonal,
    double *lpdGetExtraInfo, double *lpdForceIntUnion, double *lpdStrongBoundaries,
    char **lplpszFunction, double *lpdFunctionParameters, char **lplpszNormBy,
    double *smp_xy, double *smp_N, double *smp_N_total,
    double smp_indicator[],
    double *smp_frame,
    double *smp_rectnum, double *smp_rectsize, double *smp_nrect,
    double *smp_rectindexfrom, double *smp_rectindexto,
    double *smp_area,
    double integral[], double int_lo[], double int_hi[],
    double area[], double N_inside[],
    double *err)


extern void __declspec(dllexport) within_ellipse_exp(double res[],
    double pt[],
    double y[], double g[], double a[], double q[],
    double *npt, double *nell);

extern void __declspec(dllexport) within_ellipse_smp_exp(double *res,
    const double *smp_xy, const double *N_smp, const double *smp_frame,
    const double *smp_rectnum, const double *smp_rectsize,
    const double *smp_rectindexfrom, const double *smp_rectindexto,
    const double y[], const double g[], const double a[], const double q[],
    const double *nell);


extern void __declspec(dllexport) QMC_GetSMPRectangles_exp(
    double *lpdRectanglesX, double *lpdRectanglesY,
    const double *smp_xy, const double *N_smp, const double *smp_frame,
    const double *smp_rectnum, const double *smp_rectsize,
    const double *smp_rectindexfrom, const double *smp_rectindexto,
    const double *pt, const double *a);
```

A.4 Semivariogram Fitting and the Mean Squared Error

A.4.1 svm.mse

```
#
# Compute the (restricted) mean squared error
#
# svm.mse <- function(param,fix.param=NULL,
#    d,sv,trend=FALSE,smp=NULL,
#    param.names=NULL,param.semantics=NULL,
#    fix.param.names=NULL,fix.param.semantics=NULL,
#    mse.data=NULL, extra.info=FALSE, cov=FALSE, ...)
# calc.mse.data <- function(d,trend)
#
```

A.4.2 svm Object Class

```
#
# ‘svm’ - fitted semivariogram model
#
```

```
# Usage:
#
#    svm(d,sv,param,fix.param=NULL,trend=FALSE,smp=NULL,
#       print.level=0,iterlim=50,steptol=1e-5,mse.estimate=1,
#       trials=NULL,fun=NULL,extra.info=FALSE,...)
#    print.svm(x, digits = max(3,getOption("digits")-3),...)
#    summary.svm(object)
#    print.summary.svm(x, digits = max(3,getOption("digits")-3),...)
#
```

A.5 Kriging

```
#
# Kriging
#
# Description:
#
#    Perform ordinary or universal kriging.
#
# Usage:
#
#    svm.kriging(data,newdata,sv,param,G,method="qr",tol=1e-10,trend=FALSE,...)
#    predict.svm(object,data,newdata,...)
#
# Arguments:
#
# object:   ‘svm’ object returned by a call to ‘svm’
#
# data:     ‘svm.data’ object of observed data
#
# newdata:  ‘svm.data’ object of locations and covariables for prediction
#
# sv:       a semivariogram object
#
# param:    semivariogram ‘param’eter object
#
# G:        semivariogram matrix: ‘sv’ evaluated at ‘data’ (optional)
#
# method:   method for solving the kriging equations; "qr" (default) or "svd"
#
# tol:      tolerance for determining whether a matrix is singular
#           (default: 1e-10)
#
# trend:    if FALSE, ordinary kriging is performed, otherwise universal
#           kriging
#
# ...:      further arguments to the semivariogram object (and to
#           ‘svm.kriging’, when calling ‘predict.svm’)
#
# Value:
#
#    A ‘predict.svm’ object containing the predicted values, confidence
#    intervals, kriging variances and much more data; see details.
#
#
# Details:
#
#    In contrast to ‘svm,kriging’, ‘predict.svm’ extracts the semivariogram
#    model and the estimated ‘param’eter object from the ‘svm’ object ‘object’
#    and calls ‘svm.kriging’.
#
#    ‘tol’: If ‘method="qr"’, ‘tol’ is passed to ‘solve.qr’; otherwise it is
#    checked if the condition of the coefficient matrix with respect to the
#    spectral norm is greater than 1/tol.
#
#    The returned ‘predict.svm’ object is a list of the following components:
#
#    z:      values predicted at the locarions given in newdata$xy
#
#    z.lo, z.hi:  Confidence intervals for ‘z’ at a level of 95%
#
#    var:    kriging variances
#
#    z.coef: estimated coefficients of the observed data Z(t_1),...Z(t_n) for
#            each predicted location; a (HUGE!) (n,k)-matrix, where
#            k=nrow(newdata$xy).
#
#    f.coef: remaining coefficients from the kriging equations
#
#
# See also:
#
#    svm, solve.qr, svd
#
```

A.6 Empirical Semivariograms and Exploratory Data Analysis

```
#
# eda
#
# Description:
#
#    Exploratory Data Analysis for geostatistical data
#
# Usage:
#
#    as.direction(direc)
#    eda.plot.aoi(d,param,ellipse=FALSE,range=FALSE, add=FALSE, pch="+", ...)
#    eda.plot.positive.correlations(d, G, sill, add=FALSE,
#       draw.points=TRUE, pch="+", ...)
#    eda.plot.positive.correlations2(d, rg, add=FALSE, draw.points=TRUE, pch="+", ...)
#    eda.boxplot(z, notch = FALSE, varwidth = FALSE, notch.frac = 0.5,
#       boxwex = 0.8, border = par("fg"), col = NULL, log = "", pars = NULL, ...)
#    eda.boxplot.stats(d,direc,breaks=10,rel.breaks=TRUE,names=NULL,range=1.5,
#       indicatorfn=NULL,...)
#    eda.filter.pairs(xy, maxdist=NULL, direc=NULL, tol=NULL)
#    eda.semivariogram.cloud(d, intv="default", method="equidistant",
#       direc=NULL, tol=(1/18)*pi, extra.info=FALSE, trend=FALSE, mse.data)
#    eda.pairs(d,labels=NULL,indicatorfn=NULL,...)
#    eda.pairs2(d,labels=NULL,indicatorfn=NULL,...)
#    eda.plot.semivariogram(G, pts, ref=1, ref.name=NULL, G.lo=NULL, G.hi=NULL,
#       G.mc.lo=NULL, G.mc.hi=NULL, max.dist=NULL, max.value=NULL, xlab=NULL,
#       ylab=NULL, title=NULL, labels=NULL, add=FALSE, is.ref.center=FALSE,
#       draw.points=TRUE, cov=FALSE, pch=par("pch"), lwd=par("lwd"),
#       col=par("col"), lty=c("solid","dashed","dotted"))
#
# Details:
#
# ‘as.direc’ computes the orientation (an angle) of 2-dim. vectors.
#
# ‘eda.plot.aoi’ plots the ‘‘areas of influence’’ of the locations, i.e. circles
#  (or ellipses) of radius equal to the range (or half the range) of a semi-
#  variogram.
#
# ‘eda.plot.positive.correlations’ draws lines between points that have
# positive correlations, given a stationary semivariogram matrix C and its sill.
#
# ‘eda.boxplot’ draws a spatial boxplot using the ‘eda.boxplot.stats’ function
# instead of the original ‘boxplot.stats’. ‘eda.boxplot’ is practically the
# same code as the R function ‘boxplot’.
#
# ‘eda.boxplot.stats’ is a spatial adaption of ‘boxplot.stats’. It projects
# the coordinates on a line in the specified direction.
#
# ‘eda.filter.pairs’ selects pairs of points oriented in more or less the
# same direction.
#
# ‘eda.semivariogram.cloud’ plots empirical semivariograms using the moment
# method.
#
# ‘eda.pairs’ and ‘eda.pairs2’ are adapted to ‘scm.data’ objects and do the
# same as ‘pairs’ and (nicer:) ‘pairs2’ (code from help(pairs)).
#
# ‘eda.plot.semivariogram’: prefer ‘plot.sv’ instead.
#
```

A.7 Miscellaneous Code

```
#
# mask
#
# Description:
#
#    Generic function for extracting parts of objects
#
# Usage:
#
#    mask(x,...)
#    mask.vector(x,m)
#    mask.param(x,m)
#    mask.svm.data(x,m)
#
# Arguments:
#
# x:       An object from which to extract.
#
# m:       A vector that specifies the elements to be extracted;
#          must be of one of the many possible forms than can be used
#          to select elements of vectors or lists, e.g. a logical vector
#          of the same length as ‘x’ indicating which elements of ‘x’ to
#          extract, or a numeric vector giving the indices of elements
#          to be extracted.
#
# Value:
#
#     An object of the same class as ‘x’.
#



#
# unify
#
# Description:
#
#    A generic function for unifying object.
#
# Usage:
#
```

```
#    unify(x,...)
#    unify.list(x,y)
#    unify.vector(x,y,to.matrix=FALSE,name=NULL)
#    unify.matrix(x,y,chk=TRUE)
#    unify.param(x,y)
#    unify.svm.data(x,y,chk=TRUE)
#
# Arguments:
#
# x,y:         Objects to be unified. ‘y’ will be joint to ‘x’.
#              See Details for the object types allowed for ‘y’.
#
# to.matrix:   logical. If ‘TRUE’, the unified vector is converted to
#              an one-column matrix with column name ‘name’, otherwise
#              (the default) not.
#
# name:        character. See ‘to.matrix’.
#
# chk:         logical. ‘unify.matrix’: If ‘TRUE’ (the default) and
#              both ‘x’ and ‘y’ have column names, only columns with
#              the same names are unified.
#              ‘unify.svm.data’: ‘chk’ is applied to matrix components
#              in the same way as ‘unify.matrix’ does.
#
# Value:
#
#    An object of the same type as ‘x’.
#
# Details:
#
#    ‘unify’ does not perform a commutative operation.
#
#    ‘unify.matrix’ acts columnwise.
#
#    Possible combinations of object types:
#     ‘x’        ‘y’
#    list       list
#    matrix     matrix, (vector)
#    vector     vector, (matrix)
#    param      param, vector
#    csv        csv
#



#
# getrange
#
# Description:
#
#    Generic function that determines semivariogram ranges.
#
# Usage:
#
#    getrange(x,...)
#    getrange.param(x)
#    getrange.vector(x)
#    getrange.svfn(x,param,fix.param)
#    getrange.svc(x,param)
#    getrange.csv(x,param)
#
# Arguments:
#
# x:           Object that contains information on the range.
#              Must be a ‘param’ object, a named vector, NULL
#              or a semivariogram object.
# param,
# fix.param:   Parameter objects.
#
# Value:
#
#    The presumed range of a variogram feeded with the parameter
#    vector ‘x’.
#
# Details:
#
#    If no information could be obtained (i.e. no
#    element with name or semantics ‘range’ was found), ‘NA’
#    is the result.
#



#
# Check if points are contained by ellipses
#
# Description:
#
#    Check if points given by different types of objects
#    are within given (open) ellipses.
#
# Usage:
#
#    within.ellipse(x,...)
#    within.ellipse.matrix(x,y,g,a,q)
#    within.ellipse.smp.data(x,y,g,a,q)
#    within.ellipse.svm.data(x,y,g,a,q)
#
# Arguments:
#
# x:         an object from which to take the points
#
# y:         (k,2)-matrix (or 2-vector) specifying the centers of
#            ellipses
#
# g:         orientations of the ellipses’ longer axes (k-vector
#            of values in [0,pi[)
#
# a:         longer axis radius (k-vector)
#
# q:         axis ratios (k-vector)
#
# Value:
#
#    A logical matrix of n rows and k columns, the (i,j)th entry
#    being TRUE if pt[i,] (or whatever may be the i-th point
#    coordinate) contained in the open ellipse around y[j,]
#    with radius a[i] along g[i] and q[i]*a[i] along g[i]+pi/2.
#
# Details:
#
#    C routines are called.
#



#
# Compute the norm or condition number of a matrix or vector
#
# Usage:
#
#    norm(x,...)
#    norm.vector(x,norm=2)
#    norm.matrix(x,norm=Inf,symmetric=FALSE)
#    cond(x,norm=Inf,symmetric=FALSE)
#
# Arguments:
#
# x         a quadratic matrix or vector
#
# norm      a number specifying the norm (for vectors, values >0 and <=Inf
#           are allowed; for matrices, 1,2 and Inf only)
#
# symmetric TRUE, if the matrix is symmetric (this will not be checked!)
#
```

A.8 GIS Interface

```
'
' Step 1 - Export Data
'
' Export data from the active theme to a text file to be read by
' MoGeS / R.
' This script is just an interface to DoExportData.
'


'
' Step 2 - Model Specification
'
' Select global and local semivariograms.
' Identify variables (i.e. parameters or covariables).
' Assign fixed values to parameters.
' Specify the columns in a theme's table that contain the covariables.
'


'
' Step 3 - Semivariogram Fitting
'
' Fit the specified semivariogram to the selected data using MoGeS within R:
'
' Generate R code that prepares and performs the fitting.
' Generate a batch file for running R with the generated R file.
'    (DEPENDS ON THE ACTUAL PATH OF THE R TERMINAL PROGRAMM rterm.exe!!!!)
' Run the batch file.
'
' Small modifications of the code are necessary if the outpur of R shall be written
' to a .res-file. Currently the results are just displayed in the DOS shell's window.
'
' NOTE: THE MoGeS R ROUTINES MUST BE IN THE SAME DIRECTORY AS THE EXPORTED
' DATA IN ORDER TO BE ACCESSIBLE TO R!!!
'


'
' Step 4 - Kriging
'
' Perform kriging at given locations and using given covariable and parameter values.
'
' Steps:
' (1) Export the observed data and kriging locations and data.
'     The observed records must be identified by a numeric field
'     with the alias "observed"; values <>0 indicate that the record
'     contains an observed value. Kriging is performed based on these
'     observed values and at all (selected) points in the point shape.
' (2) Generate R code.
' (3) Run the R code.
' (4) After running this script, the user can add a table <filename.krg> into
'     his project and create an "event theme" that contains the kriging results
'     in the table's 'Z' field.
'
' Visualization in R is not yet available; for R's 'contour' and 'persp' (3D)
' functions, gridded data is needed. It is a bit complicated to generate the
' grid outside (or inside) ArcView, specify the covariables within ArcView and
' then do kriging... this is too much for just one script, the user will have to
' do that on his own...
'
' NOTE that all records that shall be exported must be SELECTED!!!
'



'
' CheckVariables
'
' Check for validity of ``variables'', i.e. parameters or covariables.
'
'
' Arguments:
'
' (1)  aliases: list of strings specifying the aliases of selected parameters
' (2)  values:  list of numbers: the values corresponding to these aliases
' (3)  names:   list of strings indicating the (original, i.e. unaliased and
'               unreduced) variable names
' (4)  semantics: same, but semantics
' (5)  vartype: either "parameter" or "covariable"
' (6)  dlgtitle: string: a title to be used for dialogs
'
' Returns:
'
' a list of two components:
' (1)  TRUE if all values were valid
' (2)  a list of elements of the form { AnAlias, AValue }, where Analias
'      is an element of argument (1) and AValue is the corresponding
'      (possibly corrected) value for the AnAlias variable
'
' Details:
'
' CheckVariables makes use of the parameter or covariable semantics in order
' to check for validity.
'
' If there are invalid values, a dialog will tell the user the valid range
' of parameter or covariable values.
'

'
' DifferentParameterSemantics
'
' Finds out whether selected parameters are of the same semantics.
' Returns a boolean value.
'

'
' DoExportData
'
' Export data from the active point theme to a text file to be read by R.
'
' Point coordinates are projected and stored in columns with names specified
' by the user. Furthermore, the user is asked which column contains the measurements
' (unless the data is used for kriging).
' The first and only argument to DoExportData indicates whether the data will
' be used for kriging (TRUE) or not (FALSE).
'
```

```
'
' EnterStartingValues
'
' Enter starting parameter values to be used for model fitting.
' Validity will be checked (CheckVariables)
'
' Arguments:
'
' (1)  list of strings: aliases of the selected parameters
' (2)  list of strings: original parameter names (before fixing
'      values, identifying parameters etc.) of the selected semivariogram
'      models
' (3)  list of strings: semantics corresponding to these names
'
' Returns a list with elements of the form { AnAlias, AValue }, where AnAlias is an
' alias string and AValue a valid parameter value corresponding to the
' parameter called AnAlias.
'


'
' FixVariables
'
' Assign fixed values to semivariogram parameters or covariables.
'
' Arguments:
'
' (1)  aliases
' (2)  names: unreduced list of variable names of the selected model
' (3)  semantics: same for parameter semantics
' (4)  vartype: either "parameter" or "covariable"
'
' Returns:
'
' (1)  unfixed aliases: remaining aliases from argument (1) that were not fixed
' (2)  fixed aliases: list of components of the form { AnAlias, AValue }
' (3)  has fixed variables: boolean
'
' Details:
'
' User is asked which variables shall be fixed, and to enter the corresponding
' values variable by variable.
'


'
' GetSemantics
'
' Returns the semantics string corresponding to a parameter or
' covariable alias.
'


'
' IdentifyVariables
'
' Asks the user if any variables (parameters or covariables) shall be
' identified, and if so, the user has to enter a common alias for
' both variables.
'



'
' QueryNewAlias
'
' Ask for a new variable alias; called by IdentifyVariables.
'
```

```
'
' R_AssignAlias
' R_AssignParam
' R_AssignSemivariogramComponent
' R_AssignSemivariogramModel
' R_ReadSvmData
'
' Write R code to a given LineFile, defining an alias variable,
' a ‘param’eter object, a ‘svc’ object or a ‘svfn’ object,
' or reading a ‘svm.data’ object from a file.
'


'
' SelectSemivariogramModel
'
' Display a dialog box for multiple semivariogram selection.
' Reuturns lists containing the selected semivariogram indices, parameter
' names, semantics and aliases, and covariable names, semantics and aliases.
'
' Note that the returned aliases are not the same as the names, because
' the semivariogram model’s name has to be added to the parameter and covariable
' names in order to get unique aliases.
'
' This script is called by the "Step 2 - Model Specification" script.
'


'
' shp2ascii
'
' Convert any shape file to an ASCII file, resolving polygon and
' line themes to point themes if necessary.
' Point coordinates are projected and added as columns to the
' theme’s table.
' Output file can be read by R; if conversion to a point shape
' has to be made, the temporary point shape file can be used by
' the ArcView/MoGeS interface.
'
```

List of Figures

List of Tables

List of Symbols

$\|\cdot\|$	Euclidian norm on $\mathbb{R}^d$, unless specified otherwise
$\mathbf{1}_A$	Characteristic function of a measurable set A
$\mathbb{1}_n$	The vector $(1,\dots,1)^T \in \mathbb{R}^n$
$\mathbb{1}_{n\times n}$	The matrix $(1)_{i,j=1,\dots,n}$
$\mathcal{B}^d$	Borel-σ-algebra in $\mathbb{R}^d$
$B^d(x,r)$	Open ball in $\mathbb{R}^d$ with center $x \in \mathbb{R}^d$ and radius $r > 0$
$B(s_1,\dots,s_N;r)$	$\bigcup_{i=1}^N B^2(s_i,r)$
$B^*(s_1,\dots,s_N;r)$	$\bigcup\{B^2(s_i,r)\cap B^2(s_j,r) : 1 \le i < j \le N\}$
$\mathrm{Ell}(x;g,a,q)$	Open ellipse in $\mathbb{R}^2$ with center $x \in \mathbb{R}^2$, longer radius a in an angle of g with the x axis, and axis ratio q
$\mathrm{GL}(n)$	group of all regular $n \times n$-matrices
I_n	$n \times n$ unit matrix
λ^d	Lebesgue-measure on $(\mathbb{R}^d, \mathcal{B}^d)$
$\lambda_i^d(A)$	$\sup_{s\in A} s^{(i)} - \inf_{s\in A} s^{(i)}$ (A bounded, $s = (s^{(1)},\dots,s^{(d)})^T$, $1 \le i \le d$) (If $A \subset \mathbb{R}^2$ is a rectangle, $\lambda_1^2(A)$ is its width and $\lambda_2^2(A)$ its height.)
mse	Mean squared error function for semivariograms (Defintion 2.5.1) and semivariogram models (Remark 2.5.2)
$\mathcal{N}$	Normalizing functional for weight functions (Remark and Definition 2.2.10)
$\rho(A)$	$\max\{\|\lambda\| : \lambda$ eigenvalue of $A\}$
rmse	Restricted mean squared error function in the presence of trend (Definition 2.6.4)

Bibliography

Antonov, I. A., and V. M. Saleev (1979): An economic method of computing lp_t-sequences. In: USSR Computational Mathematics and Mathematical Physics, 19: 252–256.

Bauer, H. (1978): Wahrscheinlichkeitstheorie und Grundzüge der Maßtheorie. De Gruyter, Berlin et al.

Billingsley, P. (1995): Probability and Measure. Wiley, New York.

Bivand, R. S. (1999): Integrating GRASS 5.0 and R: GIS and modern statistics for data analysis. In: Proc. 7th Scandinavian Research Conference on Geographical Information Science, pp. 111-127. Aalborg, Denmark. http://www.nhh.no/geo/Nyesider/GIBframe/GIB%{}20Abstract/1999-229.html.

Bivand, R. S., and M. Neteler (2000): Open Source geocomputation: using the R data analysis language integrated with GRASS GIS and PostgreSQL data base systems. In: Proc. 5th conference on GeoComputation, University of Greenwich, U.K. http://reclus.nhh.no/gc00/gc009.htm.

Bratley, P., and B. L. Fox (1988): Algorithm 659: Implementing Sobol's quasirandom sequence generator. In: ACM Transactions on Mathematical Software, 14(1): 88–100.

Cheng, J., and M. J. Druzdzel (2000): Computational investigation of low-discrepancy sequences in simulation algorithms for Bayesian networks. In: Proc. 6th Annual Conf. on Uncertainty in Artificial Intelligence, pp. 72-81. Morgan Kaufmann, San Francisco. http://www.pitt.edu/~druzdzel/psfiles/uai00a.ps.

Cressie, N. A. C. (1993): Statistics for spatial data. Wiley, New York.

ESRI Inc. (1996): Avenue. Customization and Application Development for ArcView GIS.

ESRI Inc. (2001): ESRI now shipping ArcGIS 8.1 Geospatial Analyst. The first geostatistical software integrating a powerful data exploration and surface creation environment within a GIS interface. ESRI press release, May 1, 2001, Redlands, California. http://www.esri.com/news/releases/01_2qtr/arcgis81-geostat.html.

Evans, M., and T. Swartz (2000): Approximating integrals via Monte Carlo and deterministic methods. Oxford University Press, Oxford et al.

Fischer, G. (1995): Lineare Algebra. 10th edition, Vieweg, Braunschweig et al.

Fotheringham, A. S. (ed.) (1994): Spatial analysis and GIS. Taylor & Francis, London.

Gasquet, C., and P. Witomski (1999): Fourier analysis and applications. Springer, New York et al.

Gentle, J. E. (1998): Random number generation and monte carlo methods. Springer, New York.

Goovaerts, P. (1997): Geostatistics for natural resources evaluation. Oxford University Press, New York et al.

Grabmüller, H. (1999): Grundlagen der Angewandten Analysis. Unpublished lecture notes, Friedrich-Alexander-Universität Erlangen-Nürnberg. http://www.am.uni-erlangen.de/~script/grabm/angmath.ps.

Kraas, F. (1993): Von der Reisebeschreibung zum Geographischen Informationssystem (GIS). Geographische Rundschau 45: 710–716.

Krekó, B. (1974): Optimierung. Dt. Verlag der Wiss., Berlin.

Longley, P. A., M. F. Goodchild, D. J. Maguire and D. W. Rhind (eds.) (1999): Geographical Information Systems. Vol. 1: Principles and technical issues. 2nd ed., Wiley, New York et al.

Niederreiter, H. (1992): Random number generation and quasi-Monte Carlo methods. CBMS–NSF regional conference series in applied mathematics. Soc. f. Industrial and Applied Mathematics, Philadelphia.

Nocedal, J., and S. J. Wright (1999): Numerical Optimization. Springer, New York et al.

Press, W. H., S. A. Teukolsky, W. T. Vetterling and B. P. Flannery (1992): Numerical recipes in Fortran 77. The Art of scientific computing. 2nd ed., Cambridge University Press, Cambridge et al.

Razavi, A. H. (1999): ArcView/Avenue developer's guide. OnWord Press, Albany (USA).

R Development Core Team (2000): An introduction to R. http://cran.r-project.org/doc/manuals/R-intro.pdf.

Rottmann, K. (1991): Mathematische Formelsammlung. BI-Wiss.-Verlag, Mannheim.

Stein, M. L. (1999): Interpolation of spatial data: some theory for kriging. Springer, New York.

van den Boogaart, K. G. (1999): A new possibility for modelling variograms in complex geology. Proceedings of StatGIS Klagenfurt 1999, to appear.

van den Boogaart, K. G. (2000): Fitting of variogram models in the present of trend. Unpublished, Freiberg.

Werner, J. (1992): Numerische Mathematik, Band 1. Vieweg, Braunschweig.

Index