Estimação de Modelos Geoestatísticos com Dados Funcionais usando Ondaletas

Gilberto Pereira Sassi

TESE APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA
OBTENÇÃO DO TÍTULO
DE
DOUTOR EM CIÊNCIAS

Programa: Programa de Pós-graduação em Estatística

Orientador: Profa. Dra. Chang Chiann

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro da CAPES e CNPq

São Paulo, 03 de março de 2016

Estimação de Modelos Geoestatísticos com Dados Funcionais usando Ondaletas

Esta versão da tese contém as correções e alterações sugeridas pela Comissão Julgadora durante a defesa da versão original do trabalho, realizada em 03/03/2016. Uma cópia da versão original está disponível no Instituto de Matemática e Estatística da Universidade de São Paulo.

Comissão Julgadora:

- Profª. Drª. Chang Chian (orientadora) - IME-USP
- Prof. Dr. Pedro Alberto Morettin - IME-USP
- Prof. Dr. João Ricardo Sato - CMCC-UFABC
- Prof. Dr. Luiz Koodi Hotta - IMECC-UNICAMP
- Profª. Drª. Sílvia Regina Costa Lopes - IM-UFRGS

Agradecimentos

Agradeço a minha família e meus amigos pelo suporte emocional nessa longa jornada de obter o título de doutor e a Profa. Dra. Chang Chiann pela orientação e pela paciência. Gostaria também de agradecer aos colegas de mestrado e doutorado Ariadne Nogueira, Bruno dos Santos, Daniel dos Reis e Joelson Campos por valiosas dicas de programação e redação. Agradeço em especial ao Joelson Campos que em momentos de dificuldade, cedeu o seu computador pessoal para realizar parte dos estudos de simulação contidos neste trabalho.

Resumo

SASSI, G.P. **Estimação de Modelos Geoestatísticos com Dados Funcionais usando Ondaletas**. 2016. 142 f. Tese - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2016.

Com o recente avanço do poder computacional, a amostragem de curvas indexadas espacialmente tem crescido principalmente em dados ecológicos, atmosféricos e ambientais, o que conduziu a adpatação de métodos geoestatísticos para o contexto de Análise de Dados Funcionais. O objetivo deste trabalho é estudar métodos de krigagem para Dados Funcionais, adaptando os métodos de interpolação espacial em Geoestatística. Mais precisamente, em um conjunto de dados funcionais pontualmente fracamente estacionário e isotrópico, desejamos estimar uma curva em um ponto não monitorado no espaço buscando estimadores não viciados com erro quadrático médio mínimo. Apresentamos três abordagens para aproximar uma curva em sítio não monitorado, demonstramos resultados que simplificam o problema de otimização postulado pela busca de estimadores ótimos não viciados, implementamos os modelos em MATLAB usando ondaletas, que é mais adequada para captar comportamentos localizados, e comparamos os três modelos através de estudos de simulação. Ilustramos os métodos através de dois conjuntos de dados reais: um conjunto de dados de temperatura média diária das províncias marítimas do Canadá (New Brunswick, Nova Scotia e Prince Edward Island) coletados em 82 estações no ano 2000 e um conjunto de dados da CETESB (Companhia Ambiental do Estado de São Paulo) referentes ao índice de qualidade de ar MP10 em 22 estações meteorológicas na região metropolitana da cidade de São Paulo coletados no ano de 2014.

Palavras-chave: krigagem, geoestatística, ondaletas, análise de dados funcionais, estatísica espacial, MATLAB.

Abstract

SASSI, G.P. **Estimation of Geostatistical Models with Functional Data using Wavelets**. 2016. 142 f. Tese - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2016.

The advance of the computational power in last decades has been generating a considerable increase in datasets of spatially indexed curves, mainly in ecological, atmospheric and environmental data, what have leaded to adjustments of geostatistcs for the context of Functional Data Analysis. The goal of this work is to adapt the kriging methods from geostatistcs analysis to the framework of Functional Data Analysis. More precisely, we shall interpolate a curve in an unvisited spot searching for an unbiased estimator with minimum mean square error for a pointwise weakly stationary and isotropic functional dataset. We introduce three different approaches to estimate a curve in an unvisited spot, we demonstrate some results simplifying the optimization problem postulated by the optimality from these estimators, we implement the three models in MATLAB using wavelets and we compare them by simulation. We illustrate the ideas using two dataset: a real climatic dataset from Canadian maritime provinces (New Brunswick, Nova Scotia and Prince Edward Island) sampled at year 2000 in 82 weather station consisting of daily mean temperature and data from CETESB (environmental agency from the state of São Paulo, Brazil) sampled at 22 weather station in the metropolitan region of São Paulo city at year 2014 consisting of the air quality index PM10.
Keywords: kriging, geostatistcs, wavelets, functional data analysis, spatial statistics, MATLAB.

Sumário

Capítulo 1

Introdução

Um dos pilares da evolução e da sobrevivência dos seres humanos é a capacidade de realizar predições baseadas nas informações disponíveis. Por exemplo, no ato de atravessarmos uma rua, observamos o movimento e sabemos o momento correto de atravessar, pois temos um banco de informações que nos foi passado desde a infância de como responder a essas situações; economistas diariamente fazem predições da economia considerando informações sobre a situação da atividade econômica passada e presente; o painel intergovernamental sobre mudanças climáticas da Organização das Nações Unidas (IPCC–UN) realiza estudos sobre a temperatura global no médio e longo prazo; antes de decidir se deve distribuir uma revista em uma nova banca, uma editora calcula a previsão de vendas do novo ponto de venda usando pesquisas de demanda e a previsão de vendas nas bancas próximas. Ou seja, tanto na vida cotidiana quanto na ciência, a habilidade de previsões é imprescindível.

Note que em diversos campos do conhecimento, as informações disponíveis são georeferenciadas, isto é, sabemos as coordenadas (latitude e longitude) de cada observação, e desejamos prever o valor dessa variável em um ponto não monitorado. Este tipo de estudo é chamado de análise geoestatística e seu uso é empregado em áreas como: dados hidrológicos (Hevesi *et al.*, 1992), mineração (Goovaerts, 1997; Journel e Huijbregts, 1978), estudos de qualidade do ar (Jerrett *et al.*, 2005). Vide Ecker (2003) para uma revisão histórica, metodológica e exemplos de aplicação de Geoestatística.

Em Geoestaística, merecem destaque dois cientistas pioneiros: o engenheiro de minas sul-africano Danie Gerhardus Krige (Minnitt e Assibey-Bonsu, 2014) que propôs um método de pre-

dição ótima com o objetivo de encontrar áreas com concentração de ouro (Krige, 1951) na década de 1950 e o matemático francês Georges François Paul Marie Matheron que formalizou o método proposto por Krige, chamando este método de predição ótima de *kriging* (krigagem em português), e fundou a Geoestatística com o seu trabalho (Matheron, 1963).

Com o avanço tecnológico e científico nas últimas décadas, as informações disponíveis não se restringem mais a valores escalares como no início do século XX. Imagens de satélite, curva de crescimento, curva de consumo, curva suavizada de temperatura medida a cada dez minutos são exemplos que deveriam ser tratados como curvas ao invés de valores escalares. Neste contexto, Ramsay e Dalzell (1991) introduziu o conceito de Análise de Dados Funcionais e desde então o seu uso se disseminou com aplicações em Modelos Lineares, Teoria de Resposta ao Item, Dados Dependentes, Análise Multivariada e outros (vide Ramsay e Silverman, 2002, para exemplos e aplicações de Análise de Dados Funcionais). Como referência para Análise de Dados Funcionais podemos citar: Ramsay (2006), Horváth e Kokoszka (2012) e Ferraty e Vieu (2006). Implementações de modelos para Dados Funcionais em MATLAB e em R estão documentadas em Ramsay *et al.* (2009).

Análogo ao caso escalar, existem aplicações, principalmente dados envolvendo ciências da terra e ambientais, em que as curvas no contexto de Dados Funcionais são espacialmente indexadas. Henao *et al.* (2010) e Henao *et al.* (2011) foram pioneiros em propor métodos de krigagem para Dados Funcionais. Salazar *et al.* (2015) apresentaram um preditor de krigagem para Dados Funcionais em que as curvas são funções densidade de probabilidade. Delicado *et al.* (2010) fazem uma revisão das técnicas de Geoestatística no contexto de Dados Funcionais. Bohorquez *et al.* (2015) propuseram uma metodologia e critérios de planejamento para encontrar o conjunto de localizações espaciais que minimiza a variância da predição ótima (krigagem para curvas) em um ponto não monitorado. Reyes *et al.* (2015) estenderam os métodos de krigagem em Dados Funcionais para a situação em que a função média de duas curvas não é necessariamente a mesma para duas localidades distintas. Nerini *et al.* (2010) propôs um método de interpolação espacial de curvas em que os parâmetros são funções definidas em um subconjunto de $\mathbb{R}^2$. A mesma abordagem foi apresentada na tese de doutorado de Henao (2009). Ignaccolo *et al.* (2014) introduziram um método de krigagem para Dados Funcionais adaptando *regression kriging* (vide Hengl *et al.*, 2007, para uma revisão sobre *regression kriging* em Geoestatística), em que variáveis exógenas,

potencialmente observadas como curvas, são permitidas. Caballero *et al.* (2013) forneceram uma solução para a krigagem para Dados Funcionais quando a condição de estacionaridade (verifique em Henao *et al.*, 2011, as condições de estacionaridade) não é satisfeita adaptando o modelo de krigagem universal (vide Stein e Corsten, 1991, para uma revisão de krigagem universal em Geoestatística) para Dados Funcionais.

De uma forma geral, as abordagens de Geoestatística para Dados Funcionais usam splines e estimativas paramétricas para matriz de covariância cruzada. Neste contexto, propomos três abordagens de krigagem para Dados Funcionais usando ondaletas e matriz de semivariograma: exibimos uma implementação usando ondaletas para a Krigagem Ordinária Funcional proposto por Henao *et al.* (2011); propomos o uso do semivariograma com um estimador não paramétrico ao invés da matriz de covariância cruzada para a Krigagem Tempo-Variante Funcional apresentado por Henao *et al.* (2010) usando ondaletas; e, finalmente, expomos um novo método usando os coeficientes das projeções das curvas da amostra funcional nos espaços de aproximação na Análise de Multirresolução em Ondaletas. Convém notar que uso de ondaletas é mais adequado para captar comportamentos localizados; o semivariograma, que pode ser estimado não-parametricamente, é a medida de variabilidade usada em Geoestatística para métodos de krigagem e os métodos de interpolação espacial introduzidos neste trabalho apresentam erro quadrático médio menor que os modelos presentes na literatura, conforme estudo de simulação apresentado no Capítulo 4. Além disso, implementamos todos os métodos em MATLAB.

Organizamos esse texto como segue: no Capítulo 1 fazemos uma introdução; no Capítulo 2 revisamos alguns conceitos necessários para a compreensão dos métodos de krigagem para Dados Funcionais apresentados no Capítulo 3; no Capítulo 4 comparamos os três métodos explicitados no Capítulo 3 via simulação; no Capítulo 5 ilustramos esses modelos usando duas aplicações: dados de temperatura média no ano de 2000 em 82 estações meteorológicas localizadas nas províncias marítimas canadenses (Nova Scotia, New Brunswick e Prince Edward Island) e dados do índice MP10 [1] da CETESB (Companhia Ambiental do Estado de São Paulo) no ano de 2014 em 22 estações meteorológicas na região metropolitana de São Paulo.

[1] MP10 é a quantidade em microgramas (μg) de partículas inaláveis com diâmetro inferior a 10 micrômetros (μm) em um m^3.

Capítulo 2

Conceitos Básicos

Neste capítulo, descrevemos brevemente os conceitos fundamentais de dados funcionais, geoestatística e análise de ondaletas. Não pretendemos fazer um explanação profunda, apenas uma revisão sucinta de cada tópico e, para leitores interessados, fornecemos referências para cada assunto. Este capítulo está organizado em três seções: na Seção 2.1 apresentamos dados funcionais; na Seção 2.2 resumimos os conceitos principais de geoestatística; e na Seção 2.3 revisamos os conceitos fundamentais de análise de ondaleta.

2.1 Dados Funcionais

Nesta seção, apresentamos brevemente os conceitos referentes à análise de dados funcionais. Para um estudo mais profundo deste tópico há vários livros na literatura: Ramsay (2006), Ramsay e Silverman (2002), Ramsay *et al.* (2009) , Horváth e Kokoszka (2012),
Ferraty e Vieu (2006), de Souza e Dias (2010) entre outros. Neste trabalho, seguiremos as definições e notações de Ferraty e Vieu (2006).

2.1.1 Variável Funcional, Dado Funcional e Conjunto de Dados Funcional

O progresso de ferramentas computacionais, em termos de processamento e armazenamento, faz-nos defrontar com banco de dados gigantesco. Por exemplo, é usual uma variável ser observada em vários pontos no intervalo (t_{min}, t_{max}) e, então, uma observação pode ser expressa como uma família aleatória $\{X(t_j)\}_{j=1,\ldots,J}$. Quando a distância entre t_j e t_{j+1} é suficientemente pequena po-

demos considerar a variável como proveniente de uma família contínua $\boldsymbol{\chi} = \{X(t) \mid t \in (t_{min}, t_{max})\}$. Além disso, há dados em que a natureza do estudo é nitidamente uma função contínua mesmo que a observação seja esparsa como, por exemplo, curva de crescimento e curva de consumo de energia elétrica [1]. Neste contexto, introduzimos os conceitos de variável funcional, dado funcional e conjunto de dados funcional.

Definição 2.1.1 (Variável Funcional)**.** Uma variável aleatória $\boldsymbol{\chi} = \{\boldsymbol{\chi}(t) \mid t \in \mathbb{R}\}$ é chamada de variável funcional se sua imagem estiver contida em

$$L^2(T) = \left\{ f : T \longrightarrow \mathbb{R} \mid \int_0^1 |f(t)|^2 dt \right\},$$

com $T \subset \mathbb{R}^n$. Uma particular realização $\chi = \{\chi(t) \mid t \in \mathbb{R}\}$ de uma variável funcional $\boldsymbol{\chi}$ é chamada de dado funcional.

Observações:

1. Quando $n = 1$, chamamos a variável funcional $\boldsymbol{\chi} = \{\boldsymbol{\chi}(t) \mid t \in \mathbb{R}\}$ de uma curva aleatória e o dado funcional $\chi = \{\chi(t) \mid t \in \mathbb{R}\}$ de uma curva.

2. É importante ressaltar que existe a possibilidade de considerar uma superfície aleatória se $T \subset \mathbb{R}^n,\ n \geq 2$.

3. Neste texto, usaremos a notação $\boldsymbol{\chi}(t)$ para denotar uma variável funcional e $\chi(t)$ para denotar um dado funcional proveniente de $\boldsymbol{\chi}(t)$.

Com o conceito de variável e dado funcional, podemos definir o conceito de conjunto de dados funcional.

Definição 2.1.2 (Conjunto de Dados Funcional)**.** Um conjunto de dados funcional ou amostra funcional $\chi_1(t), \ldots, \chi_n(t)$ são observações de n variáveis funcionais $\boldsymbol{\chi}_1(t), \ldots, \boldsymbol{\chi}_n(t)$ identicamente distribuídas.

[1] Ramsay e Silverman (2002) apresenta uma modelagem completa dos dados de curva de crescimento, uma aplicação de dados funcionais em teoria de resposta ao item e muitos outros exemplos.

2.2 Geoestatística

Nesta seção, apresentamos algumas definições básicas de geoestatística. Há vários clássicos de estatística espacial que podemos citar como referência: Diggle e Ribeiro (2007), Chiles e Delfiner (2009), Stein (2012), Haining (2003), Banerjee *et al.* (2014), Cressie e Cassie (1993) entre outros. Nesta seção, usamos as notações e definições conforme Banerjee *et al.* (2014).

Definição 2.2.1. Em um processo espacial $\boldsymbol{Y}(\boldsymbol{s})$, dizemos que $\boldsymbol{Y}(\boldsymbol{s}_1), \ldots, \boldsymbol{Y}(\boldsymbol{s}_n)$ são dados geoestatísticos se $\boldsymbol{Y}(\boldsymbol{s})$ é um vetor aleatório em uma localidade $\boldsymbol{s} \in \mathbb{R}^r (r = 2, 3)$, em que $\boldsymbol{s}$ varia continuamente em D subconjunto de $\mathbb{R}^r$ que contém um subconjunto mensurável com medida positiva.

A seguir mostramos como calcular a distância entre dois pontos na superfície da terra. Para maiores detalhes, vide Frederick Pearson (1990).

Calculando a distância na superfície da Terra

Para duas localidades próximas, calcular a distância euclidiana pode ser uma excelente aproximação. Entretanto, para pontos distantes, a curvatura da Terra é relevante e precisa ser levada em conta no cálculo da distância.

Afirmação 2.2.1. Seja $P_1 = (\theta_1, \lambda_1)$ e $P_2 = (\theta_2, \lambda_2)$ dois pontos na superfície da Terra em que $\theta_i, i = 1, 2$ representa a latitude e $\lambda_i, i = 1, 2$ representa a longitude . A distância entre P_1 e P_2 é dada por

$$\|P_1 - P_2\| = R \cdot \phi,$$

em que R é o raio da terra e ϕ satisfaz

$$\cos\phi = \sin\theta_1 \sin\theta_2 + \cos\theta_1 \cos\theta_2 \cos\left(\lambda_1 - \lambda_2\right),$$

Demonstração. A geodésica em uma esfera é o arco do maior círculo passando por P_1 e P_2, ou seja, o comprimento de arco do círculo com raio R e passando por P_1 e P_2. Os pontos P_1 e P_2 em

coordenadas cartesianas são obtidos por $u_1 = (x_1, y_1, z_1)$ e $u_2 = (x_2, y_2, z_2)$ com

$$\begin{cases} x_1 = R\cos\theta_1\cos\lambda_1 \\ y_1 = R\cos\theta_1\sin\lambda_1 \\ z_1 = R\sin\theta_1 \end{cases} \quad \text{e} \quad \begin{cases} x_2 = R\cos\theta_2\cos\lambda_2 \\ y_2 = R\cos\theta_2\sin\lambda_2 \\ z_2 = R\sin\theta_2 \end{cases}.$$

Note que $||u_1|| = ||u_2|| = R$,

e

$$\begin{aligned} ||u_1||||u_2||\cos\phi &= \langle u_1; u_2\rangle \\ &= x_1x_2 + y_1y_2 + z_1z_2 \\ &= R^2\cos\theta_1\cos\theta_2(\cos\lambda_1\cos\lambda_2 + \sin\lambda_1\sin\lambda_2) + R^2\sin\theta_1\theta_2 \\ &= R^2\cos\theta_1\cos\theta_2\cos(\lambda_1 - \lambda_2) + R^2\sin\theta_1\sin\theta_2. \end{aligned}$$

Assim, o ângulo entre P_1 e P_2 é a solução da equação (2.1)

$$\cos\phi = \cos\theta_1\cos\theta_2\cos(\lambda_1 - \lambda_2) + \sin\theta_1\sin\theta_2, \tag{2.1}$$

e a distância entre P_1 e P_2 é $R\phi$. □

Observação. Para calcular a distância $R\phi$ entre P_1 e P_2 usamos a *Mapping Toolbox* do MATLAB.

Ponderando que podemos calcular numericamente a distância entre dois pontos na superfície da terra, considere o processo estocástico $\{Y(\boldsymbol{s}) \mid \boldsymbol{s} \in D\}, D \subseteq \mathbb{R}^r$. Quando $r = 1$, há uma extensa literatura em séries temporais analisando este tipo de processo estocástico. Quando $r > 1$, dizemos que o processo estocástico é um processo espacial e existe um interesse especial quando $r = 2$ (Norte-Sul e leste-oeste) e quando $r = 3$ (Norte-sul, leste-oeste e nível acima do mar). Na prática, os dados serão uma realização parcial desse processo espacial em um conjunto finito de pontos $s_1, \ldots, s_n$ e desejamos predizer $Y(\boldsymbol{s})$ em $\boldsymbol{s} \notin \{s_1, \ldots, s_n\}$.

Antes de introduzir os conceitos fundamentais de estacionaridade e isotropia para um processo espacial em geoestatística, estabelecemos as seguintes notações:

i. $\mu(\boldsymbol{s}) = \mathrm{E}\left(Y(\boldsymbol{s})\right), \boldsymbol{s} \in D$;

ii. O processo espacial é Gaussiano se para quaisquer n pontos $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$ contidos em D, temos que $Y(\boldsymbol{s}_1), \ldots, Y(\boldsymbol{s}_n)$ tem um distribuição normal multivariada de ordem n.

Definição 2.2.2 (Estacionaridade)**.** Considere o processo espacial $\{Y(\boldsymbol{s}) \mid \boldsymbol{s} \in D\}$ com média $\mu(\boldsymbol{s}), \forall \boldsymbol{s} \in D$. Então dizemos que o processo espacial é

a. **Estritamente estacionário** se para quaisquer $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$ com $\boldsymbol{s}_i \in D, i = 1, \ldots, n$, e $\boldsymbol{h} \in \mathbb{R}^r$ com $\boldsymbol{s}_i + \boldsymbol{h} \in D, i = 1, \ldots, n$, a distribuição conjunta de $Y(\boldsymbol{s}_1), \ldots, Y(\boldsymbol{s}_n)$ é a mesma de $Y(\boldsymbol{s}_1 + \boldsymbol{h}), \ldots, Y(\boldsymbol{s}_n + \boldsymbol{h})$.

b. **Fracamente estacionário** se

   i) a média é constante, ou seja, $\mu(\boldsymbol{s}) = \mu, \forall \boldsymbol{s} \in D$;

   ii) a covariância depende apenas do vetor de separação $\boldsymbol{h}$, ou seja, $\mathrm{Cov}\left(Y(\boldsymbol{s}); Y(\boldsymbol{s} + \boldsymbol{h})\right) = C(\boldsymbol{h}), \forall \boldsymbol{s} \in D, \forall \boldsymbol{h} \in \mathbb{R}^r$ com $\boldsymbol{s} + \boldsymbol{h} \in D$.

c. **Intrinsecamente estacionário** se

   i) $\mathrm{E}\left(Y(\boldsymbol{s} + \boldsymbol{h}) - Y(\boldsymbol{s})\right) = 0, \forall \boldsymbol{s} \in D, \forall \boldsymbol{h} \in \mathbb{R}^r$ com $\boldsymbol{s} + \boldsymbol{h} \in D$;

   ii) $\mathrm{E}\left(Y(\boldsymbol{s} + \boldsymbol{h}) - Y(\boldsymbol{s})\right)^2 = \mathrm{Var}\left(Y(\boldsymbol{s} + \boldsymbol{h}) - Y(\boldsymbol{s})\right) = 2\gamma(\boldsymbol{h}), \forall \boldsymbol{s} \in D, \forall \boldsymbol{h} \in \mathbb{R}^r$ com $\boldsymbol{s} + \boldsymbol{h} \in D$.

Observações:

1. Estacionaridade fraca implica que a covariância do processo espacial entre duas localidades quaisquer pode ser sumarizada sem prejuízo por uma função $C(\boldsymbol{h})$ que depende apenas do vetor de separação $\boldsymbol{h}$.

2. Se um processo espacial é fracamente estacionário e Gaussiano, então ele é estritamente estacionário.

3. Na literatura de processo espacial, $2\gamma(\boldsymbol{h})$ é chamado de variograma, $\gamma(\boldsymbol{h})$ é chamado de semivariograma e $C(\boldsymbol{h})$ é chamado de covariograma.

4. Estacionaridade intrínseca é definida em termos dos dois primeiros momentos de $Y(\boldsymbol{s}+\boldsymbol{h}) - Y(\boldsymbol{s})$. Nenhuma suposição é feita em termos da distribuição conjunta de $Y(\boldsymbol{s}_1), \ldots, Y(\boldsymbol{s}_n)$.

5. Conhecendo o covariograma podemos obter o variograma, de fato

$$\begin{aligned} 2\gamma(\boldsymbol{h}) &= \mathrm{Var}\left(Y(\boldsymbol{s}+\boldsymbol{h}) - Y(\boldsymbol{s})\right) \\ &= \mathrm{Var}\left(Y(\boldsymbol{s}+\boldsymbol{h})\right) - 2\,\mathrm{Cov}\left(Y(\boldsymbol{s}+\boldsymbol{h}); Y(\boldsymbol{s})\right) + \mathrm{Var}\left(Y(\boldsymbol{s})\right) \\ &= 2C(\mathbf{0}) - 2C(\boldsymbol{h}), \end{aligned}$$

ou seja, $\gamma(\boldsymbol{h}) = C(\mathbf{0}) - C(\boldsymbol{h})$.

6. Estacionaridade fraca implica estacionaridade intrínseca, mas a recíproca não é verdade.

7. Para um processo intrinsecamente estacionário, o semivariograma $\gamma(h)$ é definido não positivo condicional, isto é, se $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n \in D$ e $a_1, \ldots, a_n \in \mathbb{R}$ com $\sum_{i=1}^{n} a_i = 0$, então $\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j \gamma(\boldsymbol{s}_i - \boldsymbol{s}_j) \leq 0$.
   De fato,

$$\begin{aligned} \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j \gamma(\boldsymbol{s}_i - \boldsymbol{s}_j) &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j \,\mathrm{E}\left[\left(Y(\boldsymbol{s}_i) - Y(\boldsymbol{s}_j)\right)^2\right] \\ &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j \,\mathrm{E}\left[\left(Y(\boldsymbol{s}_i)^2\right)\right] - \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j \,\mathrm{E}\left(Y(\boldsymbol{s}_i)Y(\boldsymbol{s}_j)\right) + \\ &+ \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j \,\mathrm{E}\left[\left(Y(\boldsymbol{s}_j)\right)^2\right] \\ &= -\,\mathrm{E}\left[\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j Y(\boldsymbol{s}_i)Y(\boldsymbol{s}_j)\right] \\ &= -\,\mathrm{E}\left[\left(\sum_{i=1}^{n} a_i Y(\boldsymbol{s}_i)\right)^2\right] \\ &\leq 0. \end{aligned}$$

Para analisar o caso em que estacionaridade intrínseca implica estacionaridade fraca introduzimos o conceito de ergodicidade para processos espaciais.

Definição 2.2.3 (Ergodicidade)**.** Dizemos que um processo espacial é ergódico se
$\mathrm{Cov}\left(Y(\boldsymbol{s}+\boldsymbol{h}); Y(\boldsymbol{s})\right) \longrightarrow 0$ quando $||\boldsymbol{h}|| \longrightarrow \infty$.

Em outras palavras, a Definição 2.2.3 estabelece que em um processo espacial ergódico o covari-

ograma tende a zero quando os pontos ficam distantes em $D \subseteq \mathbb{R}^r$. Com essa suposição podemos estabelecer condições em que um processo espacial intrinsecamente estacionário é fracamente estacionário conforme o Teorema 2.2.2 a seguir. Além disso, este teorema estabelece uma técnica para calcular o covariograma usando o semivariograma.

Teorema 2.2.2. *Seja* $\{Y(\boldsymbol{s}) \mid \boldsymbol{s} \in D\}$ *um processo espacial intrinsecamente estacionário e ergódico com variância constante e suponha que* $\lim_{||\boldsymbol{h}||\longrightarrow\infty} \gamma(\boldsymbol{h})$ *existe. Então, o covariograma é dado por*

$$C(\boldsymbol{h}) = \lim_{||\boldsymbol{u}||\longrightarrow\infty} \gamma(\boldsymbol{u}) - \gamma(\boldsymbol{h}).$$

Demonstração. Em primeiro lugar, observe que $C(\boldsymbol{h}) = C(\boldsymbol{0}) - \gamma(\boldsymbol{h})$. Como o processo é ergódico temos que $\lim_{||\boldsymbol{h}||\longrightarrow\infty} C(\boldsymbol{h}) = 0$ e então $C(\boldsymbol{0}) = \lim_{||\boldsymbol{h}||\longrightarrow\infty} \gamma(\boldsymbol{h})$, ou seja,

$$C(\boldsymbol{h}) = \lim_{||\boldsymbol{h}||\longrightarrow\infty} \gamma(\boldsymbol{h}) - \gamma(\boldsymbol{h}).$$

□

Outra propriedade importante em processos espaciais é o conceito de isotropia que será definida a seguir.

Definição 2.2.4 (Isotropia)**.** Considere um processo espacial com semivariograma $\gamma(\boldsymbol{h})$. Dizemos que o processo é isotrópico se

$$\gamma(\boldsymbol{h}_1) = \gamma(\boldsymbol{h}_2), \quad \text{para } \|\boldsymbol{h}_1\| = \|\boldsymbol{h}_2\|.$$

Se o processo não é isotrópico, dizemos que ele é anisotrópico.
Se o processo é intrinsecamente estacionário e isotrópico dizemos que ele é homogêneo.

Observações:

1. Para um processo isotrópico, $\gamma(\boldsymbol{h})$ é uma função de $||\boldsymbol{h}||$ e usamos a notação $\gamma(||\boldsymbol{h}||)$.

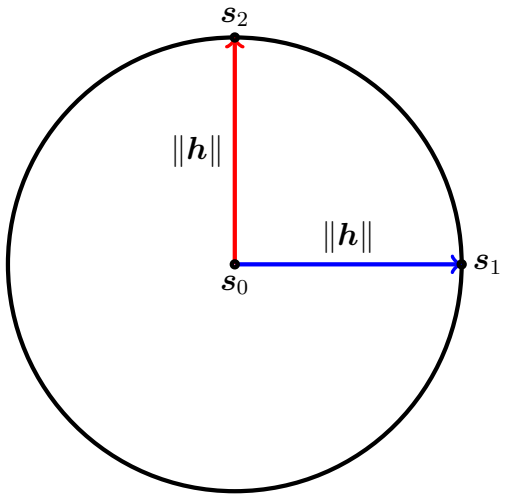


Figura 2.1: *Ilustração do conceito de isotropia: o semivariograma entre s_0 e s_1 e o semivariograma entre s_0 e s_1 é o mesmo.*

2. Processos espaciais isotrópicos são simples, facilmente interpretáveis e seus modelos paramétricos dependem de poucos parâmetros.

3. Para um processo espacial isotrópico, um estimador não-paramétrico para o semivariograma é o semivariograma empírico dado por

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

em que $N(h) = \{(i,j) \in \mathbb{N} \times \mathbb{N} \mid ||\boldsymbol{s}_i - \boldsymbol{s}_j|| = t\}$ e $|N(h)|$ é o número de pontos de $N(h)$.

2.3 Ondaletas

A ideia, tanto na análise de séries de Fourier, na aproximação de Taylor, na análise de ondaletas ou em qualquer outra base, é aproximar uma função por uma combinação de senos e cossenos, polinômios ou ondaletas. Particularmente no caso de ondaletas, desejamos aproximar uma função $f \in L^2(\mathbb{R}) = \{f : \mathbb{R} \to \mathbb{R} \mid \int_{-\infty}^{\infty} |f(u)|^2 du < \infty\}$ por uma combinação linear de dilatações binárias 2^j e translações diádicas $k2^{-j}$ de funções $\phi(x)$, chamada de função escala ou ondaleta pai, e / ou de funções $\psi(x)$, chamada de função ondaleta ou ondaleta mãe.

O campo de aplicação de ondaletas é extenso e inclui processamento de sinais, tratamento de imagens, compressão de dados, estimação de densidade, regressão não paramétrica e outros. Como referência podemos citar Boggess e Narcowich (2009), Daubechies *et al.* (1992), Meyer e Ryan

(1993), Morettin (2014), Meyer e Salinger (1995) e Chui (2014).

Nesta seção, expomos as ideias básicas sobre ondaletas, começando pela Seção 2.3.1 onde descrevemos a função ondaleta, também chamada de ondaleta mãe, e a função escala, também denominada ondaleta pai, e na Seção 2.3.2 onde apresentamos o conceito de análise de multirresolução.

2.3.1 Função Escala e Função Ondaleta

Considere o espaço $L^2(\mathbb{R}) = \{f : \mathbb{R} \to \mathbb{R} \mid \int_{\mathbb{R}} |f(u)|^2 du < \infty\}$ das funções quadrado integráveis em $\mathbb{R}$. A ideia é considerar dilatações (ou compressões) e translações de uma função ψ de modo a cobrir todo $\mathbb{R}$. Mais precisamente, consideramos

$$\psi_{j,k}(t) = 2^{\frac{j}{2}} \psi(2^j t - k),$$

em que $\{\psi_{j,k}(t) \mid j, k \in \mathbb{Z}\}$ é uma base para $L^2(\mathbb{R})$ e ψ é chamada de ondaleta mãe ou função ondaleta.

Suponha que $\{\psi_{j,k}(t) \mid j, k \in \mathbb{Z}\}$ é uma base ortonormal para $L^2(\mathbb{R})$ [2], isto é,

$$\langle \psi_{j,k}, \psi_{j,m} \rangle = \delta_{j,l} \delta_{k,m}, \quad j, k, l, m \in \mathbb{Z},$$

em que $\delta_{i,j} = \begin{cases} 1, \text{ se } i = j \\ 0, \text{ caso contrário} \end{cases}$ e

$$f(t) = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} d_{j,k} \psi_{j,k}(t), \tag{2.2}$$

em que a convergência é considerada em média quadrática. Adicionalmente, a série na equação (2.2) é chamada de série de ondaletas de $f(t)$ e $d_{j,k},\ j, k \in \mathbb{Z}$ são denominados coeficientes

[2] Meyer (1985) demonstrou que existe uma função $\psi(x)$ tal que $\{\psi_{j,k}(x) \mid j, k \in \mathbb{Z}\}$ é uma base ortonormal para $L^2(\mathbb{R})$.

de ondaletas ou coeficientes de detalhes e são calculados por

$$\begin{aligned} d_{j,k} &= \langle f(t), \psi_{j,k}(t) \rangle \\ &= \int_{-\infty}^{\infty} f(u)\psi_{j,k}(u)du. \end{aligned}$$

Neste contexto de ortogonalidade e analogamente a análise de série de Fourier, a relação de parseval é válida. Mais precisamente,

$$\int_{-\infty}^{\infty} f(u)du = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} d_{j,k}^2.$$

Além disso, as seguintes propriedades para a ondaleta mãe são válidas:

i. $\int_{-\infty}^{\infty} \psi(t)dt = 0$,

ii. $\int_{-\infty}^{\infty} |\psi(t)|dt < \infty$,

iii. $\int_{-\infty}^{\infty} |\psi(t)|^2 dt = 1$,

iv. Existe $r \in \{j \in \mathbb{Z} \mid j \geq 1\}$ tal que

$$\begin{aligned} &\int_{-\infty}^{\infty} t^j \psi(t)dt = 0, \quad j = 0, 1, \ldots, r-1; \\ &\int_{-\infty}^{\infty} |t^r \psi(t)|dt < \infty. \end{aligned}$$

O valor de r na propriedade iv. está ligado ao grau de regularidade (suavidade) de ψ: quanto maior o valor de r, mais suave será o ψ. Ademais, se a ondaleta tem suporte compacto, o valor de r está relacionado ao seu suporte (vide Härdle *et al.*, 1998, para maiores detalhes).

Poderíamos ter considerado translações (ou compressões) e dilatações de uma função ϕ, denominada função escala ou ondaleta pai, para aproximar $f(t)$. A função escala está intrinsecamente relacionada com a ondaleta mãe através das equações de dilatação descritas pela equações (2.3) e

(2.4).

$$\phi(t) = \sqrt{2}\sum_k l_k \phi(2t-k), \tag{2.3}$$

$$\psi(t) = \sqrt{2}\sum_k h_k \phi(2t-k). \tag{2.4}$$

em que h_k e l_k são associadas pela igualdade $h_k = (-1)^k l_{1-k}$, chamada de *quadrature mirror filter relation*. Além disso, temos que

$$l_k = \sqrt{2}\int_{-\infty}^{\infty} \phi(t)\phi(2t-k)dt,$$
$$h_k = \sqrt{2}\int_{-\infty}^{\infty} \psi(t)\phi(2t-k)dt,$$

em que l_k é denominado filtro passo-baixo e h_k é chamado de filtro passo-alto.

Exemplo 2.3.1. O exemplo mais antigo e mais simples é o caso Haar, introduzido inicialmente por Alfred Haar no trabalho Haar (1909). A ideia consiste em aproximar funções quadrado integráveis por funções escadas.

A ondaleta Haar é descrita por

$$\psi(t) = \begin{cases} 1, & 0 \geq t < \frac{1}{2} \\ -1, & \frac{1}{2} \geq t < 1 \\ 0, & \text{caso contrário} \end{cases},$$

para qual a função escala é dada por $\phi(t) = \begin{cases} 1, & t \in [0,1) \\ 0, & \text{caso contário} \end{cases}$.

Note que $\psi(t)$ e $\phi(t)$ tem suporte compacto e

$$\begin{aligned} \phi(t) &= \phi(2t) + \phi(2t-1) \\ &= \frac{1}{\sqrt{2}}\sqrt{2}\phi(2t) + \frac{1}{\sqrt{2}}\sqrt{2}\phi(2t-1), \end{aligned}$$

e

$$\begin{aligned}\psi(t) &= \phi(2t) - \phi(2t-1) \\ &= \frac{1}{\sqrt{2}}\sqrt{2}\phi(2t) - \frac{1}{\sqrt{2}}\sqrt{2}\phi(2t-1),\end{aligned}$$

ou seja, $l_0 = l_1 = \sqrt{\frac{1}{2}}$, $h_0 = l_0$ e $h_1 = -l_1$. Ademais, é possível verificar que $\{\psi_{j,k}(t) \mid j,k \in \mathbb{Z}\}$ é uma base ortonormal para $L^2(\mathbb{R})$. Na Figura 2.2 mostramos o gráfico da função escala e na Figura 2.3 a ondaleta Haar.

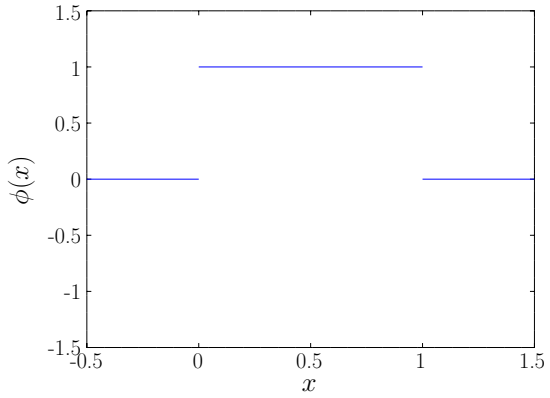


Figura 2.2: *Função escala Haar.*

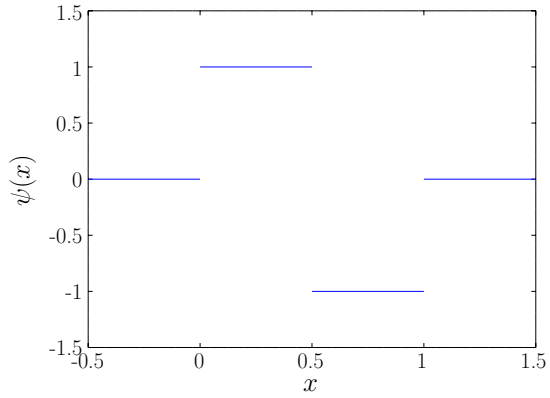


Figura 2.3: *Ondaleta Haar.*

Exemplo 2.3.2. Além das ondaletas Haar, as ondaletas Daubechies (*daublets*) são ondaletas ortogonais de suporte compacto bastante utilizadas. Essas ondaletas dependem de um parâmetro N que controla a suavidade de ϕ e ψ e são obtidas usando os filtros passa-baixo e passo-alto (vide Daubechies *et al.*, 1992, para maiores detalhes). Como ilustração, na Figura 2.5 mostramos o gráfico da ondaleta mãe para $N = 2, 3, 4, 5$ e na Figura 2.4 apresentamos a função escala para $N = 2, 3, 4, 5$.

2.3.2 Análise de Multirresolução

De acordo com Morettin (2014), a análise de multirresolução permite analisar os dados disponíveis em várias escalas de resolução. Formalmente, temos que:

Definição 2.3.1. Uma coleção de subespaços vetoriais $\{V_j \mid j \in \mathbb{Z}\}$ de $L^2(\mathbb{R})$ é uma análise de multirresolução se as seguintes condições são satifeitas:

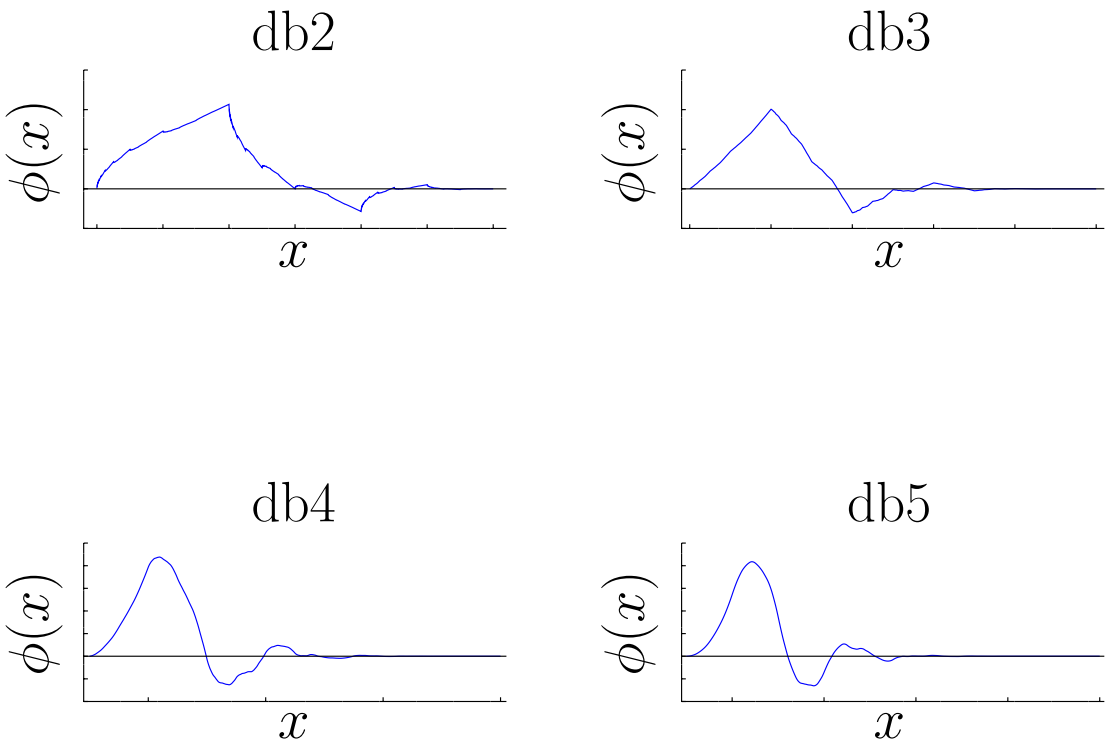


Figura 2.4: *Função escala para a família Daubechies quando* $N = 2, 3, 4, 5$.

(aninhamento) $V_j \subset V_{j+1}, \forall j \in \mathbb{Z}$;

(densidade) $L^2(\mathbb{R}) = \overline{\bigcup_{j\in\mathbb{Z}} V_j}$;

(separação) $\bigcap_{j\in\mathbb{Z}} V_j = \{0\}$;

(escolamento) $f(x) \in V_j \Leftrightarrow f(2x) \in V_{j+1}, \forall j \in \mathbb{Z}$.

Denominamos V_j de espaço de aproximação e j de nível de aproximação ou nível de resolução.

Em uma análise de multirresolução $\{V_j \mid j \in \mathbb{Z}\}$, aproximamos uma função $f(t) \in L^2(\mathbb{R})$ ao nível de resolução j pela projeção de $f(t)$ em V_j. A propriedade de aninhamento implica que quando passamos do nível j ao $j+1$, ganhamos informação sobre $f(t)$ e no limite $(j \to \infty)$ recuperamos completamente $f(t)$. Por outro lado, quando passamos do nível j ao $j-1$, perdemos informação sobre $f(t)$ e no limite $(j \to -\infty)$ a projeção é a função nula.

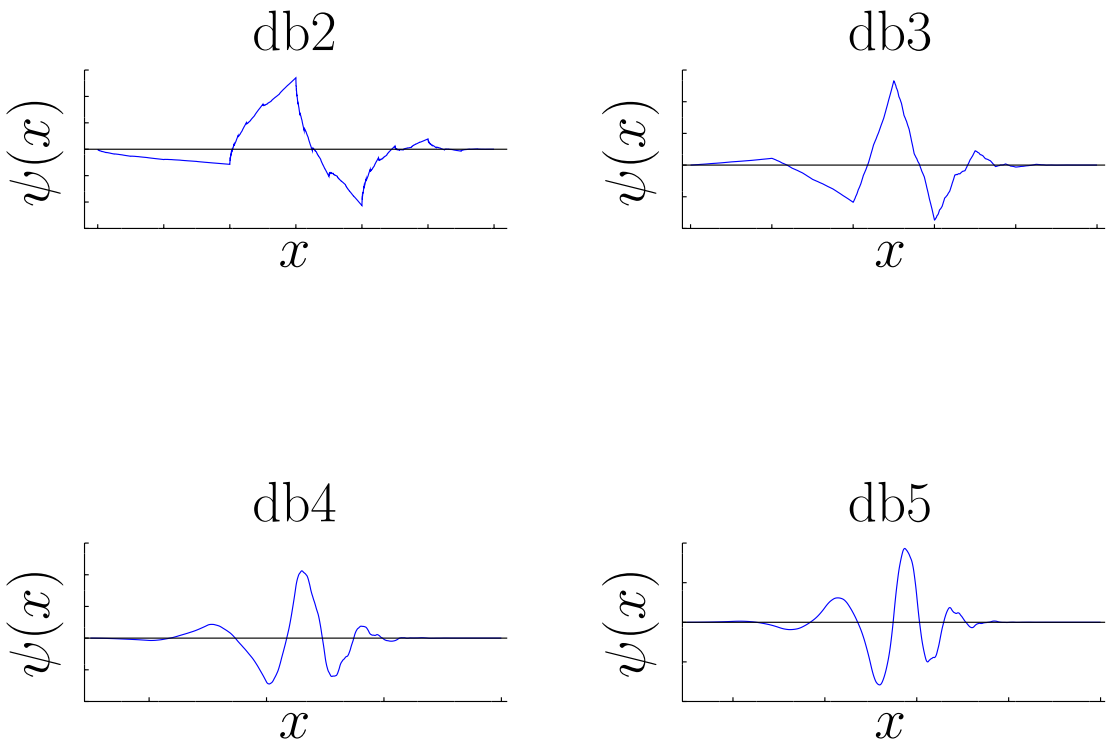


Figura 2.5: *Ondaleta mãe para a família Daubechies quando* $N = 2, 3, 4, 5$.

Pode-se provar que existe uma função $\phi \in L^2(\mathbb{R})$ tal que, $\{\phi_{j,k} \mid k \in \mathbb{Z}\}$ é uma base ortonormal de V_j (Mallat, 1989a,b). Ademais, a informação perdida quando passamos de V_{j+1} a V_j pode ser representada pelo subespaço W_j, ou seja, $V_{j+1} = V_j \oplus W_j$ e, consequentemente, $V_j = \bigoplus_{k=-\infty}^{\infty} W_k$. Pode-se mostrar que $\{\psi_{j,k} \mid k \in \mathbb{Z}\}$ é uma base ortornormal para W_j. Observe também que,

$$\begin{aligned} L^2(\mathbb{R}) &= \bigoplus_{j=-\infty}^{\infty} W_j \\ &= V_j \oplus \bigoplus_{k=j}^{\infty} W_k \\ &= \overline{\bigcup_{j=-\infty}^{\infty} V_j}, \end{aligned}$$

e, então, os conjuntos $\{\phi_{j,k} \mid j, k \in \mathbb{Z}\}$, $\{\psi_{j,k} \mid j, k \in \mathbb{Z}\}$ e $\{\phi_{j,k} \mid k \in \mathbb{Z}\} \cup \{\psi_{l,k} \mid l, k \in$

$\mathbb{Z}$ com $l \geq j\}$ são bases ortonormais para $L^2(\mathbb{R})$.

Nesse trabalho, optamos pelo uso das ondaletas Haar e Daubechies por terem suporte compacto e uma certa regularidade. Além disso, escolhemos aproximar uma curva através da projeção de $f(t) \in L^2(\mathbb{R})$ em V_j, isto é, fazemos uso apenas da função escala. No Capítulo 3, apresentamos as três abordagens de estimação de uma curva em um ponto não monitorado, em que usamos os conceitos apresentados neste capítulo.

Capítulo 3

Métodos de Krigagem para Dados Funcionais

Neste trabalho, estamos num contexto em que temos uma amostra funcional $\chi_{\boldsymbol{s}_1}(t), \ldots, \chi_{\boldsymbol{s}_n}(t)$, em que $\chi_{\boldsymbol{s}_i}(t), i = 1, \ldots, n, t \in [0, 1]$, é uma curva amostrada no ponto $\boldsymbol{s}_i = (\theta_i, \eta_i)$ da região em estudo com θ_i sendo a latitude e η_i a longitude, e o objetivo é estimar a curva $\chi_{\boldsymbol{s}_0}(t)$ em um ponto não monitorado $\boldsymbol{s}_0 = (\theta_0, \eta_0)$. Três abordagens diferentes de estimação são apresentadas: a primeira proposta é estimar $\chi_{\boldsymbol{s}_0}(t)$ por meio de uma combinação linear das curvas $\chi_{\boldsymbol{s}_i}(t), i = 1, \ldots, n$, conforme descrito na Seção 3.1; o segundo método de estimação propõe o uso de uma combinação linear em que os coeficientes são funções ao invés de constantes escalares conforme detalhado na 3.2; e, finalmente, na Seção 3.3 propomos um novo método de krigagem usando os coeficientes das expansões em ondaletas das curvas $\chi_{\boldsymbol{s}_i}(t), i = 1, \ldots, n$. A inovação dos métodos implementados nesta tese consiste no uso de ondaletas, no lugar de B-splines, para as duas primeiras abordagens além de propor o uso dos coeficientes da aproximação de $\chi_{\boldsymbol{s}_1}(t), \ldots, \chi_{\boldsymbol{s}_n}(t)$ para obter uma estimativa para $\chi_{\boldsymbol{s}_0}(t)$.

Em todos os modelos de krigagem deste trabalho, supomos que o processo estocástico espacial associado à amostra funcional é pontualmente fracamente estacionário e isotrópico. Mais precisamente,

i. $\mathrm{E}\left[\boldsymbol{\chi}_{\boldsymbol{s}}(t)\right] = m(t), \forall \boldsymbol{s} \in D, \forall t \in [0, 1]$;

ii. $\mathrm{Cov}\left(\boldsymbol{\chi}_{\boldsymbol{s}_1}(t), \boldsymbol{\chi}_{\boldsymbol{s}_2}(t)\right) = C(\|\boldsymbol{s}_1 - \boldsymbol{s}_2\|; t), \forall \boldsymbol{s}_1, \boldsymbol{s}_2 \in D, \forall t \in [0, 1]$ em que $\|\cdot\|$ é a geodésica

entre dois pontos na superfície da Terra conforme descrito na Seção 2.2 (vide Banerjee *et al.*, 2014, para maiores detalhes);

iii. $\frac{1}{2}\text{Var}\left(\boldsymbol{\chi}_{\boldsymbol{s}_1}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_2}(t)\right) = \gamma(\|\boldsymbol{s}_1 - \boldsymbol{s}_2\|; t), \forall \boldsymbol{s}_1, \boldsymbol{s}_2 \in D, \forall t \in [0,1]$. Para $t \in [0,1]$ fixo, $\gamma(\|\boldsymbol{s}_1 - \boldsymbol{s}_2\|; t)$ é chamada de variograma.

3.1 Krigagem Ordinária Funcional

Esse método de interpolação espacial para curvas pode ser considerado o modelo mais simples abordado neste trabalho e foi introduzido inicialmente por Henao *et al.* (2011), onde eles usaram B-splines para aproximar as curvas $\chi_{\boldsymbol{s}_1}(t), \ldots, \chi_{\boldsymbol{s}_n}(t)$.

A estimativa $\hat{\chi}_{\boldsymbol{s}_0}(t)$ da curva na localização $\boldsymbol{s}_0$ é combinação linear das curvas da amostra, isto é,

$$\hat{\chi}_{\boldsymbol{s}_0}(t) = \sum_{i=1}^{n} \lambda_i \chi_{\boldsymbol{s}_i}(t), \tag{3.1}$$

em que $\lambda_1, \ldots, \lambda_n$ são estimados por

$$\arg\min_{\lambda_1,\ldots,\lambda_n} \text{E}\left[\|\hat{\chi}_{\boldsymbol{s}_0}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\|^2\right] \text{ sujeito a } \text{E}\left[\hat{\chi}_{\boldsymbol{s}_0}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\right] = 0. \tag{3.2}$$

Em outras palavras, desejamos encontrar $\lambda_1, \ldots, \lambda_n$ tal que $\hat{\chi}_{\boldsymbol{s}_0}(t)$ seja não viciado com erro quadrático médio mínimo. Note que resolver o problema de otimização não linear (3.2) pode ser uma tarefa analítica e computacionalmente complexa, mas Henao *et al.* (2011) apresentaram uma alternativa viável para resolver este problema apresentado no Teorema 3.1.1.

Teorema 3.1.1 (Henao *et al.* (2011))**.** *Resolver o problema de otimização não linear*

$$\arg\min_{\lambda_1,\ldots,\lambda_n} \text{E}\left[\|\hat{\chi}_{\boldsymbol{s}_0}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\|^2\right] \textit{ sujeito a } \text{E}\left[\hat{\chi}_{\boldsymbol{s}_0}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\right] = 0,$$

é equivalente a resolver o sistema de equações lineares dado por

$$\begin{pmatrix} \gamma(\|\boldsymbol{s}_1-\boldsymbol{s}_1\|) & \gamma(\|\boldsymbol{s}_2-\boldsymbol{s}_1\|) & \dots & \gamma(\|\boldsymbol{s}_n-\boldsymbol{s}_1\|) & 1 \\ \gamma(\|\boldsymbol{s}_1-\boldsymbol{s}_2\|) & \gamma(\|\boldsymbol{s}_2-\boldsymbol{s}_2\|) & \dots & \gamma(\|\boldsymbol{s}_n-\boldsymbol{s}_2\|) & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \gamma(\|\boldsymbol{s}_1-\boldsymbol{s}_n\|) & \gamma(\|\boldsymbol{s}_2-\boldsymbol{s}_n\|) & \dots & \gamma(\|\boldsymbol{s}_n-\boldsymbol{s}_n\|) & 1 \\ 1 & 1 & \dots & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_n \\ \mu \end{pmatrix} = \begin{pmatrix} \gamma(\|\boldsymbol{s}_1-\boldsymbol{s}_0\|) \\ \vdots \\ \gamma(\|\boldsymbol{s}_n-\boldsymbol{s}_0\|) \\ 1 \end{pmatrix}, \quad (3.3)$$

em que $\gamma(h) = \int_0^1 \gamma(h,t)dt$ *é chamado de traço variograma.*

Demonstração. Em primeiro lugar, note que

$$\mathrm{E}\left[\boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\right] = m(t), \forall s \in D, \forall t \in [0,1],$$

e a restrição pode ser reescrita em

$$\begin{aligned} \mathrm{E}\left[\hat{\chi}_{\boldsymbol{s}_0}(t)\right] &= \mathrm{E}\left[\boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\right] \\ &= m(t). \end{aligned}$$

Mas observe que

$$\begin{aligned} \mathrm{E}\left[\hat{\chi}_{\boldsymbol{s}_0}(t)\right] &= \sum_{i=1}^{n} \lambda_i \, \mathrm{E}\left[\boldsymbol{\chi}_{\boldsymbol{s}_i}(t)\right] \\ &= m(t) \sum_{i=1}^{n} \lambda_i, \end{aligned}$$

e, consequentemente, a restrição $\mathrm{E}\left[\hat{\chi}_{\boldsymbol{s}_0}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\right] = 0$ é equivalente a $\sum_{i=1}^{n} \lambda_i = 1$.

Além disso, pelo Teorema de Fubini e sob a restrição de $\mathrm{E}\left[\hat{\chi}_{\boldsymbol{s}_0}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\right] = 0$, temos que

$$\begin{aligned} \mathrm{E}\left[\|\hat{\chi}_{\boldsymbol{s}_0}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\|^2\right] &= \mathrm{E}\left[\int_0^1 |\hat{\chi}_{\boldsymbol{s}_0}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_0}(t)|^2 dt\right] \\ &= \int_0^1 \mathrm{Var}\left(\hat{\chi}_{\boldsymbol{s}_0}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\right) dt. \end{aligned}$$

Como $\boldsymbol{\chi}_{\boldsymbol{s}_s}(t), s \in D$ é um processo estocástico pontualmente fracamente estacionário, $\sum_{i=1}^{n} \lambda_i =$

1 e $C(h;t) = C(0;t) - \gamma(h;t)$, temos que,

$$\begin{aligned}
\operatorname{Var}\left(\hat{\chi}_{\boldsymbol{s}_0}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\right) &= \operatorname{Var}\left(\chi_{\boldsymbol{s}_0}(t)\right) - 2\sum_{i=1}^{n}\lambda_i \operatorname{Cov}\left(\chi_{\boldsymbol{s}_i}(t); \chi_{\boldsymbol{s}_0}(t)\right) + \operatorname{Var}\left(\sum_{i=1}^{n}\lambda_i\chi_{\boldsymbol{s}_i}\right) \\
&= C(0;t) - 2\sum_{i=1}^{n}\lambda_i[C(0;t) - \gamma(\|\boldsymbol{s}_i - \boldsymbol{s}_0\|)] + \sum_{i=1}^{n}\lambda_i^2 C(0;t) + \\
&+ 2\sum_{j=1}^{n}\sum_{k=j+1}^{n}\lambda_j\lambda_k[C(0;t) - \gamma(\|\boldsymbol{s}_j - \boldsymbol{s}_k\|;t)] \\
&= C(0;t) - 2C(0;t)\sum_{i=1}^{n}\lambda_i + C(0;t)\left[\sum_{i=1}^{n}\lambda_i^2 + 2\sum_{j=1}^{n}\sum_{k=j+1}^{n}\lambda_j\lambda_k\right] + \\
&+ 2\sum_{i=1}^{n}\lambda_i\gamma(\|\boldsymbol{s}_i - \boldsymbol{s}_0\|;t) - 2\sum_{j=1}^{n}\sum_{k=j+1}^{n}\lambda_j\lambda_k\gamma(\|\boldsymbol{s}_j - \boldsymbol{s}_k\|;t) - \sum_{i=1}^{n}\lambda_i^2\gamma(0;t) \\
&= 2\sum_{i=1}^{n}\lambda_i\gamma(\|\boldsymbol{s}_i - \boldsymbol{s}_0\|;t) - 2\sum_{j=1}^{n}\sum_{k=j+1}^{n}\lambda_j\lambda_k\gamma(\|\boldsymbol{s}_j - \boldsymbol{s}_k\|;t) - \sum_{i=1}^{n}\lambda_i^2\gamma(0;t),
\end{aligned}$$

com $\gamma(0;t) = 0$.

Logo, temos que,

$$\begin{aligned}
\mathrm{E}\left[\|\hat{\chi}_{\boldsymbol{s}_0}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\|^2\right] &= \int_0^1 \operatorname{Var}\left(\hat{\chi}_{\boldsymbol{s}_0}(t) - \chi_{\boldsymbol{s}_0}(t)\right) dt \\
&= 2\sum_{i=1}^{n}\lambda_i\gamma(\|\boldsymbol{s}_i - \boldsymbol{s}_0\|) - 2\sum_{j=1}^{n}\sum_{k=j+1}^{n}\lambda_j\lambda_k\gamma(\|\boldsymbol{s}_j - \boldsymbol{s}_k\|) - \sum_{i=1}^{n}\lambda_i^2\gamma(0),
\end{aligned}$$

em que $\gamma(h;t) = \int_0^1 \gamma(h;t)dt$ é chamado de traço variograma.

Em outras palavras, desejamos resolver o seguinte problema de otimização linear

$$\begin{aligned}
&\arg\min_{\lambda_1,\ldots,\lambda_n} 2\sum_{i=1}^{n}\lambda_i\gamma(\|\boldsymbol{s}_i - \boldsymbol{s}_0\|) - 2\sum_{j=1}^{n}\sum_{k=j+1}^{n}\lambda_j\lambda_k\gamma(\|\boldsymbol{s}_j - \boldsymbol{s}_k\|) - \sum_{i=1}^{n}\lambda_i^2\gamma(0) \\
&\text{sujeito a } \sum_{i=1}^{n}\lambda_i = 1,
\end{aligned} \tag{3.4}$$

que pode ser resolvido usando multiplicadores de Lagrange. Mais precisamente, sejam

$$\begin{aligned}
f(\lambda_1,\ldots,\lambda_n) &= 2\sum_{i=1}^{n}\lambda_i\gamma(\|\boldsymbol{s}_i - \boldsymbol{s}_0\|) - 2\sum_{j=1}^{n}\sum_{k=j+1}^{n}\lambda_j\lambda_k\gamma(\|\boldsymbol{s}_j - \boldsymbol{s}_k\|) - \sum_{i=1}^{n}\lambda_i^2\gamma(0) \\
g(\lambda_1,\ldots,\lambda_n) &= \sum_{i=1}^{n}\lambda_i.
\end{aligned}$$

Então a encontrar a solução do problema de otimização linear (3.4) é equivalente a resolver o sistema de equações lineares dado por

$$\begin{cases} \nabla f(\lambda_1, \ldots, \lambda_n) = \delta \nabla g(\lambda_1, \ldots, \lambda_n), \\ g(\lambda_1, \ldots, \lambda_n) = 1, \end{cases}$$

isto é,

$$\begin{cases} 2\gamma(\|\boldsymbol{s}_i - \boldsymbol{s}_0\|) - 2\sum_{j=1}^{n} \lambda_j \gamma(\|\boldsymbol{s}_j - \boldsymbol{s}_i\|) = \delta, i = 1, \ldots, n, \\ \sum_{i=1}^{n} \lambda_i = 1. \end{cases}$$

Em forma matricial, temos que

$$\begin{pmatrix} \gamma(\|\boldsymbol{s}_1 - \boldsymbol{s}_1\|) & \gamma(\|\boldsymbol{s}_2 - \boldsymbol{s}_1\|) & \ldots & \gamma(\|\boldsymbol{s}_n - \boldsymbol{s}_1\|) & 1 \\ \gamma(\|\boldsymbol{s}_1 - \boldsymbol{s}_2\|) & \gamma(\|\boldsymbol{s}_2 - \boldsymbol{s}_2\|) & \ldots & \gamma(\|\boldsymbol{s}_n - \boldsymbol{s}_2\|) & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \gamma(\|\boldsymbol{s}_1 - \boldsymbol{s}_n\|) & \gamma(\|\boldsymbol{s}_2 - \boldsymbol{s}_n\|) & \ldots & \gamma(\|\boldsymbol{s}_n - \boldsymbol{s}_n\|) & 1 \\ 1 & 1 & \ldots & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_n \\ \mu \end{pmatrix} = \begin{pmatrix} \gamma(\|\boldsymbol{s}_1 - \boldsymbol{s}_0\|) \\ \vdots \\ \gamma(\|\boldsymbol{s}_n - \boldsymbol{s}_0\|) \\ 1 \end{pmatrix}$$

em que $\mu = -\frac{\delta}{2}$. □

Para construir o sistema linear do Teorema 3.1.1, precisamos de um estimador para o traço variograma e Henao *et al.* (2011) propuseram o seguinte estimador não paramétrico

$$\hat{\gamma}(h) = \frac{1}{2|N(h)|} \sum_{(i,j) \in N(h)} \int_0^1 (\chi_{\boldsymbol{s}_i}(t) - \chi_{\boldsymbol{s}_j}(t))^2 dt,$$

em que $N(h) = \{(i,j) \in \{1, \ldots, n\} \times \{1, \ldots, n\} \mid \|\boldsymbol{s}_i - \boldsymbol{s}_j\| \in (h - \epsilon, h + \epsilon)\}$ para $\epsilon > 0$ apropriado e $|N(h)|$ é o número de elementos de $N(h)$. Na Figura 3.1, ilustramos o conjunto $N(h)$.

Na prática, não observamos a curva inteira, apenas uma série temporal $\chi_{\boldsymbol{s}_i}\left(\frac{k-1}{2^L}\right), k = 1, \ldots, 2^L$. Para estimar cada curva $\chi_{s_i}(t), t \in [0,1]$, usamos regressão não-paramétrica (vide Härdle *et al.*, 1998; Ogden, 1997; Vidakovic, 2009, para maiores detalhes sobre regressão não

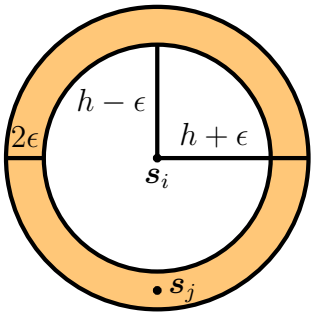


Figura 3.1: *Gráfico ilustrativo de* $N(h)$ *com* $\epsilon > 0$. *Note que* $(i,j) \in N(h)$, *pois* $h-\epsilon < \|\boldsymbol{s}_i - \boldsymbol{s}_j\| < h+\epsilon$.

paramétrica usando ondaletas). Mais precisamente, para s_i, observamos $Y_j = \chi_{\boldsymbol{s}_i}\left(\frac{j-1}{2^L}\right), j = 1, \ldots, 2^L$ e desejamos realizar a regressão descrita pela equação (3.5),

$$Y_j = f_{\boldsymbol{s}_i}(x_j) + \epsilon_j, \quad j = 1, \ldots, 2^L, \tag{3.5}$$

em que $x_j = \dfrac{j-1}{2^L}, j = 1, \ldots, 2^L$. A função estimada $\hat{f}_{\boldsymbol{s}_i}(x), x \in [0,1]$ pela equação (3.5) será a nossa curva de interesse, ou seja, $\hat{\chi}_{s_i}(t) = \hat{f}_{\boldsymbol{s}_i}(t), t \in [0,1]$.

Ao nível de precisão $J = \dfrac{L}{2}$, a função estimada $\hat{f}_{\boldsymbol{s}_i}(\cdot)$ é a projeção de $f_{\boldsymbol{s}_i}(\cdot)$ em V_J dada por

$$\begin{aligned}\hat{f}_{\boldsymbol{s}_i}(u) &= \left(P^J f_{\boldsymbol{s}_i}\right)(u) \\ &= \sum_{k\in\mathbb{Z}} c^J_{i,k}\phi_{J,k}(u),\end{aligned}$$

em que $P^J f_{\boldsymbol{s}_i}$ é a projeção ortogonal de $f_{\boldsymbol{s}_i}$ em V_J. Como escolhemos trabalhar com ondaletas de suporte compacto e $\chi_{\boldsymbol{s}_i}(t), i = 0, 1, \ldots, n$, tem também suporte compacto, existe $M \in \mathbb{N}$ tal que $c^J_{i,k} = 0$ para $|k| > M$.

Em forma matricial, temos que

$$\chi_{\boldsymbol{s}_i}(t) \approx \boldsymbol{c}_i^\top \boldsymbol{\phi}_J(t),$$

com $\boldsymbol{c}_i^\top = (c^J_{i,-M}, \ldots, c^J_{i,M})^\top$ e $\boldsymbol{\phi}^\top(t) = (\phi_{J,-M}(t), \ldots, \phi_{J,M}(t))^\top$. Pela propriedade da ortonormalidade

da base de ondaletas, o estimador não paramétrico para o traço variograma se resume a

$$\begin{aligned}\hat{\gamma}(h) &\approx \frac{1}{2|N(h)|}\sum_{(j,k)\in N(h)}\int_0^1 (\boldsymbol{c}_j-\boldsymbol{c}_k)^\top \boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J^\top(t)\,(\boldsymbol{c}_j-\boldsymbol{c}_k)\,dt \\ &= \frac{1}{2|N(h)|}\sum_{(j,k)\in N(h)}(\boldsymbol{c}_j-\boldsymbol{c}_k)^\top(\boldsymbol{c}_j-\boldsymbol{c}_k).\end{aligned}$$

3.2 Krigagem Tempo-Variante Funcional

Este modelo de interpolação espacial denominado Krigagem Tempo-Variante Funcional pode ser considerado uma melhoria do método apresentado na Seção 3.1 e foi proposto por Henao *et al.* (2010), em que B-splines foram utilizadas para aproximar as curvas $\chi_{\boldsymbol{s}_i}(t), i = 1,\ldots,n$. A idéia é estimar $\chi_{\boldsymbol{s}_0}(t)$ como uma combinação linear das curvas $\chi_{\boldsymbol{s}_i}(t), i = 1,\ldots,n$, mas em vez de considerar constantes $\lambda_1,\ldots,\lambda_n$, desejamos estimar funções $\lambda_1(t),\ldots,\lambda_n(t)$. Mais precisamente,

$$\hat{\chi}_{\boldsymbol{s}_0}(t) = \sum_{i=1}^{n}\lambda_i(t)\chi_{\boldsymbol{s}_i}(t),$$

em que aproximamos as funções $\lambda_i(t), i = 1,\ldots,n$, usando ondaletas. Então, temos que $\lambda_i(t) \approx \boldsymbol{b}_i^\top\boldsymbol{\phi}_J(t), i = 1,\ldots,n$ e $\chi_{\boldsymbol{s}_i}(t) \approx \boldsymbol{a}_i^\top\boldsymbol{\phi}_J(t), i = 0,1,\ldots,n$, com

$$\begin{aligned}\boldsymbol{\phi}_J(t) &= (\phi_{J,-M}(t),\ldots,\phi_{J,M}(t))^\top, \\ \boldsymbol{a}_i &= (a_{i,-M}^J,\ldots,a_{i,M}^J)^\top, i = 0,1,\ldots,n, \\ \boldsymbol{b}_i &= (b_{i,-M}^J,\ldots,b_{i,M}^J)^\top, i = 1,\ldots,n,\end{aligned}$$

e, então, desejamos estimar o seguinte problema de otimização não linear:

$$\boldsymbol{b}_1,\ldots,\boldsymbol{b}_n = \arg\min_{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_n}\mathrm{E}\left[\|\hat{\chi}_{\boldsymbol{s}_0}(t)-\boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\|^2\right] \text{ sujeito a } \mathrm{E}\left[\hat{\chi}_{\boldsymbol{s}_0}(t)-\boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\right] = 0. \quad (3.6)$$

Note que resolver o problema de otimização linear descrito pela Equação (3.6) pode não ser uma tarefa simples. O Teorema 3.2.1 fornece uma simplificação do problema de otimização não linear (3.6).

Teorema 3.2.1. *Suponha que $\boldsymbol{a}_0, \boldsymbol{a}_1\ldots,\boldsymbol{a}_n$ são observações de um campo aleatório multivariado*

fracamente estacionário e isotrópico, então resolver o problema de otimização não linear dado por

$$\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n = \arg\min_{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_n} \mathrm{E}\left[\|\hat{\chi}_{\boldsymbol{s}_0}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\|^2\right] \quad \textit{sujeito a } \mathrm{E}\left[\hat{\chi}_{\boldsymbol{s}_0}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\right] = 0$$

é equivalente a encontrar o ponto de mínimo da forma quadrática descrita por

$$\boldsymbol{\beta}^\top Q \boldsymbol{\beta} - 2\boldsymbol{\beta}^\top \boldsymbol{P}$$

em que

$$\begin{aligned}
\boldsymbol{\beta} &= (\boldsymbol{b}_1^\top, \ldots, \boldsymbol{b}_n^\top, \boldsymbol{m}^\top)^\top \\
Q &= \begin{pmatrix} Q_{11} & Q_{12} & \ldots & Q_{1n} & I_N \\ Q_{21} & Q_{22} & \ldots & Q_{2n} & I_N \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ Q_{n1} & Q_{n2} & \ldots & Q_{nn} & I_N \\ I_N & I_N & \ldots & I_N & \mathbf{0}_N \end{pmatrix} \\
P &= (\boldsymbol{P}_1^\top, \ldots, \boldsymbol{P}_n^\top, \boldsymbol{c}^\top)^\top
\end{aligned}$$

com

$$\begin{aligned}
Q_{i,j} &= \int_0^1 \boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J(t)^\top \mathrm{Cov}(\boldsymbol{a}_i; \boldsymbol{a}_j)\boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J(t)^\top dt \\
\boldsymbol{P}_i &= \int_0^1 \boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J(t)^\top \mathrm{Cov}(\boldsymbol{a}_i; \boldsymbol{a}_0)\boldsymbol{\phi}_J(t) dt \\
\boldsymbol{c}^\top \boldsymbol{\phi}_J(t) &\approx \begin{cases} 1, & \textit{se } t \in [0,1] \\ 0, & \textit{caso contrário} \end{cases};
\end{aligned}$$

onde N é ordem da matriz Q_{11}, $\mathbf{0}_N$ é matriz nula de ordem N e I_N é a matriz identidade de ordem N.

Demonstração. Antes de prosseguir, observamos que a demonstração desse teorema é uma adaptação para ondaletas dos resultados demonstrados por Henao *et al.* (2010) para BSplines.

Primeiramente, observe que a restrição $\mathrm{E}\left[\hat{\chi}_{\boldsymbol{s}_0}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\right] = 0$ é equivalente a $\sum_{i=1}^{n} \lambda_i(t) = 1, \forall t \in [0,1]$, pois $\boldsymbol{\chi}_{\boldsymbol{s}}(t), s \in D$ é pontualmente fracamente estacionário. Além disso, pelo Teorema de Fubini, temos que

$$\mathrm{E}\left[\|\hat{\chi}_{\boldsymbol{s}_0}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\|^2\right] = \int_0^1 \mathrm{Var}\left(\hat{\chi}_{\boldsymbol{s}_0}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\right) dt.$$

Note que usamos as aproximações

$$\begin{aligned}
\lambda_i(t) &\approx \boldsymbol{b}_i^\top \boldsymbol{\phi}_J(t), \\
\chi_{\boldsymbol{s}_i}(t) &\approx \boldsymbol{a}_i^\top \boldsymbol{\phi}_J(t), \\
\chi_{\boldsymbol{s}_0}(t) &\approx \boldsymbol{a}_0^\top \boldsymbol{\phi}_J(t),
\end{aligned}$$

com $\sum_{i=1}^{n} \lambda_i(t) = \sum_{i=1}^{n} \boldsymbol{b}_i \boldsymbol{\phi}_J(t) = 1, \forall t \in [0,1]$, isto é,

$$\sum_{i=1}^{n} \sum_{k=-M}^{M} b_{i,k}^J \phi_{J,k}(t) = 1, \forall t \in [0,1].$$

Note que podemos aproximar a função constante 1 por $\boldsymbol{\alpha}^\top \boldsymbol{\phi}_J(t) \approx 1$ com $\boldsymbol{\alpha} = (\alpha_{-M}, \ldots, \alpha_M)^\top$ e, então, temos que

$$\sum_{i=1}^{n} \sum_{k=-M}^{M} b_{i,k} \phi_{J,k}(t) = \sum_{i=1}^{n} \alpha_k \phi_{J,k}(t).$$

e, consequentemente, temos que

$$\left\langle \sum_{i=1}^{n} \sum_{k=-M}^{M} b_{i,k}^J \phi_{J,k}(t); \phi_{J,l}(t) \right\rangle = \left\langle \sum_{i=1}^{n} \alpha_k \phi_{J,k}(t); \phi_{J,l}(t) \right\rangle, \quad l = -M, \ldots, M,$$

ou seja,

$$\sum_{i=1}^{n} b_{i,l}^J = \alpha_l, \quad l = -M, \ldots, M,$$

ou em forma matricial

$$\begin{pmatrix} \sum_{i=1}^n b_{i,-M}^J \\ \vdots \\ \sum_{i=1}^n b_{i,0}^J \\ \vdots \\ \sum_{i=1}^n b_{i,M}^J \end{pmatrix} = \begin{pmatrix} \alpha_{-M} \\ \vdots \\ \alpha_0 \\ \vdots \\ \alpha_M \end{pmatrix}.$$

Ou seja, a restrição $\sum_{i=1}^n \lambda_i(t) = 1$ pode ser reescrita por $\sum_{i=1}^n \boldsymbol{b}_i = \boldsymbol{\alpha}$.

Observe que $\hat{\chi}_{\boldsymbol{s}_0}(t) = \sum_{i=1}^n \lambda_i(t)\chi_{\boldsymbol{s}_i}(t) = \sum_{i=1}^n \boldsymbol{b}_i^\top \boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J(t)^\top \boldsymbol{a}_i$ e

$$\begin{aligned}
\operatorname{Var}\left(\hat{\chi}_{\boldsymbol{s}_0}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\right) &= \sum_{i=1}^n \sum_{j=1}^n \operatorname{Cov}(\boldsymbol{b}_i^\top \boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J(t)^\top \boldsymbol{a}_i; \boldsymbol{b}_j^\top \boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J(t)^\top \boldsymbol{a}_j) + \boldsymbol{\phi}_J(t)^\top \operatorname{Var}(\boldsymbol{a}_0)\, \boldsymbol{\phi}_J(t) - \\
&\quad - 2\sum_{i=1}^n \operatorname{Cov}\left(\boldsymbol{b}_i^\top \boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J(t)^\top \boldsymbol{a}_i; \boldsymbol{a}_0 \boldsymbol{\phi}_J(t)\right) \\
&= \sum_{i=1}^n \sum_{j=1}^n \boldsymbol{b}_i^\top \boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J(t)^\top \operatorname{Cov}(\boldsymbol{a}_i, \boldsymbol{a}_j)\boldsymbol{\phi}_J(t)^\top \boldsymbol{\phi}_J(t)\boldsymbol{b}_j^\top + \boldsymbol{\phi}_J(t)^\top \operatorname{Var}(\boldsymbol{a}_0)\, \boldsymbol{\phi}_J(t) - \\
&\quad - 2\sum_{i=1}^n \boldsymbol{b}_i^\top \boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J(t)^\top \operatorname{Cov}(\boldsymbol{a}_i, \boldsymbol{a}_0)\, \boldsymbol{\phi}_J(t). \qquad (3.7)
\end{aligned}$$

Considere as seguintes notações

$$\begin{aligned}
Q_{i,j} &= \int_0^1 \boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J(t)^\top \operatorname{Cov}(\boldsymbol{a}_i, \boldsymbol{a}_j)\boldsymbol{\phi}_J(t)^\top \boldsymbol{\phi}_J(t)dt, \quad i, j \in \{i, \ldots, n\}, \\
D &= \boldsymbol{\phi}_J(t)^\top \operatorname{Var}(\boldsymbol{a}_0)\, \boldsymbol{\phi}_J(t), \\
\boldsymbol{P}_i &= \int_0^1 \boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J(t)^\top \operatorname{Cov}(\boldsymbol{a}_i, \boldsymbol{a}_0)\, \boldsymbol{\phi}_J(t)dt, \quad i \in \{1, \ldots, n\},
\end{aligned}$$

então o problema de otimização (3.6) pode ser reescrito em

$$\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n = \arg\min_{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_n} \sum_{i=1}^n \sum_{j=1}^n \boldsymbol{b}_i^\top Q_{ij} \boldsymbol{b}_j + D - 2\sum_{i=1}^n \boldsymbol{b}_i^\top \boldsymbol{P}_i \quad \text{sujeito a } \sum_{i=1}^n \boldsymbol{b}_i = \boldsymbol{\alpha}.$$

Usando multiplicadores de Lagrange, temos que

$$\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n = \arg\min_{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_n} \sum_{j=1}^n \boldsymbol{b}_i^\top Q_{ij} \boldsymbol{b}_j + D - 2\sum_{i=1}^n \boldsymbol{b}_i^\top \boldsymbol{P}_i + 2\boldsymbol{m}^\top \left(\sum_{i=1}^n \boldsymbol{b}_i - \boldsymbol{\alpha}\right), \qquad (3.8)$$

como o termo D é constante em relação a $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$, ele não interfere na busca pelo ponto de mínimo e podemos reescrever a equação 3.8 na seguinte forma matricial

$$(\boldsymbol{b}_1^\top, \ldots, \boldsymbol{b}_n^\top, \boldsymbol{m}^\top)^\top = \arg\min \boldsymbol{\beta}^\top Q \boldsymbol{\beta} - 2\boldsymbol{\beta}\boldsymbol{P},$$

em que,

$$\begin{aligned}
\boldsymbol{\beta} &= (\boldsymbol{b}_1^\top, \ldots, \boldsymbol{b}_n^\top, \boldsymbol{m}^\top)^\top, \\
Q &= \begin{pmatrix} Q_{11} & Q_{12} & \ldots & Q_{1n} & I_N \\ Q_{21} & Q_{22} & \ldots & Q_{2n} & I_N \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ Q_{n1} & Q_{n2} & \ldots & Q_{nn} & I_N \\ I_N & I_N & \ldots & I_N & \mathbf{0}_N \end{pmatrix}, \\
\boldsymbol{P} &= (\boldsymbol{P}_1^\top, \ldots, \boldsymbol{P}_n^\top, \boldsymbol{c}^\top)^\top.
\end{aligned}$$

□

Note que no Teorema 3.2.1 é necessário estimar a matriz de covariância cruzada $\mathrm{Cov}(\boldsymbol{a}_i, \boldsymbol{a}_j)$ de um campo aleatório multivariado fracamente estacionário [1], passo que envolve técnicas elaboradas de estimação, pois precisamos garantir que as estimativas sejam positiva-definidas. Para os leitores interessados, Genton e Kleiber (2015) faz um revisão de métodos para estimar a matriz covariância cruzada de um campo aleatório estacionário. Contudo, é possível contornar esse desafio usando o semivariograma de um campo aleatório de uma maneira análoga ao proposto no Teorema 3.1.1 usando o Teorema 3.2.2.

Teorema 3.2.2. *Suponha que $\boldsymbol{a}_0, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ são observações de um campo aleatório multivariado fracamente estacionário e isotrópico, então resolver o problema de otimização descrito por*

$$\arg\min_{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_n} \mathrm{E}\left[\|\hat{\chi}_{\boldsymbol{s}_0}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\|^2\right] \text{ sujeito a } \sum_{i=1}^{n} \lambda_i(t) = \boldsymbol{\alpha}^\top \boldsymbol{\phi}_J(t),$$

[1] Vide, por exemplo, Goulard e Voltz (1992) para uma revisão sobre campo aleatório fracamente estacionário e isotrópico.

em que $\hat{\chi}_{\boldsymbol{s}_0} = \sum_{i=1}^{n} \boldsymbol{b}_i^\top \boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J(t)^\top \boldsymbol{a}_i$, $\lambda_i(t) = \boldsymbol{b}_i^\top \boldsymbol{\phi}_J(t)$ *e* $\boldsymbol{\alpha}^\top \boldsymbol{\phi}_J(t) \approx 1, \forall t \in [0,1]$ *é equivalente a encontrar o ponto de máximo da seguinte forma quadrática*

$$\boldsymbol{\beta}^\top R\boldsymbol{\beta} - 2\boldsymbol{U}^\top \boldsymbol{\beta}$$

em que

$$\begin{aligned}
\boldsymbol{\beta} &= (\boldsymbol{b}_1^\top, \ldots, \boldsymbol{b}_n^\top, \boldsymbol{m}^\top)^\top, \\
R &= \begin{pmatrix} R_{11} & R_{12} & \ldots & R_{1n} & I_N \\ R_{21} & R_{22} & \ldots & R_{2n} & I_N \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ R_{n1} & R_{n2} & \ldots & R_{nn} & I_N \\ I_N & I_N & \ldots & I_N & \mathbf{0}_N \end{pmatrix}, \\
\boldsymbol{U} &= (\boldsymbol{U}_1^\top, \ldots, \boldsymbol{U}_n^\top, \boldsymbol{\alpha}^\top)^\top,
\end{aligned}$$

com N *a ordem da matriz* R_{11} *e*

$$\begin{aligned}
R_{ij} &= \int_0^1 \boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J(t)^\top \frac{1}{2}\operatorname{Var}(\boldsymbol{a}_i - \boldsymbol{a}_j)\boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J(t)^\top dt, \quad i,j \in \{1,\ldots,n\}, \\
\boldsymbol{U}_i &= \int_0^1 \boldsymbol{\phi}(t)\boldsymbol{\phi}_J(t)^\top \frac{1}{2}\operatorname{Var}(\boldsymbol{a}_i - \boldsymbol{a}_0)\boldsymbol{\phi}_J(t)dt, \quad i \in \{1,\ldots,n\}, \\
\boldsymbol{\alpha}^\top \boldsymbol{\phi}_J(t) &\approx \begin{cases} 1, & \textit{se } t \in [0,1] \\ 0, & \textit{caso contrário} \end{cases}.
\end{aligned} \tag{3.9}$$

Demonstração. Por hipótese, $\boldsymbol{a}_0, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ são observações de um campo aleatório multivariado fracamente estacionário e isotrópico, isto é,

$$\begin{aligned}
\operatorname{Cov}(\boldsymbol{a}_i, \boldsymbol{a}_j) &= C(\|\boldsymbol{s}_i - \boldsymbol{s}_j\|), \quad i,j \in \{0,1,\ldots,n\}, \\
\frac{1}{2}\operatorname{Var}(\boldsymbol{a}_i - \boldsymbol{a}_j) &= G(\|\boldsymbol{s}_i - \boldsymbol{s}_j\|), \quad i,j \in \{0,1,\ldots,n\}, \\
C(\|\boldsymbol{s}_i - \boldsymbol{s}_j\|) &= C(0) - G(\|\boldsymbol{s}_i - \boldsymbol{s}_j\|), \quad i,j \in \{0,1,\ldots,n\},
\end{aligned}$$

em que $G(h)$ é o semivariograma para o campo multivariado.

De acordo com o Teorema 3.2.1, resolver o problema de otimização

$$\arg\min_{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_n} \mathrm{E}\left[\|\hat{\chi}_{\boldsymbol{s}_0}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\|\right] \text{ sujeito a } \sum_{i=1}^{n} \lambda_i(t) = 1,$$

com $\lambda_i(t) = \boldsymbol{b}_i^\top \boldsymbol{\phi}_J(t)$, é equivalente a achar o ponto de mínimo da forma quadrática

$$\boldsymbol{\beta}^\top Q \boldsymbol{\beta} - 2\boldsymbol{\beta}^\top \boldsymbol{P},$$

com

$$Q = \begin{pmatrix} Q_{11} & Q_{12} & \ldots & Q_{1n} & I_N \\ Q_{21} & Q_{22} & \ldots & Q_{2n} & I_N \\ \vdots & \vdots & \ddots & \vdots & I_N \\ Q_{n1} & Q_{n2} & \ldots & Q_{nn} & I_N \\ I_N & I_N & \ldots & I_N & \mathbf{0}_N \end{pmatrix},$$

$$\boldsymbol{P} = (\boldsymbol{P}_1^\top, \ldots, \boldsymbol{P}_n^\top, \boldsymbol{\alpha}^\top)^\top,$$

em que

$$\begin{aligned} Q_{ij} &= \int_0^1 \boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J(t)^\top C(\|\boldsymbol{s}_i - \boldsymbol{s}_j\|)\boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J(t)^\top dt, \quad i,j \in \{0,1,\ldots,n\}, \\ \boldsymbol{P}_i &= \int_0^1 \boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J(t)^\top C(\|\boldsymbol{s}_i - \boldsymbol{s}_0\|)\boldsymbol{\phi}_J(t) dt, \quad i \in \{0,1,\ldots,n\}, \\ \boldsymbol{\beta} &= (\boldsymbol{b}_1^\top, \ldots, \boldsymbol{b}_n^\top, \boldsymbol{m}^\top)^\top. \end{aligned}$$

Como $C(\|\boldsymbol{s}_i - \boldsymbol{s}_j\|) = C(0) - G(\|\boldsymbol{s}_i - \boldsymbol{s}_j\|)$, temos que

$$\begin{aligned} Q_{ij} &= \int_0^1 \boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J(t)^\top C(\|\boldsymbol{s}_i - \boldsymbol{s}_j\|)\boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J(t)^\top dt \\ &= \int_0^1 \boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J(t)^\top \left[C(0) - G(\|\boldsymbol{s}_i - \boldsymbol{s}_j\|)\right] \boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J(t)^\top dt \\ &= Q_{00} - R_{ij}, \end{aligned}$$

e

$$\begin{aligned}\boldsymbol{P}_i &= \int_0^1 \boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J(t)^\top C(\|\boldsymbol{s}_i - \boldsymbol{s}_0\|)\boldsymbol{\phi}_J(t)dt \\ &= \int_0^1 \boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J(t)^\top \left[C(0) - G(\|\boldsymbol{s}_i - \boldsymbol{s}_0\|)\right]\boldsymbol{\phi}_J(t)dt \\ &= \boldsymbol{P}_0 - \boldsymbol{U}_i.\end{aligned}$$

Logo, $Q = Q^0 - R$ e $P = P^0 - U$ com

$$Q^0 = \begin{pmatrix} (\mathbf{1}\mathbf{1}^\top) \otimes Q_{00} & \mathbf{1} \otimes \mathbf{0} \\ \mathbf{1}^\top \otimes \mathbf{0} & \mathbf{0} \end{pmatrix},$$

$$P^0 = \begin{pmatrix} \mathbf{1} \otimes \boldsymbol{P}_0 \\ \mathbf{0} \cdot \mathbf{1} \end{pmatrix}.$$

em que $\mathbf{1} = (1, \ldots, 1)^\top$ é um vetor coluna N linhas e $\mathbf{0}$ é uma matriz nula de ordem $N \times N$. Observe que

$$\begin{aligned}\boldsymbol{\beta}^\top Q^0 \boldsymbol{\beta} &= \left(\left(\sum_{i=1}^n \boldsymbol{b}_i^\top \right) Q_{00}, \ldots, \left(\sum_{i=1}^n \boldsymbol{b}_i^\top \right) Q_{00}, \mathbf{1}^\top \cdot \mathbf{0} \right) \begin{pmatrix} \boldsymbol{b}_1 \\ \vdots \\ \boldsymbol{b}_n \\ \mathbf{0} \cdot \mathbf{1} \end{pmatrix} \\ &= \sum_{j=1}^n \sum_{i=1}^n \boldsymbol{b}_i^\top Q_{00} \boldsymbol{b}_j \\ &= \boldsymbol{\alpha}^\top Q_{00} \boldsymbol{\alpha},\end{aligned}$$

e

$$\boldsymbol{P}^{0\top} \boldsymbol{\beta} = \boldsymbol{P}_0^\top \sum_{i=1}^n \boldsymbol{b}_i = \boldsymbol{P}_0^\top \boldsymbol{\alpha}. \tag{3.10}$$

Finalmente, temos que

$$\begin{aligned}\arg\min_{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_n}\left[\boldsymbol{\beta}^\top Q\boldsymbol{\beta}-2\boldsymbol{\beta}\boldsymbol{P}\right] &= \arg\min_{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_n}\left[\boldsymbol{\beta}^\top Q^0\boldsymbol{\beta}-\boldsymbol{\beta}^\top R\boldsymbol{\beta}-2\boldsymbol{P}_0^\top\boldsymbol{\beta}+2\boldsymbol{\beta}^\top\boldsymbol{U}\right]\\ &= \arg\min_{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_n}\left[\boldsymbol{\alpha}Q_{00}\boldsymbol{\alpha}-\boldsymbol{\beta}^\top R\boldsymbol{\beta}-2\boldsymbol{P}_0^\top\boldsymbol{\alpha}+2\boldsymbol{\beta}^\top\boldsymbol{U}\right]\\ &= \arg\max_{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_n}\left[\boldsymbol{\beta}^\top R\boldsymbol{\beta}-2\boldsymbol{\beta}^\top\boldsymbol{U}\right].\end{aligned}$$

□

O Teorema 3.2.2 substitui a necessidade de estimação da matriz covariância cruzada $\mathrm{Cov}\left(\boldsymbol{a}_i,\boldsymbol{a}_j\right)$ no modelo proposto por Henao *et al.* (2010) pela estimação do semivariograma $\frac{1}{2}\mathrm{Var}\left(\boldsymbol{a}_i-\boldsymbol{a}_j\right)$ que pode ser estimada via estimador não paramétrico apresentado por Goulard e Voltz (1992):

$$G(h)=\frac{1}{2|N(h)|}\sum_{(i,j)\in N(h)}\left[\boldsymbol{a}_i-\boldsymbol{a}_j\right]\left[\boldsymbol{a}_i-\boldsymbol{a}_j\right]^\top \tag{3.11}$$

em que $N(h)=\{(i,j)\in\{1,\ldots,n\}\times\{1,\ldots,n\}\mid\|\boldsymbol{s}_i-\boldsymbol{s}_j\|\in(h-\epsilon,h+\epsilon)\}$ e $\epsilon>0$ é uma banda de tolerância (vide Wackernagel, 2003, para maiores detalhes).

De uma forma geral, a função escala $\phi(t)$ em uma análise de multirresolução não tem forma analítica fechada. Para resolver as integrais nas equações de $R_{i,j}$ e $\boldsymbol{U}_i$, usamos uma aproximação motivada pela integração de Riemann: $\phi\left(\frac{l}{2^q}\right)$ é conhecido para $l,q\in\mathbb{Z}$ (vide Boggess e Narcowich, 2009, para maiores detalhes), então temos que

i. $\int_0^1\boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J^\top(t)G\left(\|\boldsymbol{s}_i-\boldsymbol{s}_j\|\right)\boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J^\top(t)dt\approx\sum_{k\in C}\frac{1}{2^q}\boldsymbol{\phi}_J\left(\frac{k}{2^q}\right)\boldsymbol{\phi}_J^\top\left(\frac{k}{2^q}\right)\cdot$

$$\cdot G\left(\|\boldsymbol{s}_i-\boldsymbol{s}_j\|\right)\boldsymbol{\phi}_J\left(\frac{k}{2^q}\right)\boldsymbol{\phi}_J^\top\left(\frac{k}{2^q}\right)\text{ com }C=\left\{k\in\mathbb{Z}\mid\frac{k}{2^q}\in[0,1]\right\}$$

ii. $\int_0^1\boldsymbol{\phi}_J(t)\boldsymbol{\phi}_J^\top(t)G\left(\|\boldsymbol{s}_i-\boldsymbol{s}_j\|\right)\boldsymbol{\phi}_J(t)dt\approx\sum_{k\in C}\frac{1}{2^q}\boldsymbol{\phi}_J\left(\frac{k}{2^q}\right)\boldsymbol{\phi}_J^\top\left(\frac{k}{2^q}\right)G\left(\|\boldsymbol{s}_i-\boldsymbol{s}_j\|\right)\boldsymbol{\phi}_J\left(\frac{k}{2^q}\right)$
com $C=\left\{k\in\mathbb{Z}\mid\frac{k}{2^q}\in[0,1]\right\}$.

A Figura 3.2 ilustra a ideia das aproximações i. e ii..

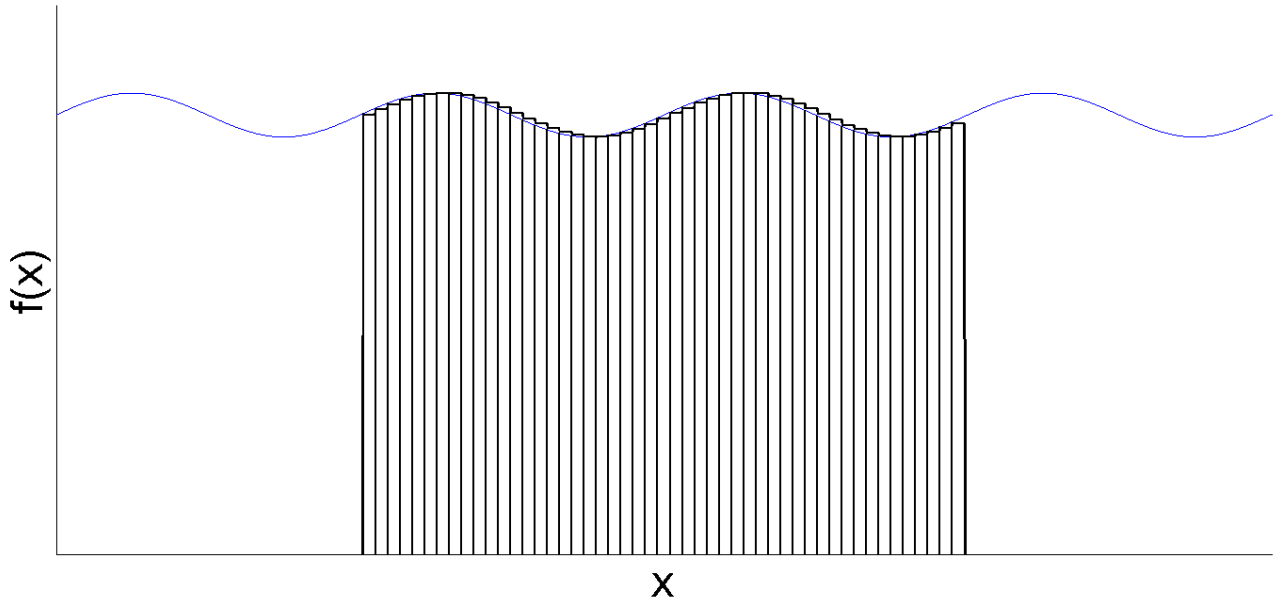


Figura 3.2: *Ilustração do método numérico utilizado nas integrais envolvendo a Krigagem Tempo-Variante Funcional.*

3.3 Krigagem Funcional por Campo

Nessa seção introduzimos uma abordagem de interpolação espacial distinta daquelas apresentadas na Seção 3.1 e na Seção 3.2. Em vez de buscarmos estimar a curva $\chi_{\boldsymbol{s}_0}(t)$ como combinação linear (envolvendo valores escalares ou funções) das curvas $\chi_{\boldsymbol{s}_i}(t)$ em nossa amostra funcional, desejamos estimar os coeficientes $\boldsymbol{a}_0$ da projeção de $\chi_{\boldsymbol{s}_0}(t)$ no espaço de aproximação V_J. Mais precisamente, suponha que aproximamos as curvas $\chi_{\boldsymbol{s}_1}(t), \ldots, \chi_{\boldsymbol{s}_n}(t)$ por

$$\chi_{\boldsymbol{s}_i}(t) \approx \boldsymbol{a}_i^\top \boldsymbol{\phi}_J(t), \quad i = 1, \ldots, n,$$

em que $\boldsymbol{a}_i = (a_{i,-M}^J, \ldots, a_{i,M}^J)^\top$ e $\boldsymbol{\phi}_J(t) = (\phi_{J,-M}(t), \ldots, \phi_{J,M}(t))^\top$. Nosso objetivo é encontrar uma estimativa $\hat{\boldsymbol{a}}_0$ para $\boldsymbol{a}_0$ e, consequentemente, obter uma estimativa para $\chi_{\boldsymbol{s}_0}(t)$ através de $\hat{\chi}_{\boldsymbol{s}_0} = \hat{\boldsymbol{a}}_0^\top \boldsymbol{\phi}_J(t)$.

A ideia para obter uma estimativa de $\boldsymbol{a}_0$ é usar uma combinação linear de $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$, mas em vez de usar a mesma constante escalar θ_i para cada coordenada $a_{i,-M}^J, \ldots, a_{i,M}^J$ da projeção de $\chi_{\boldsymbol{s}_i}(t)$ em V_J, optamos por atribuir uma constante $\theta_{i,j}$ para cada $a_{i,j}^J, i = 1, \ldots, n, j = -M, \ldots, M$. Analogamente aos estimadores descritos na Seção 3.1 e na Seção 3.2, desejamos encontrar $\theta_{i,j}, i = 1, \ldots, n, j = -M, \ldots, M$ com erro quadrático médio mínimo e não viciado. Formalmente, suponha que $\boldsymbol{a}_0, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ são observações de um campo aleatório multivariado

fracamente estacionário e isotrópico e considere as matrizes

$$\Theta_i = \begin{pmatrix} \theta_{i,-M} & 0 & \dots & 0 \\ 0 & \theta_{i,-M+1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \theta_{iM} \end{pmatrix}, \quad i = 1, \dots, n.$$

Então, um estimador para $\boldsymbol{a}_0$ é dado por

$$\hat{\boldsymbol{a}}_0 = \sum_{i=1}^{n} \Theta_i \boldsymbol{a}_i.$$

em que $\Theta_1, \dots, \Theta_n$ é solução do problema de otimização descrito por

$$\begin{aligned} \Theta_1, \dots, \Theta_n = \arg\min_{\Theta_1,\dots,\Theta_n} \mathrm{E}\left[\left(\sum_{i=1}^{n} \Theta_i \boldsymbol{a}_i - \boldsymbol{a}_0 \right)^{\top} \left(\sum_{i=1}^{n} \Theta_i \boldsymbol{a}_i - \boldsymbol{a}_0 \right) \right] \\ \text{sujeito a } \mathrm{E}\left(\sum_{i=1}^{n} \Theta_i \boldsymbol{a}_i - \boldsymbol{a}_0 \right) = \mathbf{0}, \end{aligned} \tag{3.12}$$

em que $\mathbf{0} = (0, \dots, 0)^{\top}$ é um vetor coluna nulo com a mesma dimensão de $\boldsymbol{a}_i$. Logo, uma estimativa para a curva em $\boldsymbol{s}_0$ é dada por

$$\hat{\chi}_{\boldsymbol{s}_0}(t) = \hat{\boldsymbol{a}_0}^{\top} \boldsymbol{\phi}_J(t). \tag{3.13}$$

Denominamos o interpolador espacial de curva descrito pela equação 3.13 de método de Krigagem Funcional por Campo. A escolha desse nome se deve ao fato que supomos que os coeficientes da projeção da curvas no espaço de aproximação em nossa amostra funcional constituem um campo aleatório isotrópico e fracamente estacionário.

Teorema 3.3.1. *Seja $\boldsymbol{a}_0, \boldsymbol{a}_1, \dots, \boldsymbol{a}_n$ observações de um campo aleatório multivariado estacionário e isotrópico. Então, resolver o problema de otimização descrito por*

$$\arg\min_{\Theta_1,\dots,\Theta_n} \mathrm{E}\left[\left(\sum_{i=1}^{n} \Theta_i \boldsymbol{a}_i - \boldsymbol{a}_0 \right)^{\top} \left(\sum_{i=1}^{n} \Theta_i \boldsymbol{a}_i - \boldsymbol{a}_0 \right) \right] \text{ sujeito a } \mathrm{E}\left(\sum_{i=1}^{n} \Theta_i \boldsymbol{a}_i - \boldsymbol{a}_0 \right) = \mathbf{0},$$

é equivalente a encontrar o ponto de máximo da forma quadrática

$$\boldsymbol{\beta}^\top A\boldsymbol{\beta} - 2\boldsymbol{\beta}^\top \boldsymbol{B}, \tag{3.14}$$

em que

$$\begin{aligned}
\boldsymbol{\beta} &= (\boldsymbol{\theta}_1^\top, \ldots, \boldsymbol{\theta}_n^\top, \boldsymbol{\delta}^\top)^\top, \\
\boldsymbol{B} &= \left((G(\boldsymbol{a}_1, \boldsymbol{a}_0) \circ I)\cdot \mathbf{1}, \ldots, (G(\boldsymbol{a}_n, \boldsymbol{a}_0) \circ I)\cdot \mathbf{1}, -\mathbf{1}^\top\right)^\top, \\
A &= \begin{pmatrix}
G(\boldsymbol{a}_1, \boldsymbol{a}_1) \circ I & G(\boldsymbol{a}_1, \boldsymbol{a}_2) \circ I & \ldots & G(\boldsymbol{a}_1, \boldsymbol{a}_n) \circ I & I \\
G(\boldsymbol{a}_2, \boldsymbol{a}_1) \circ I & G(\boldsymbol{a}_2, \boldsymbol{a}_2) \circ I & \ldots & G(\boldsymbol{a}_2, \boldsymbol{a}_n) \circ I & I \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
G(\boldsymbol{a}_n, \boldsymbol{a}_1) \circ I & G(\boldsymbol{a}_n, \boldsymbol{a}_2) \circ I & \ldots & G(\boldsymbol{a}_n, \boldsymbol{a}_n) \circ I & I \\
I & I & \ddots & I & \mathbf{0}
\end{pmatrix},
\end{aligned}$$

em que $\boldsymbol{\theta}_i = (\theta_{i,-M}, \ldots, \theta_{i,M})^\top$*,* $\boldsymbol{\delta}$ *é um vector coluna de mesma dimensão de* $\boldsymbol{\theta}_i$*,* $\circ$ *é o produto de Hadamard (vide Styan, 1973, para maiores detalhes),* $G(\boldsymbol{a}_i, \boldsymbol{a}_j) = \frac{1}{2}\operatorname{Var}(\boldsymbol{a}_i - \boldsymbol{a}_j)$*,* $\mathbf{0}$ *é matriz nula com mesma dimensão de* $G(\boldsymbol{a}_i, \boldsymbol{a}_i)$ *e* I *é a matriz identidade com a mesma ordem de* $G(\boldsymbol{a}_i, \boldsymbol{a}_j)$*.*

Demonstração. Primeiro note que, sob a restrição $\mathrm{E}\left(\sum_{i=1}^n \Theta_i \boldsymbol{a}_i - \boldsymbol{a}_0\right) = \mathbf{0}$, temos que

$$\mathrm{E}\left(\sum_{i=1}^n \Theta_i \boldsymbol{a}_i - \boldsymbol{a}_0\right)^\top \left(\sum_{i=1}^n \Theta_i \boldsymbol{a}_i - \boldsymbol{a}_0\right) = \text{traço}\left[\operatorname{Var}\left(\sum_{i=1}^n \Theta_i \boldsymbol{a}_i - \boldsymbol{a}_0\right)\right].$$

Observe que a restrição $\mathrm{E}\left(\sum_{i=1}^n \Theta_i \boldsymbol{a}_i - \boldsymbol{a}_0\right) = \mathbf{0}$ é equivalente a $\sum_{i=1}^n \Theta_i = I$ e

$$\begin{aligned}
\operatorname{Var}\left(\sum_{i=1}^{n}\Theta_i\boldsymbol{a}_i-\boldsymbol{a}_0\right) &= \operatorname{Var}(\boldsymbol{a}_0)+\sum_{i=1}^{n}\Theta_i\operatorname{Var}(\boldsymbol{a}_i)\Theta_i-2\sum_{i=1}^{n}\operatorname{Cov}(\Theta_i\boldsymbol{a}_i;\boldsymbol{a}_0)+\\
&\quad+2\sum_{j=1}^{n}\sum_{k=j+1}^{n}\Theta_j\operatorname{Cov}(\boldsymbol{a}_j;\boldsymbol{a}_k)\Theta_k\\
&= C(0)-2\sum_{i=1}^{n}\Theta_i\left[C(0)-G(\boldsymbol{a}_i,\boldsymbol{a}_0)\right]+\sum_{i=1}^{n}\Theta_iC(0)\Theta_i+\\
&\quad+2\sum_{j=1}^{n}\sum_{k=j+1}^{n}\Theta_j\left[C(0)-G(\boldsymbol{a}_j,\boldsymbol{a}_k)\right]\Theta_k\\
&= C(0)+\sum_{j=1}^{n}\sum_{k=1}^{n}\Theta_jC(0)\Theta_k-2C(0)+2\sum_{i=1}^{n}\Theta_iG(\boldsymbol{a}_i,\boldsymbol{a}_0)\\
&\quad-\sum_{j=1}^{n}\sum_{k=1}^{n}\Theta_jG(\boldsymbol{a}_j,\boldsymbol{a}_k)\Theta_k\\
&= 2\sum_{i=1}^{n}\Theta_iG(\boldsymbol{a}_i,\boldsymbol{a}_0)-\sum_{j=1}^{n}\sum_{k=1}^{n}\Theta_jG(\boldsymbol{a}_j,\boldsymbol{a}_k)\Theta_k.
\end{aligned}$$

Logo, o problema de otimização 3.12 é equivalente a

$$\arg\min_{\Theta_1,\ldots,\Theta_n} 2\sum_{i=1}^{n}\text{traço}\left(\Theta_iG(\boldsymbol{a}_i,\boldsymbol{a}_0)\right)-\sum_{j=1}^{n}\sum_{k=1}^{n}\text{traço}\left(\Theta_jG(\boldsymbol{a}_j,\boldsymbol{a}_k)\Theta_k\right)\ \text{sujeito a}\ \sum_{i=1}^{n}\Theta_i=I,$$

em que traço$(\cdot)$ é o traço de uma matriz. Seja $\boldsymbol{\theta}_i=(\theta_{i,-M},\ldots,\theta_{i,M})$, então o problema de otimização (3.12) pode ser resumido em

$$\begin{aligned}
&\arg\min_{\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_n} 2\sum_{i=1}^{n}\boldsymbol{\theta}_i^{\top}\left(G(\boldsymbol{a}_i,\boldsymbol{a}_0)\circ I\right)\mathbf{1}-\sum_{j=1}^{n}\sum_{k=1}^{n}\boldsymbol{\theta}_j^{\top}\left(G(\boldsymbol{a}_j,\boldsymbol{a}_k)\circ I\right)\boldsymbol{\theta}_k\\
&\qquad\text{sujeito a}\ \sum_{i=1}^{n}\boldsymbol{\theta}_i=\mathbf{1},
\end{aligned}$$

em que $\circ$ é operador de Hadamard (vide Styan, 1973, para maiores detalhes) e $\mathbf{1}=(1,\ldots,1)^{\top}$ é vetor coluna com a mesma dimensão de $\boldsymbol{a}_i$. Usando multiplicadores de Lagrange, o problema de otimização descrito por (3.12) é equivalente a encontrar o ponto de mínimo de

$$2\sum_{i=1}^{n}\boldsymbol{\theta}_i^{\top}\left[G(\boldsymbol{a}_i,\boldsymbol{a}_0)\circ I\right]\mathbf{1}-\sum_{j=1}^{n}\sum_{k=1}^{n}\boldsymbol{\theta}_j^{\top}\left(G(\boldsymbol{a}_j,\boldsymbol{a}_k)\circ I\right)\boldsymbol{\theta}_k+2\boldsymbol{\delta}^{\top}\left(\sum_{j=1}^{n}\boldsymbol{\theta}_j-\mathbf{1}\right),$$

em que $\boldsymbol{\delta}$ é um vetor coluna de mesma dimensão de $\boldsymbol{\theta}_i$. Em forma matricial temos que

$$\begin{aligned}(\boldsymbol{\theta}_1^\top, \ldots, \boldsymbol{\theta}_n^\top, \boldsymbol{\delta}^\top)^\top &= \arg\min_{\boldsymbol{\beta}} 2\boldsymbol{\beta}^\top \boldsymbol{B} - \boldsymbol{\beta}^\top A \boldsymbol{\beta} \\ &= \arg\max_{\boldsymbol{\beta}} \boldsymbol{\beta}^\top A \boldsymbol{\beta} - 2\boldsymbol{\beta}^\top \boldsymbol{B},\end{aligned}$$

onde

$$\begin{aligned}\boldsymbol{\beta} &= (\boldsymbol{\theta}_1^\top, \ldots, \boldsymbol{\theta}_n^\top, \boldsymbol{\delta}^\top)^\top, \\ \boldsymbol{B} &= \left((G(\boldsymbol{a}_1, \boldsymbol{a}_0) \circ I) \cdot \mathbf{1}, \ldots, (G(\boldsymbol{a}_n, \boldsymbol{a}_0) \circ I) \cdot \mathbf{1}, -\mathbf{1}^\top\right)^\top, \\ A &= \begin{pmatrix} G(\boldsymbol{a}_1, \boldsymbol{a}_1) \circ I & G(\boldsymbol{a}_1, \boldsymbol{a}_2) \circ I & \ldots & G(\boldsymbol{a}_1, \boldsymbol{a}_n) \circ I & I \\ G(\boldsymbol{a}_2, \boldsymbol{a}_1) \circ I & G(\boldsymbol{a}_2, \boldsymbol{a}_2) \circ I & \ldots & G(\boldsymbol{a}_2, \boldsymbol{a}_n) \circ I & I \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ G(\boldsymbol{a}_n, \boldsymbol{a}_1) \circ I & G(\boldsymbol{a}_n, \boldsymbol{a}_2) \circ I & \ldots & G(\boldsymbol{a}_n, \boldsymbol{a}_n) \circ I & I \\ I & I & \ldots & I & \mathbf{0} \end{pmatrix}.\end{aligned}$$

□

Note que a forma quadrática na Equação (3.14) no Teorema 3.3.1 não envolve integrações, como nos métodos apresentados na Seção 3.1 e na Seção 3.2, evitando possíveis erros de aproximação numérica. Observe também que transformamos o problema de otimização não linear da Equação (3.12) em um problema de encontrar ponto de máximo de uma forma quadrática: uma tarefa analítica e computacionalmente mais viável.

No Capítulo 4, realizamos um estudo de simulação dos três métodos apresentados nesse capítulo, Krigagem Ordinária Funcional, Krigagem Tempo-Variante Funcional e Krigagem Funcional por Campo , com o objetivo de compará-los usando métricas inspiradas no erro quadrático médio. Note que não é preciso estudar o vício, pois em todos métodos de krigagem para curvas apresentados nesse capítulo restringimos nossa busca em estimadores não viciados.

Capítulo 4

Estudo de Simulação

Em todos modelos de krigagem para dados funcionais espaciais expostos no Capítulo 3, supomos que o processo espacial é pontualmente fracamente estacionário e isotrópico. Mais precisamente, para $\boldsymbol{s}, \boldsymbol{s}_1, \boldsymbol{s}_2$ pertencente a um subconjunto D limitado e de volume positivo contido na área de estudo, temos que

i. $\mathrm{E}\left[\boldsymbol{\chi}_{\boldsymbol{s}}(t)\right] = m(t), \forall t \in [0,1]$;

ii. $\mathrm{Cov}\left(\boldsymbol{\chi}_{\boldsymbol{s}_1}(t), \boldsymbol{\chi}_{\boldsymbol{s}_2}(t)\right) = C(\|\boldsymbol{s}_1 - \boldsymbol{s}_2\|; t), \forall t \in [0,1]$ em que $\|\cdot\|$ é a geodésica conforme descrita por Banerjee *et al.* (2014);

iii. $\dfrac{1}{2}\mathrm{Var}\left(\boldsymbol{\chi}_{\boldsymbol{s}_1}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_2}(t)\right) = \gamma(\|\boldsymbol{s}_1 - \boldsymbol{s}_2\|; t),\ \forall t \in [0,1]$. Para $t \in [0,1]$ fixo, $\gamma(\|\boldsymbol{s}_1 - \boldsymbol{s}_2\|; t)$ é chamada de variograma.

Adicionalmente, em nosso estudo de simulação, supomos que para $t \in [0,1]$ fixo, temos que

$$\begin{pmatrix} \boldsymbol{\chi}_{\boldsymbol{s}_1}(t) \\ \vdots \\ \boldsymbol{\chi}_{\boldsymbol{s}_n}(t) \end{pmatrix} \sim N\left(\begin{pmatrix} m(t) \\ \vdots \\ m(t) \end{pmatrix}; \Sigma(t) \right), \tag{4.1}$$

em que $\Sigma(t)$ é uma matriz cuja i-ésima linha e j-ésima coluna é dada por

$\sigma_{i,j}(t) = C(\|\boldsymbol{s}_i - \boldsymbol{s}_j\|; t)$ e $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n \in D$.

Usamos os seguintes modelos para a média e a covariância em nosso estudo de simulação:

a) A função de covariância (modelo exponencial (Banerjee *et al.*, 2014)):

$$C(h,t) = \sigma^2 \exp(-\phi(t) \cdot h), \quad t \in [0,1], h \geq 0,$$

com $\sigma^2 = 0,005$ e $\phi(t) = 10\sqrt{\frac{t+1}{2}}$. Na Figura 4.1, mostramos o gráfico da função de covariância $C(h,t)$.

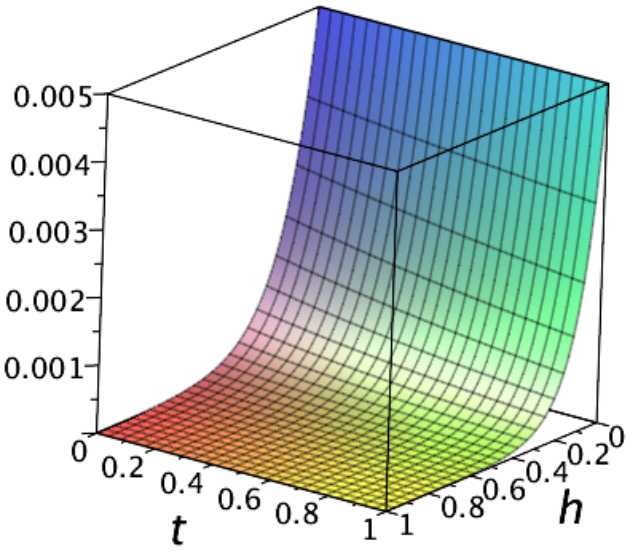


Figura 4.1: *Gráfico da função de covariância.*

b) Foram consideradas duas funções médias. Uma função média contínua dada por

$$m_1(t) = \sin(2\pi t),$$

e outra função média constante por partes com descontinuidades em $\{\frac{1}{3}, \frac{2}{3}\}$ dada por

$$m_2(t) = \begin{cases} 0.5, & t \in \left[0, \frac{1}{3}\right) \bigcup \left[\frac{2}{3}, 1\right] \\ 1, & t \in \left[\frac{1}{3}, \frac{2}{3}\right) \end{cases}.$$

Na construção das amostras funcionais, primeiramente, geramos 50 pontos na região ilustrada na Figura 4.2. Estes pontos são os mesmos para todas as amostras funcionais simuladas.

Após estabelecidos os pontos $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{50}$, consideramos $t_k = \dfrac{k-1}{2^9}, k = 1, \ldots, 2^9$ e para cada t_k geramos 50 valores da distribuição normal multivariada com média $(m(t_k), \ldots, m(t_k))^\top$ e matriz de covariância $\Sigma(t_k)$.

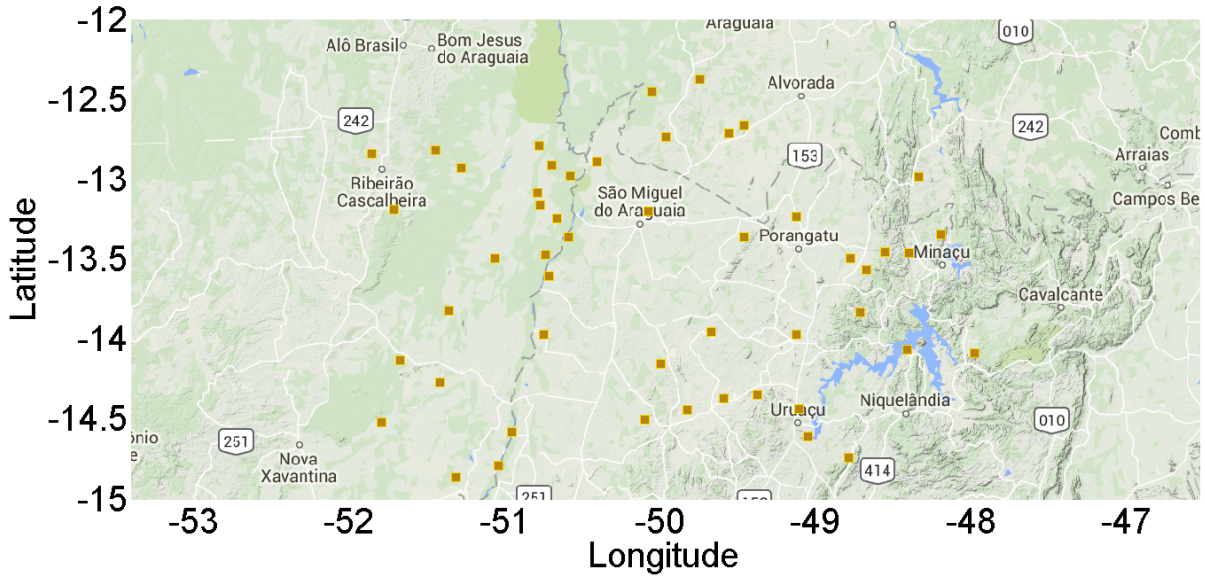


Figura 4.2: *Região onde os dados foram simulados. Cada ponto amarelo é uma localidade selecionada em nosso estudo de simulação.*

4.1 Medidas de Qualidade de Ajuste

Considere um contexto em que temos $M = 100$ amostras funcionais com 50 curvas cada. Então, obtemos M estimativas $\chi^1_{\boldsymbol{s}_0}(t), \ldots, \chi^M_{\boldsymbol{s}_0}(t)$ para a curva $\chi_{\boldsymbol{s}_0}(t)$ no ponto não monitorado $\boldsymbol{s}_0$. Em ordem para avaliar o desempenho dos três métodos de Krigagem introduzidos no Capítulo 3, Krigagem Ordinária Funcional, Krigagem Tempo-Variante Funcional e Krigagem Funcional por Campo, usamos duas medidas de comparação: erro quadrático médio e erro quadrático por amostra, descritas a seguir.

4.1.1 Erro Quadrático Médio

Podemos estimar o erro quadrático médio dos interpoladores espaciais estudados nesse trabalho usando uma aproximação para integral inspirada na integração de Riemann. Mais precisamente,

$$\begin{aligned}
\mathrm{E}\|\hat{\chi}_{\boldsymbol{s}_0}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\|^2 &= \mathrm{E}\int_0^1 |\hat{\chi}_{\boldsymbol{s}_0}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_0}(t)|^2 dt \\
&= \int_0^1 \mathrm{E}\left[\hat{\chi}_{\boldsymbol{s}_0}(t) - \boldsymbol{\chi}_{\boldsymbol{s}_0}(t)\right]^2 dt \\
&\approx \int_0^1 \frac{1}{M}\sum_{k=1}^{M} |\hat{\chi}^k_{\boldsymbol{s}_0}(t) - \chi_{\boldsymbol{s}_0}(t)|^2 \\
&\approx \sum_{l=2}^{2^9} (t_l - t_{l-1}) \frac{1}{M}\sum_{k=1}^{M} |\hat{\chi}^k_{\boldsymbol{s}_0}(t_l) - \chi_{\boldsymbol{s}_0}(t_l)|^2 \\
&= \frac{1}{2^9 M}\sum_{l=2}^{2^9}\sum_{k=1}^{M} |\hat{\chi}^k_{\boldsymbol{s}_0}(t_l) - \chi_{\boldsymbol{s}_0}(t_l)|^2,
\end{aligned}$$

em que $t_l \in [0,1], l = 1, \ldots, 2^9$ são pontos igualmente espaçados com $t_l - t_{l-1} = \dfrac{1}{2^9}$. Usamos a notação EQM para essa medida.

4.1.2 Erro Quadrático Médio por Amostra

Para avaliar a performance dos métodos de krigagem em cada amostra, calculamos a norma em $L^2(\mathbb{R})$ da diferença entre a curva estimada $\chi^k_{\boldsymbol{s}_0}(t)$ e a verdadeira curva observada $\chi_{\boldsymbol{s}_0}(t)$ e usamos a notação EQM_k para essa medida. Mais precisamente,

$$EQM_k = \int_0^1 |\chi^k_{\boldsymbol{s}_0}(t) - \chi_{\boldsymbol{s}_0}(t)|^2 dt. \tag{4.2}$$

A integral na equação (4.2) é aproximada por retângulos de uma maneira análoga a Seção 4.1.1, isto é

$$\begin{aligned}
EQM_k &= \int_0^1 |\chi^k_{\boldsymbol{s}_0}(t) - \chi_{\boldsymbol{s}_0}(t)|^2 dt \\
&\approx \sum_{l=2}^{2^9} (t_l - t_{l-1}) |\chi^k_{\boldsymbol{s}_0}(t_l) - \chi_{\boldsymbol{s}_0}(t_l)|^2 \\
&= \frac{1}{2^9}\sum_{l=2}^{2^9} |\chi^k_{\boldsymbol{s}_0}(t_l) - \chi_{\boldsymbol{s}_0}(t_l)|^2,
\end{aligned} \tag{4.3}$$

em que $t_l \in [0,1], l = 1, \ldots, 2^9$ são pontos igualmente espaçados com $t_l - t_{l-1} = \frac{1}{2^9}$.

4.2 Resultado – Simulação

Nessa seção, analisamos empiricamente os métodos de Krigagem para o contexto de Dados Funcionais propostos no Capítulo 3: Krigagem Ordinária Funcional, Krigagem Tempo-Variante Funcional e Krigagem Funcional por Campo. Dividimos nosso estudo de simulação em dois contextos: quando o ponto não monitorado s_0 está no centro da região sob estudo e quando o ponto não monitorado s_1 está afastado e / ou no canto da região. Na Figura 4.3, exibimos os 50 pontos selecionados espaciais para simular as amostras funcionais e os pontos s_0 e s_1 onde desejamos aplicar os métodos de krigagem para estimar uma curva. Desejamos com isso avaliar o desempenho dos métodos propostos quando existem várias curvas próximas e em volta do ponto não monitorado e quando há poucas curvas ao redor. Ademais, usamos três bases nesse estudo de simulação: ondaletas Haar, ondaletas Daubechies com $N = 3$ ou brevemente $db3$ e B-Splines que foi a base usada nos trabalhos originais de Henao *et al.* (2010) e Henao *et al.* (2011). Para a média constante usamos Haar cujas ondaletas pai e mãe são constantes por parte e para o média contínua usamos a ondaleta $db3$ cujas ondaletas pai e mãe são também contínuas. Os resultados obtidos são comparados com BSplines que foi empregado nos trabalhos originais de Henao (2009); Henao *et al.* (2010, 2011). Organizamos essa seção em duas subseções: na primeira seção analisamos o comportamento dos modelos para média contínua $m_1(t)$ para o ponto centralizado s_0 e para o ponto no canto s_1 e na segunda seção verificamos a performance dos métodos de krigagem para a média constante por partes $m_2(t)$ para os pontos s_0 e s_1.

Para facilitar a elaboração de gráficos e tabelas, usamos as seguintes notações:

i. Krigagem Ordinária Funcional abreviamos por KOF,

ii. Krigagem Tempo-Variante Funcional abreviamos por KTVF,

iii. Krigagem Funcional por Campo abreviamos por KFC.

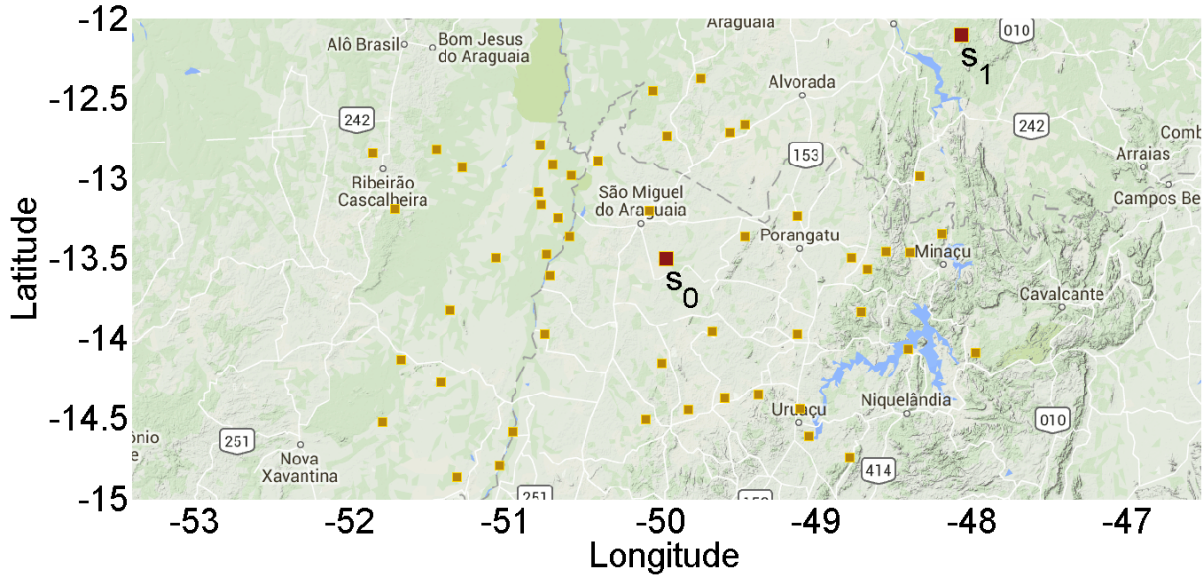


Figura 4.3: *Os pontos amarelos indicam as localidades das curvas nas amostras funcionais simuladas. Os pontos vermelhos s_0 e s_1 são os pontos não monitorados onde desejamos interpolar usando krigagem.*

4.2.1 Simulação 1: Média Contínua

Erro Quadrático Médio para o ponto centralizado s_0 e para o ponto no canto s_1 é mostrado nas tabelas 4.1 e 4.2, respectivamente. Primeiramente, ressaltamos que o uso de BSplines teve o maior EQM entre todos os métodos de Krigagem analisados. Além disso, o método Krigagem Funcional por Campo obteve o menor valor de EQM, chegando a ser entre 10% e 20% menor do que os métodos estabelecidos na literatura que usam BSplines para Krigagem Ordinária Funcional e Krigagem Tempo-Variante Funcional. Ressaltamos ainda que a performance do método Krigagem Tempo-Variante Funcional apresentou EQM menor a Krigagem Ordinária Funcional (modelo mais básico), mas um pouco maior que a Krigagem Funcional por Campo (nova proposta de interpolação espacial).

Por outro, analisando o Erro Quadrático Médio por Amostra, notamos que a Krigagem Funcional por Campo é o estimador com menor variabilidade e a menor média segundo as tabelas 4.3 e 4.4 e as implementações com BSPlines apresentaram maior variabilidade e média.

Nas figuras 4.4 e 4.5, apresentamos os diagramas em caixa para o Erro Quadrático Médio por Amostras. Nestes diagramas retiramos os *outliers* para facilitar a visualização. Esses gráficos reforçam as afirmações sobre a Krigagem Funcional por Campo e ilustram o desempenho intermediário da Krigagem Tempo-Variante Funcional. Além disso, nas tabelas 4.11 e 4.12 observamos que o

vício [1] é pequeno em concordância com os teoremas 3.1.1, 3.2.1, 3.2.2 e 3.3.1 que estabelecem os métodos de Krigagem deste trabalho são não viciados.

Tabela 4.1: *Erro Quadrático Médio para* s_0.

Método	Base	EQM	Base	EQM
KOF	db3	0.0049	Bsplines	0.0061
KTVF		0.0051		0.0058
KFC		**0.0049**	–	–

Abreviações: KOF – Krigagem Ordinária Funcional; KTVF – Krigagem Tempo-Variante Funcional; KFC – Krigagem Funcional por Campo; $db3$ *– ondaleta Daubechies com* $N = 3$.

Tabela 4.2: *Erro Quadrático Médio para* s_1.

Método	Base	EQM	Base	EQM
KOF	db3	0.0054	Bsplines	0.0063
KTVF		0.0050		0.0060
KFC		**0.0048**	–	–

Abreviações: KOF – Krigagem Ordinária Funcional; KTVF – Krigagem Tempo-Variante Funcional; KFC – Krigagem Funcional por Campo; $db3$ *– ondaleta Daubechies com* $N = 3$.

Tabela 4.3: *Sumário para o Erro Quadrático Médio por Amostra para o ponto centralizado* s_0.

Método – Base	Mínimo	1-Qua	Média	Mediana	3-Qua	Máximo	DP
KOF – db3	0.004817	0.004846	0.004872	0.004859	0.004882	0.005203	0.000050
KOF – Bsplines	0.005320	0.005777	0.006094	0.005823	0.005870	0.019033	0.001507
KTVF – db3	0.005048	0.005105	0.005133	0.005120	0.005141	0.005682	0.000073
KTVF – Bsplines	0.005717	0.005780	0.005804	0.005804	0.005830	0.005894	0.000039
KFC – db3	0.004796	0.004850	**0.004869**	**0.004870**	0.004885	0.004954	0.000029

Abreviações: KOF – Krigagem Ordinária Funcional; KTVF – Krigagem Tempo-Variante Funcional; KFC – Krigagem Funcional por Campo; $db3$ *– ondaleta Daubechies com* $N = 3$.

[1] O vício por amostra é calculado pela equação $\frac{1}{2^J}\sum_{k=1}^{2^J}\left(\chi_{\boldsymbol{s}_0}\left(\frac{k}{2^J}\right) - \hat{\chi}_{\boldsymbol{s}_0}\left(\frac{k}{2^J}\right)\right)$.

Tabela 4.4: *Sumário para o Erro Quadrático Médio por Amostra para o ponto no canto s_1.*

Método – Base	Mínimo	1-Qua	Média	Mediana	3-Qua	Máximo	DP
KOF – db3	0.00466	0.00476	0.00539	0.00481	0.00493	0.02128	0.00255
KOF – Bsplines	0.00539	0.00600	0.00629	0.00605	0.00610	0.01851	0.00141
KTVF – db3	0.00481	0.00486	0.00501	0.00488	0.00491	0.01436	0.00097
KTVF – Bsplines	0.00590	0.00599	0.00604	0.00603	0.00608	0.00627	0.00007
KFC – db3	0.00466	0.00475	**0.00479**	**0.00478**	0.00483	0.00505	0.00007

Abreviações: KOF – Krigagem Ordinária Funcional; KTVF – Krigagem Tempo-Variante Funcional; KFC – Krigagem Funcional por Campo; db3 – ondaleta Daubechies com $N = 3$.

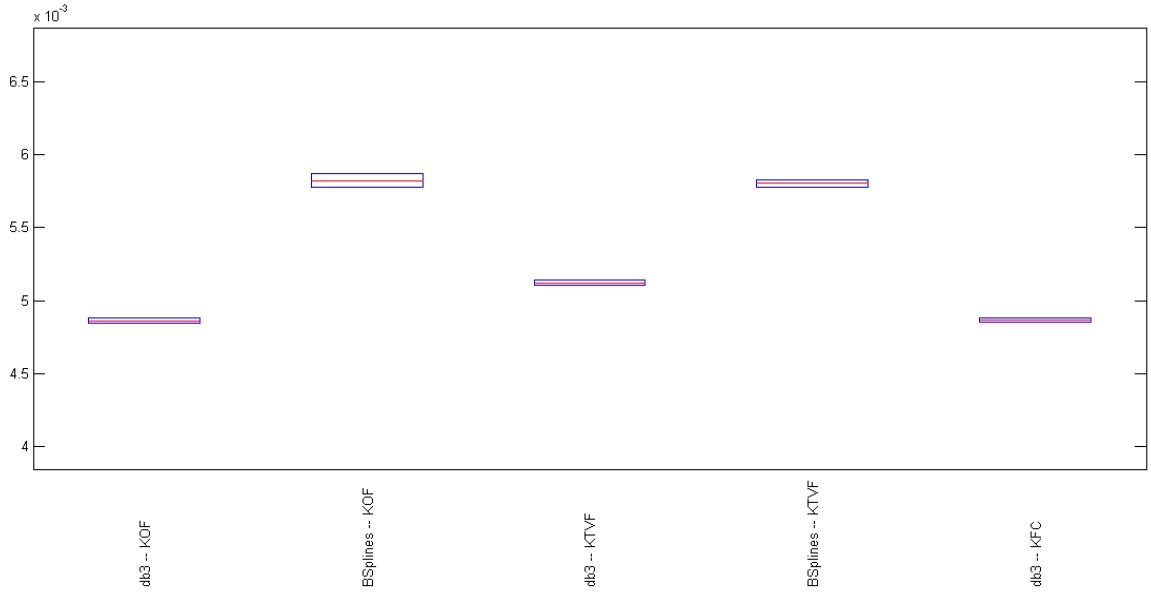


Figura 4.4: *Boxplot do Erro Quadrático Médio por Amostra para o ponto no centro da região s_0 para cada método de krigagem e para cada base no contexto de média contínua.*

Tabela 4.5: *Sumário para o Vício por Amostra para o ponto centralizado s_0*

Método – Base	Mínimo	1-Qua	Média	Mediana	3-Qua	Máximo	DP
KOF – db3	-0.00332	0.00057	0.00126	0.00130	0.00212	0.00535	0.00132
KOF – Bsplines	-0.01253	-0.00007	0.00198	0.00091	0.00203	0.04498	0.00692
KTVF – db3	-0.00325	0.00064	0.00137	0.00141	0.00222	0.00548	0.00132
KTVF – Bsplines	-0.00218	0.00035	0.00072	0.00073	0.00128	0.00256	0.00080
KFC – db3	-0.00209	0.00017	0.00090	0.00089	0.00175	0.00346	0.00116

Abreviações: KOF – Krigagem Ordinária Funcional; KTVF – Krigagem Tempo-Variante Funcional; KFC – Krigagem Funcional por Campo; db3 – ondaleta Daubechies com $N = 3$.

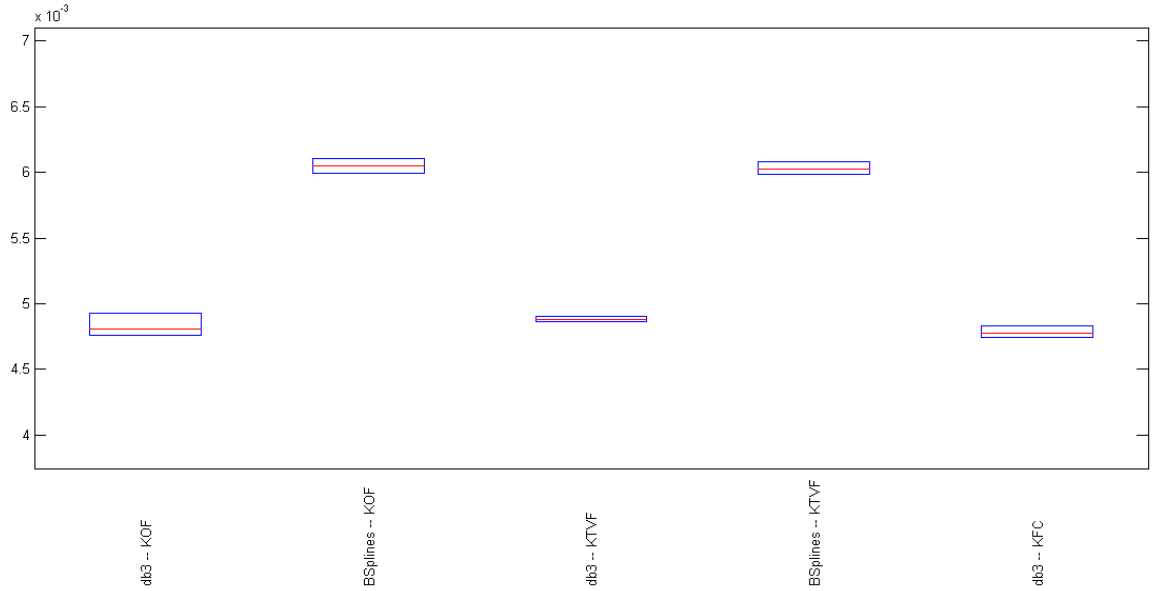


Figura 4.5: *Boxplot do Erro Quadrático Médio por Amostra para o ponto no canto da região s_1 para cada método de krigagem e para cada base no contexto de média contínua.*

Tabela 4.6: *Sumário para o Vício por Amostra para o ponto no canto $\boldsymbol{s}_1$*

Método – Base	Mínimo	1-Qua	Média	Mediana	3-Qua	Máximo	DP
KOF – db3	-0.0277	-0.0046	-0.0033	-0.0024	-0.0009	0.0030	0.0047
KOF – Bsplines	-0.0159	-0.0035	-0.0014	-0.0025	-0.0014	0.0416	0.0069
KTVF – db3	-0.0255	-0.0031	-0.0024	-0.0024	-0.0013	0.0031	0.0027
KTVF – Bsplines	-0.0057	-0.0030	-0.0025	-0.0026	-0.0018	-0.0002	0.0011
KFC – db3	-0.0086	-0.0040	-0.0025	-0.0024	-0.0010	0.0030	0.0023

Abreviações: KOF – Krigagem Ordinária Funcional; KTVF – Krigagem Tempo-Variante Funcional; KFC – Krigagem Funcional por Campo; db3 – ondaleta Daubechies com $N = 3$.

4.2.2 Simulação 2: Média Não Contínua

Análogo ao caso em que tínhamos média contínua, o método Krigagem Funcional por Campo teve o menor EQM tanto para o ponto s_0 e s_1 conforme tabelas 4.9 e 4.10. Além disso, quando analisamos também o Erro Quadrático Médio por amostra nos diagramas de caixa [2] nas figuras 4.4 e 4.7 e nas estatísticas descritivas nas tabelas 4.9 e 4.10, notamos novamente que interpolação espacial usando Krigagem Funcional por Campo tem desempenho satisfatório e os métodos usando BSplines apresentam a maior mediana e média dessa medida de erro quadrático. Ressaltamos ainda que os melhores métodos usando ondaletas Haar (Krigagem Tempo-Variante Funcional e Krigagem Funcional por Campo) têm Erro Quadrático Médio 16% menor que o método usando BSplines (Krigagem Tempo-Variante Funcional). Além disso, nas tabelas 4.11 e 4.12 observamos que o vício é pequeno em concordância com os teoremas 3.1.1, 3.2.1, 3.2.2 e 3.3.1 que estabelecem os métodos de Krigagem deste trabalho são não viciados.

Tabela 4.7: *Erro Quadrático Médio para* $\boldsymbol{s}_0$.

Método	Base	EQM	Base	EQM
KOF		0.01252		0.01463
KTVF	Haar	0.01212	Bsplines	0.01435
KFC		**0.01207**		–

Abreviações: KOF – Krigagem Ordinária Funcional; KTVF – Krigagem Tempo-Variante Funcional; KFC – Krigagem Funcional por Campo.

Tabela 4.8: *Erro Quadrático Médio para* $\boldsymbol{s}_1$.

Método	Base	EQM	Base	EQM
KOF		0.02640		0.01528
KTVF	Haar	0.01578	Bsplines	0.01502
KFC		**0.01197**		–

Abreviações: KOF – Krigagem Ordinária Funcional; KTVF – Krigagem Tempo-Variante Funcional; KFC – Krigagem Funcional por Campo.

[2] Nesses diagramas, retiramos os *outliers* para facilitar a visualização.

Tabela 4.9: *Sumário para o Erro Quadrático Médio por Amostra para o ponto centralizado s_0.*

Método – Base	Mínimo	1-Qua	Média	Mediana	3-Qua	Máximo	DP
KOF – Haar	0.012002	0.012051	0.012514	0.012073	0.012095	0.039457	0.002977
KOF – Bsplines	0.013947	0.014315	0.014633	0.014363	0.014432	0.029682	0.001652
KTVF – Haar	0.012002	0.012050	0.012119	0.012070	0.012090	0.013630	0.000231
KTVF – Bsplines	0.014239	0.014328	0.014351	0.014346	0.014370	0.014465	0.000035
KFC – Haar	0.012015	0.012049	**0.012069**	**0.012066**	0.012088	0.012153	0.000027

Abreviações: KOF – Krigagem Ordinária Funcional; KTVF – Krigagem Tempo-Variante Funcional; KFC – Krigagem Funcional por Campo.

Tabela 4.10: *Sumário para o Erro Quadrático Médio por Amostra para o ponto centralizado s_1.*

Método – Base	Mínimo	1-Qua	Média	Mediana	3-Qua	Máximo	DP
KOF – Haar	0.011848	0.011920	0.026226	0.011983	0.012115	0.724572	0.081861
KOF – Bsplines	0.014500	0.014976	0.015275	0.015035	0.015097	0.029605	0.001559
KTVF – Haar	0.011850	0.011926	0.015780	0.011984	0.012107	0.243905	0.025245
KTVF – Bsplines	0.014827	0.014991	0.015020	0.015025	0.015049	0.015189	0.000059
KFC – Haar	0.011840	0.011926	**0.011965**	**0.011963**	0.012001	0.012115	0.000064

Abreviações: KOF – Krigagem Ordinária Funcional; KTVF – Krigagem Tempo-Variante Funcional; KFC – Krigagem Funcional por Campo.

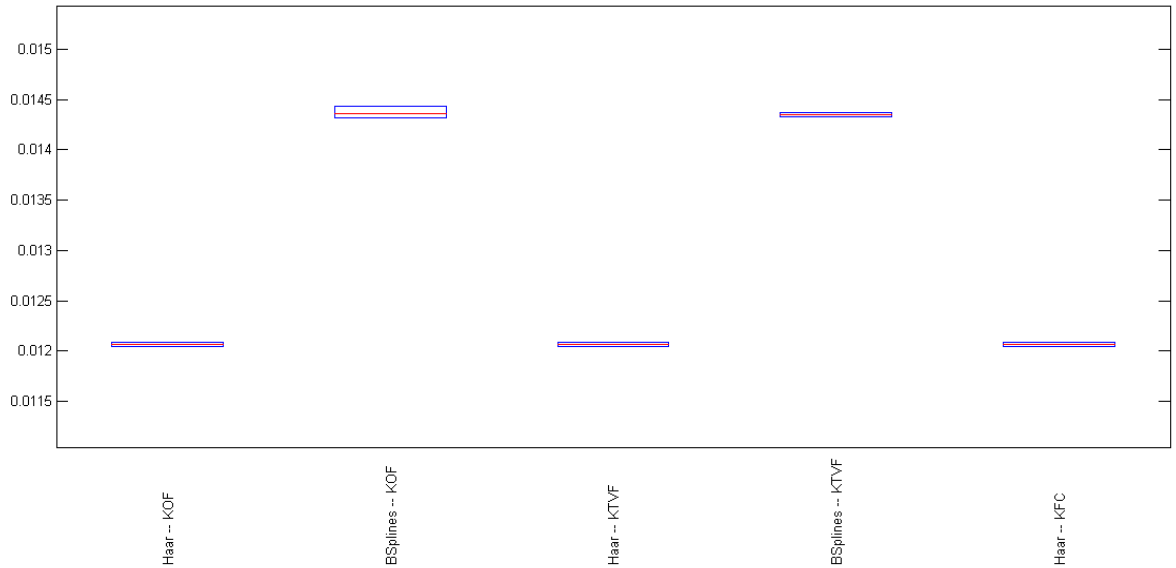


Figura 4.6: *Boxplot do Erro Quadrático Médio por Amostra para o ponto no centro da região s_0 para cada método de krigagem e para cada base no contexto de média não contínua.*

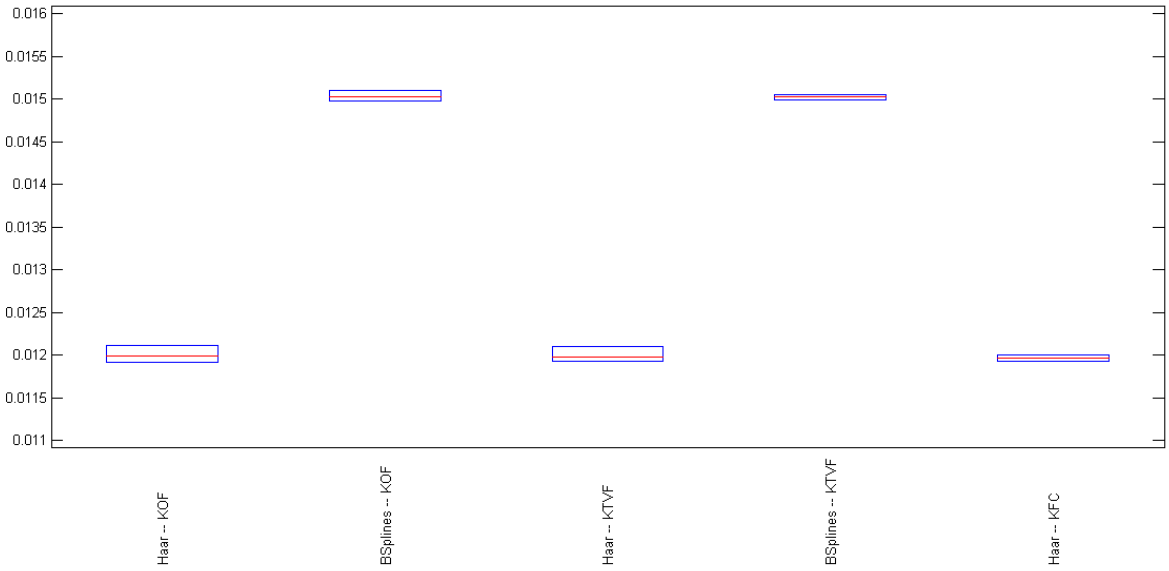


Figura 4.7: *Boxplot do Erro Quadrático Médio por Amostra para o ponto no canto da região s_1 para cada método de krigagem e para cada base no contexto de média não contínua.*

Tabela 4.11: *Sumário para o Erro Quadrático Médio por Amostra para o ponto no canto $\boldsymbol{s}_0$.*

Método – Base	Mínimo	1-Qua	Média	Mediana	3-Qua	Máximo	DP
KOF – Haar	-0.00475	0.00070	0.00164	0.00180	0.00267	0.01491	0.00215
KOF – Bsplines	-0.01592	-0.00346	-0.00141	-0.00248	-0.00136	0.04159	0.00692
KTVF – Haar	-0.00467	0.00017	0.00109	0.00121	0.00203	0.01426	0.00205
KTVF – Bsplines	-0.00573	-0.00303	-0.00251	-0.00259	-0.00181	-0.00018	0.00106
KFC – Haar	-0.00145	0.00077	0.00142	0.00142	0.00217	0.00482	0.00115

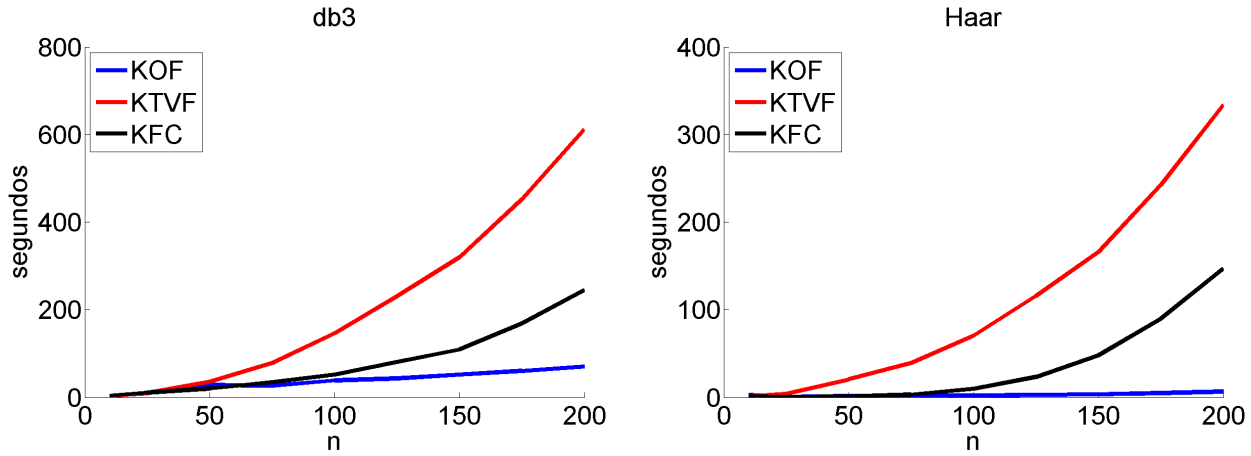
Abreviações: KOF – Krigagem Ordinária Funcional; KTVF – Krigagem Tempo-Variante Funcional; KFC – Krigagem Funcional por Campo.

Tabela 4.12: *Sumário para o Vício por Amostra para o ponto centralizado $\boldsymbol{s}_1$.*

Método – Base	Mínimo	1-Qua	Média	Mediana	3-Qua	Máximo	DP
KOF – db3	-0.01814	0.00499	0.00621	0.00720	0.00867	0.01259	0.00466
KOF – Bsplines	-0.01592	-0.00346	-0.00141	-0.00248	-0.00136	0.04159	0.00692
KTVF – db3	-0.01819	0.00931	0.01054	0.01145	0.01290	0.01682	0.00470
KTVF – Bsplines	-0.00573	-0.00303	-0.00251	-0.00259	-0.00181	-0.00018	0.00106
KFC – db3	0.00092	0.00554	0.00705	0.00719	0.00852	0.01256	0.00229

Abreviações: KOF – Krigagem Ordinária Funcional; KTVF – Krigagem Tempo-Variante Funcional; KFC – Krigagem Funcional por Campo.

Além do novo método proposto nessa tese, denominado Krigagem Funcional por Campo, ter menor Erro Quadrático Médio, ele é computacionalmente mais eficiente que a Krigagem Tempo-Variante Funcional, conforme gráficos das Figura 4.8. Observamos que as aproximação numéricas na Krigagem Tempo-Variante Funcional tornam esse método consideravelmente menos eficiente que a Krigagem Funcional por Campo. Nas tabelas 4.13(a) e 4.13(b), percebemos que o tempo de execução em segundos da Krigagem Tempo-Variável Funcional é quase três vezes o tempo de execução da Krigagem Funcional por Campo para $n = 200$, além do método KTVF ter crescimento ser exponencial. Ressaltamos que melhorias nas aproximações numéricas das integrais podem amenizar este problema, contudo com a Krigagem Funcional por Campo essa questão é contornada.



(a) Tempo de execução para ondaleta $db3$ em segundos. (b) Tempo de execução para ondaleta Haar em segundos.

Abreviações: KOF – Krigagem Ordinária Funcional; KTVF – Krigagem Tempo-Variante Funcional; KFC – Krigagem Funcional por Campo.

Figura 4.8: *Comparação do tempo de execução em segundos do métodos de interpolação espacial de curvas, Krigagem Ordinária Funcional, Krigagem Tempo-Variante Funcional e Krigagem Funcional por Campo, usando ondaletas Haar e $db3$.*

Tabela 4.13: *Tempo de execução para os métodos de krigagem, Krigagem Ordinária Funcional (KOF), Krigagem Tempo-Variante Funcional (KTVF) e Krigagem Funcional por Campo (KFC), para vários tamanhos amostrais n.*

(a) Tempo de execução para ondaletas $db3$ em segundos.

n	KOF	KTVF	KFC
10	3.44	2.09	3.59
25	9.29	9.42	10.64
50	27.97	35.62	20.08
75	26.12	78.03	34.10
100	38.46	145.46	51.84
125	42.87	230.73	80.29
150	51.39	319.54	109.05
175	60.40	451.61	168.09
200	70.43	611.44	244.60

(b) Tempo de execução para ondaletas Haar em segundos.

n	KOF	KTVF	KFC
10	2.69	1.11	1.04
25	0.45	4.01	0.12
50	1.22	20.31	1.04
75	1.34	39.16	3.16
100	1.89	70.20	9.71
125	2.78	115.76	23.23
150	3.39	165.75	47.88
175	4.70	242.38	89.58
200	6.63	333.89	146.70

Abreviações: KOF – Krigagem Ordinária Funcional; KTVF – Krigagem Tempo-Variante Funcional; KFC – Krigagem Funcional por Campo.

Após verificarmos que tanto o uso de ondaletas como o método de Krigagem Funcional por Campo melhoram a interpolação espacial de curvas, apresentamos no Capítulo 5 duas aplicações a dados reais: dados de temperatura média das províncias marítimas canadenses e dados do índice MP10 da região metropolitana de São Paulo.

Capítulo 5

Aplicação

Neste capítulo, ilustramos as três abordagens de krigagem para curvas apresentadas neste trabalho, Krigagem Ordinária Funcional, Krigagem Tempo-Variante Funcional e Krigagem Funcional por Campo, usando dois conjuntos de dados reais: um conjunto de dados de temperatura média no ano de 2000 em 82 estações meteorológicas localizadas nas províncias marítimas canadenses (Nova Scotia, New Brunswick e Prince Edward Island) e dados do índice MP10 da CETESB (Companhia Ambiental do Estado de São Paulo) no ano de 2014 em 14 estações meteorológicas na região metropolitana de São Paulo.

Em cada um dos conjuntos de dados escolhemos uma localidade $\boldsymbol{s}_i$, estimamos a curva $\chi_{\boldsymbol{s}_i}(t)$ usando as curvas $\chi_{\boldsymbol{s}_1}(t), \ldots, \chi_{\boldsymbol{s}_{i-1}}(t), \chi_{\boldsymbol{s}_{i+1}}(t), \ldots, \chi_{\boldsymbol{s}_n}(t)$ e criamos um gráfico com a curva interpolada pela krigagem e os valores observados $\chi_{\boldsymbol{s}_i}(t_j), j = 1, \ldots, 2^J$ da curva em $\boldsymbol{s}_i$, com o objetivo de ilustrar visualmente a eficácia dos métodos. Além disso, calculamos uma medida de Erro Quadrático Médio dada por

$$EQM = \sum_{j=1}^{2^J} \left(\hat{\chi}_{\boldsymbol{s}_i}(t_j) - \chi_{\boldsymbol{s}_i}(t_j)\right)^2$$

em que $\hat{\chi}_{\boldsymbol{s}_i}(t_j)$ é a estimativa da curva estimada no instante $t_j, = 1, \ldots, 2^J$ usando as curvas $\chi_{\boldsymbol{s}_1}(t), \ldots, \chi_{\boldsymbol{s}_{i-1}}(t), \chi_{\boldsymbol{s}_{i+1}}(t), \ldots, \chi_{\boldsymbol{s}_n}(t)$.

5.1 Temperatura Média Diária das Províncias Marítimas Canadenses

Nessa seção, aplicamos os métodos de Krigagem aos dados climáticos das províncias marítimas do Canadá: New Brunswick, New Scotia e Prince Edward. Em nosso conjunto de dados, consideramos a temperatura média diária registrada em 82 estações meteorológicas das províncias marítimas ao longo do ano de 2000. Na Figura 5.1, mostramos as estações selecionadas em nosso conjunto de dados e na Tabela B.1 do Apêndice B, apresentamos informações tais como ID, nome da estação, latitude e longitude das estações consideradas. Esses dados são bem difundidos na literatura e estão presentes no livro clássico de Ramsay (2006). Conforme Henao *et al.* (2010), as províncias marítimas cobrem uma área pequena e homogênea que satisfaz as condições de estacionaridade e isotropia descritas na introdução do Capítulo 3. Os dados utilizados nessa seção podem ser facilmente obtidos pelo sítio http://climate.weather.gc.ca/.

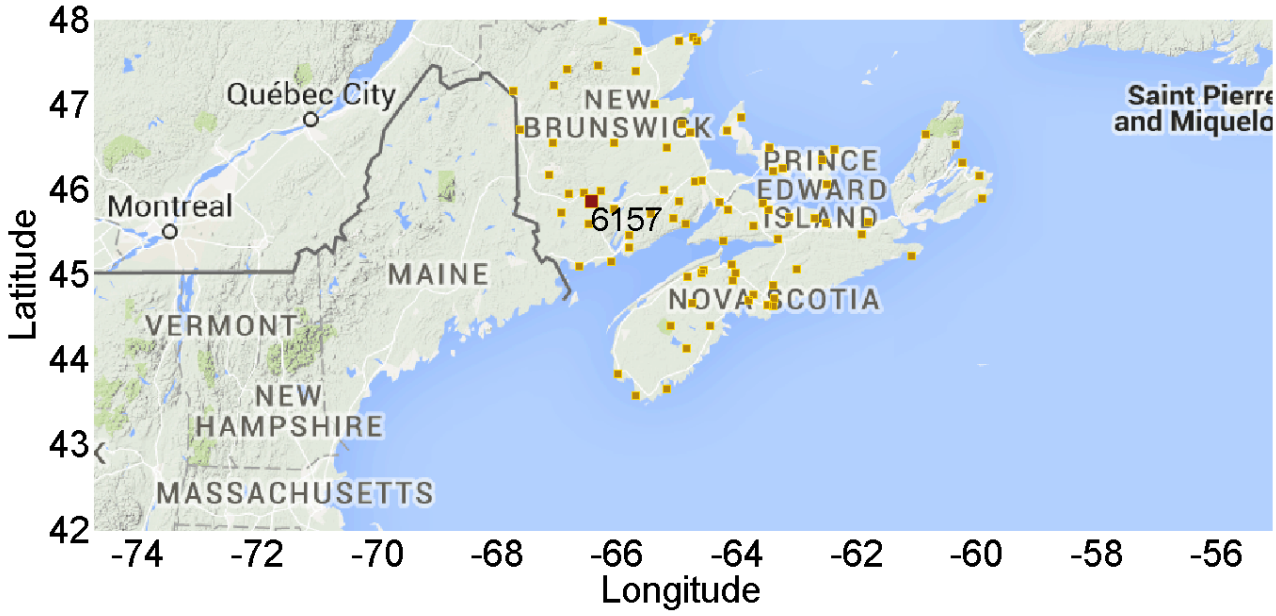
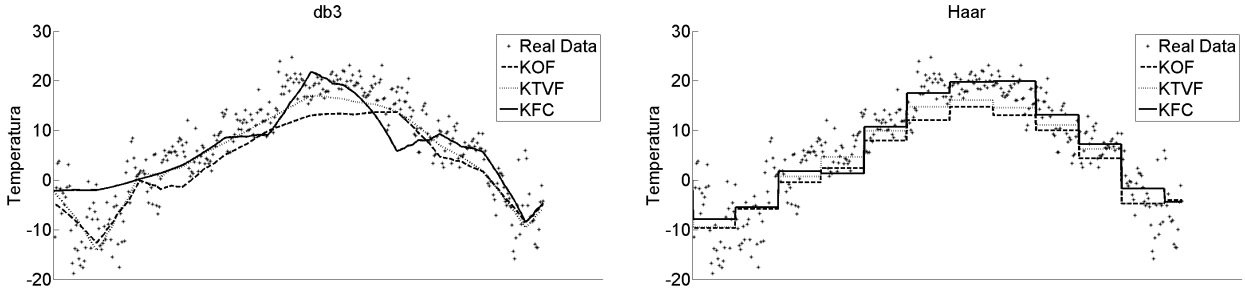


Figura 5.1: *Estações meteorológicas selecionadas das províncias marítimas.*

Para avaliar a performance dos métodos de Krigagem para a temperatura média diária, escolhemos a estação Fredericon A indicada pelo ID 6157 na Figura 5.1 (quadrado vermelho) e estimamos a temperatura média usando todas as outras estações destacadas no mapa (quadrados amarelos). Temos 82 estação em nosso conjunto de dados e, consequentemente, usamos 81 estações para estimar a curva de temperatura média na estação Fredericon A no ano 2000. Nas

Figuras 5.2(a) e 5.2(b), mostramos o ajuste da curva pelo modelo de Krigagem Ordinária, Krigagem Tempo-Variante Funcional e Krigagem Funcional por Campo usando ondaletas Haar e $db3$. Visualmente a Krigagem Tempo-Variante Funcional e a Krigagem Funcional por Campo interpolam melhor os dados do que o método de Krigagem Ordinário Funcional. Este comportamento é confirmado pelo Erro Quadrático Médio apresentado na Tabela 5.1. Repetimos esse processo de estimar a curva $\chi_{\boldsymbol{s}_i}(t)$ usando $\chi_{\boldsymbol{s}_1}(t), \ldots, \chi_{\boldsymbol{s}_{i-1}}(t), \chi_{\boldsymbol{s}_{i+1}}(t), \ldots, \chi_{\boldsymbol{s}_n}(t)$ e calcular $EQM = \sum_{j=1}^{2^J} \left(\hat{\chi}_{\boldsymbol{s}_i}(t_j) - \chi_{\boldsymbol{s}_i}(t_j)\right)^2$ para todas as localidades $\boldsymbol{s}_i, i = 1, \ldots, n$ e podemos verificar na Tabela 5.2 e na Figura 5.3 que, em média, a Krigagem Tempo-Variante Funcional e a Krigagem Funcional por Campo tem Erro Quadrático Médio menor que a Krigagem Ordinária Funcional.



(a) $db4$ (b) Haar

Abreviações: KOF – Krigagem Ordinária Funcional; KTVF – Krigagem Tempo-Variante Funcional; KFC – Krigagem Funcional por Campo; $db3$ – ondaleta Daubechies com $N = 3$.

Figura 5.2: *Curvas ajustadas pelos métodos de interpolação espacial para curvas, Krigagem Ordinária Funcional (KOF), Krigagem Tempo-Variante Funcional e Krigagem Funcional por Campo, usando ondaletas Haar e Daubechies com $N = 3$.*

Tabela 5.1: *Erro Quadrático Médio para a estação Fredericon A.*

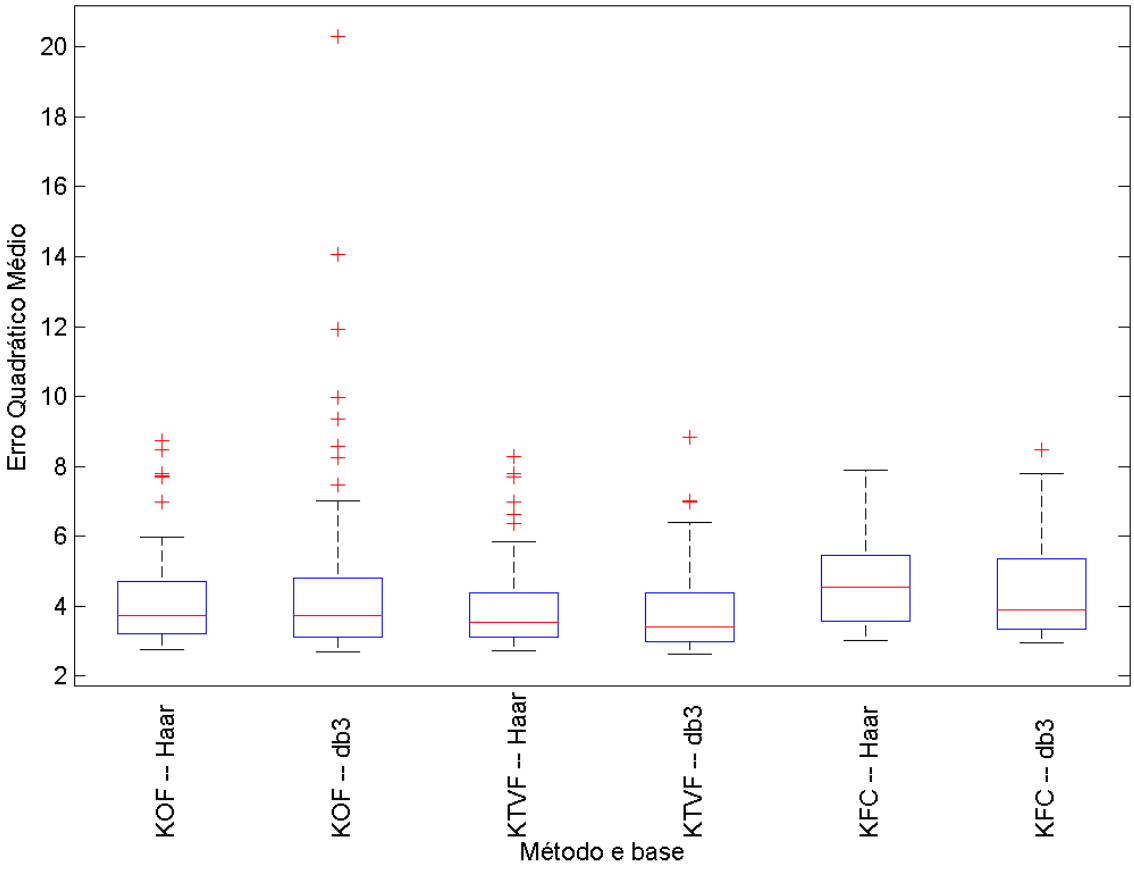
Método	Ondaleta	EQM	Ondaleta	EQM
KOF		28.26228		26.02128
KTVF	Haar	**22.86882**	db3	**18.93903**
KFC		25.312483		22.62712

Abreviações: KOF – Krigagem Ordinária Funcional; KTVF – Krigagem Tempo-Variante Funcional; KFC – Krigagem Funcional por Campo; $db3$ – ondaleta Daubechies com $N = 3$.

Tabela 5.2: *Sumário de EQM para todas estações.*

Método – Base	Mínimo	1-Qua	Média	Mediana	3-Qua	Máximo	DP	Soma
KTVF – db3	13.73	19.74	**175.24**	**30.47**	80.91	6844.12	768.21	14369.42
KTVF – Haar	15.13	22.55	188.49	34.21	79.47	3910.21	566.19	15456.07
KFC – Haar	20.76	36.00	266.34	95.50	231.20	2715.57	502.78	21839.80
KFC – db3	19.29	28.36	279.80	48.66	213.75	4773.36	653.29	22943.74
KOF – Haar	15.88	24.69	312.99	42.29	109.68	6315.45	961.14	25665.56
KOF – db3	14.58	22.34	8.08E+06	41.50	120.31	6.61E+08	7.30E+07	6.62E+08

Abreviações: KOF – Krigagem Ordinária Funcional; KTVF – Krigagem Tempo-Variante Funcional; KFC – Krigagem Funcional por Campo; db3 – ondaleta Daubechies com $N = 3$.



Abreviações: KOF – Krigagem Ordinária Funcional; KTVF – Krigagem Tempo-Variante Funcional; KFC – Krigagem Funcional por Campo; db3 – ondaleta Daubechies com $N = 3$.

Figura 5.3: *Boxplot de EQM para todas as 82 estações em escala logarítmica.*

5.2 Material Particulado Inalável – MP10

Poluição por partículas (também chamada de material particulado ou, brevemente, MP) é o termo para uma mistura de partículas sólidas e gotículas encontradas no ar. Algumas partículas, tais como poeira, sujeira, fuligem ou fumaça, são grandes e escuras o suficiente para serem vistas a olho nu. Outras são tão pequenas que apenas podem ser detectadas usando um microscópio eletrônico. Poluição por partículas é composta por "partículas grossas inaláveis" com diâmetro maior que $2.5\mu m$ e menor que $10\mu m$ [1] e "partículas finas" com diâmetro menor que $2.5\mu m$. Algumas partículas, conhecidas como partículas primárias, são emitidas diretamente de uma fonte, como construções civis, estradas não pavimentadas, chaminés de fábricas ou incêndios. Outras são resultados de reações complexas de químicos na atmosfera tais como dióxido sulfúrico e óxido de nitrogênio que são emitidos de usinas térmicas movidas a combustíveis fósseis, indústrias e automóveis [2]. A concentração de microgramas "Material Particulado Grosso" por m^3, referenciada como MP10, é um índice fortemente relacionado com doenças respiratórias, problemas cardiovasculares e taxa de mortalidade (vide, por exemplo, Ostro *et al.*, 1999; Pope III e Dockery, 2006; Pope III *et al.*, 1991; Schwartz *et al.*, 1993; Seaton *et al.*, 1995, para estudos sobre os efeitos da concentração de material particulado na saúde) e, consequentemente, seu monitoramento e análise são indispensáveis. Nessa Seção, usamos o conjunto de dados composto por medições de MP10 realizadas pela CETESB (Companhia Ambiental do Estado de São Paulo) em 14 localidades listadas na Tabela C.1 do Apêndice C no ano 2014 para interpolar a concentração de microgramas de material particulado grosso por metro cúbico em um local não monitorado pela CETESB usando os três métodos de Krigagem para curvas introduzidos no Capítulo 3: Krigagem Ordinária Funcional, Krigagem Tempo-Variante Funcional e Krigagem Funcional por Campo. Esses dados podem ser obtidos pelo endereço http://www.cetesb.sp.gov.br/.

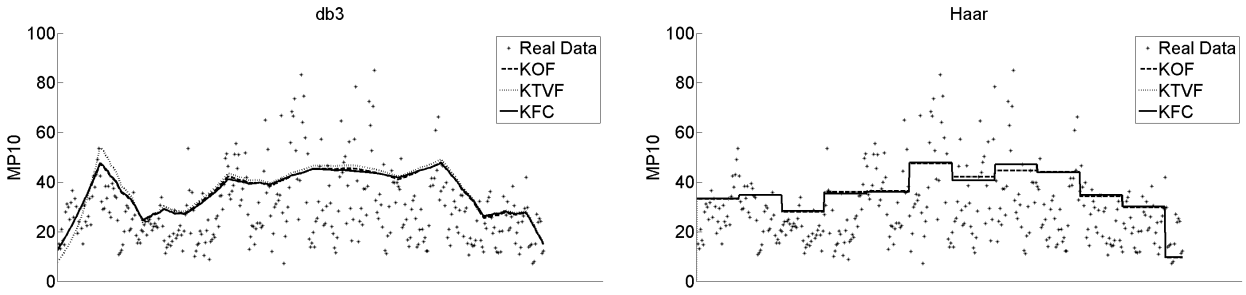
Para avaliar a performance dos métodos de Krigagem para o índice MP10, escolhemos a estação Parque Dom Pedro II da CETESB identificada pelo ID 72 na Figura 5.4 (quadrado vermelho) e estimamos o índice MP10 usando todas as outras estações destacadas no mapa (quadrados amarelos). Temos 14 estação em nosso conjunto de dados e, consequentemente, usamos 13 estações

[1] A sigla μm se refere a micrômetro.

[2] Para maiores informações sobre poluição por partícula, visite a página oficial da agência americana *United States Environmental Protection Agency* http://www3.epa.gov/airquality/particlepollution/basic.html.

Figura 5.4: *Estações meteorológicas na Região Metropolitana de São Paulo.*

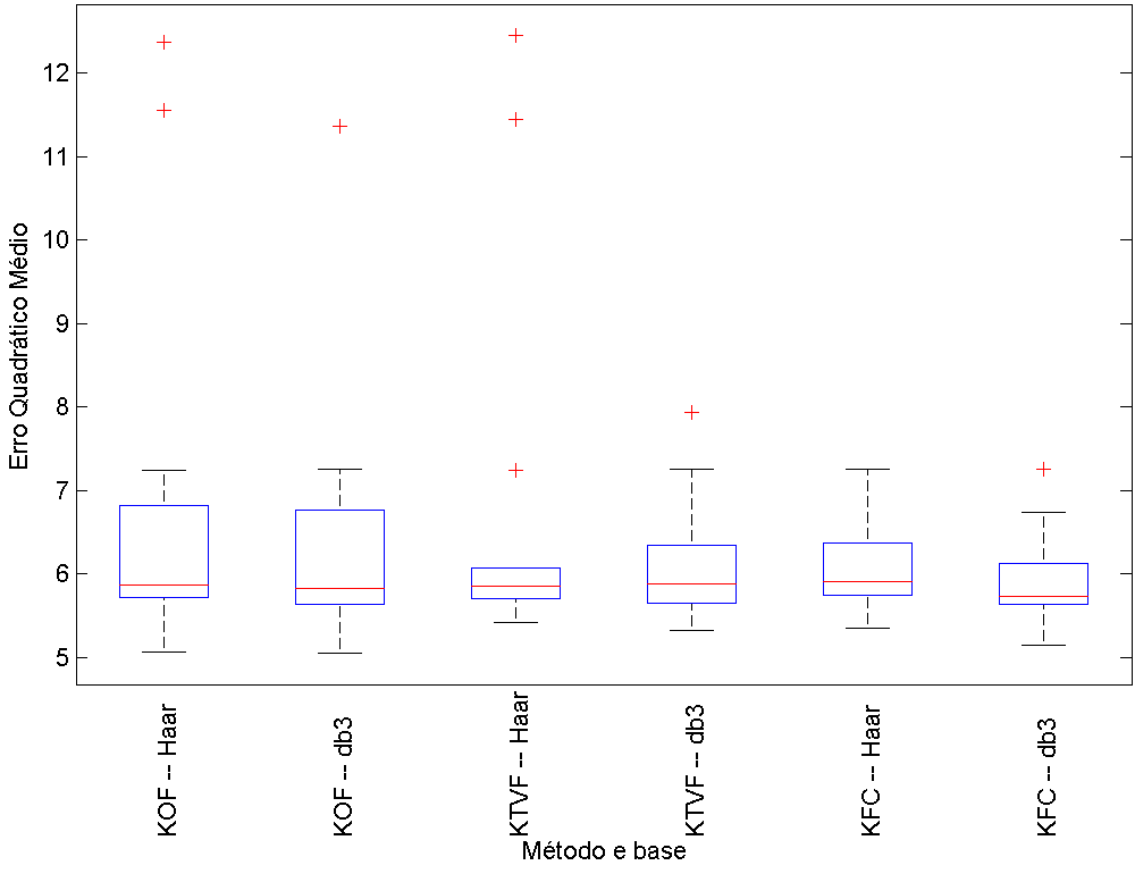
para estimar a curva de MP10 no ano 2014 na estação Parque Dom Pedro II. Nas Figuras 5.5(a) e 5.5(b), mostramos o ajuste da curva pelo modelo de Krigagem Ordinária Funcional, Krigagem Tempo-Variante Funcional e Krigagem Funcional por Campo. Diferentemente dos dados de temperatura média do Canadá, visualmente os três métodos aparentam ter um desempenho parecido. Contudo, ao analisarmos a Tabela 5.4 e a Figura 5.6, em que apresentamos estatísticas descritivas e o Boxplot em escala logarítmica do EQM calculado para todas estação, notamos que a Krigagem Funcional por Campo é o método de interpolação espacial de curvas com menor Erro Quadrático Médio, em concordância com o estudo de simulação co Capítulo 4.



(a) $db3$ (b) Haar

Abreviações: KOF – Krigagem Ordinária Funcional; KTVF – Krigagem Tempo-Variante Funcional; KFC – Krigagem Funcional por Campo; $db3$ – ondaleta Daubechies com $N = 3$.

Figura 5.5: *Curvas ajustadas pelos métodos de interpolação espacial para curvas, Krigagem Ordinária Funcional (KOF), Krigagem Tempo-Variante Funcional e Krigagem Funcional por Campo, usando ondaletas Haar e Daubechies com $N = 3$.*



Abreviações: KOF – Krigagem Ordinária Funcional; KTVF – Krigagem Tempo-Variante Funcional; KFC – Krigagem Funcional por Campo; db3 – ondaleta Daubechies com $N = 3$.

Figura 5.6: *Boxplot de EQM para todas as 14 estações em escala logarítmica.*

Tabela 5.3: *Erro Quadrático Médio para a estação Parque Dom Pedro II.*

Método	Ondaleta	EQM	Ondaleta	EQM
KOF		236.11		**224.48**
KTVF	Haar	**235.66**	db3	249.75
KFC		238.04		225.19

Abreviações: KOF – Krigagem Ordinária Funcional; KTVF – Krigagem Tempo-Variante Funcional; KFC – Krigagem Funcional por Campo; $db3$ – ondaleta Daubechies com $N = 3$.

Tabela 5.4: *Sumário de EQM para todas estações.*

Método	Mínimo	1-Qua	Média	Mediana	3-Qua	Máximo	DP	Soma
KFC – db3	172.17	279.04	**430.45**	**310.32**	454.55	1406.20	323.20	6026.35
KFC – Haar	211.56	312.56	525.06	369.69	584.09	1423.64	386.25	7350.78
KTVF – db3	204.59	283.30	647.73	356.03	564.63	2796.38	704.11	9068.28
KOF – db3	155.46	280.56	6631.50	340.32	870.12	8.63E+04	2.29E+04	9.28E+04
KOF – Haar	158.35	303.48	2.47E+04	352.88	918.39	2.36E+05	6.67E+04	3.45E+05
KTVF – Haar	224.35	300.57	2.54E+04	349.98	430.48	2.57E+05	7.11E+04	3.55E+05

Abreviações: KOF – Krigagem Ordinária Funcional; KTVF – Krigagem Tempo-Variante Funcional; KFC – Krigagem Funcional por Campo; $db3$ – ondaleta Daubechies com $N = 3$.

Capítulo 6

Estudos Futuros

Desde os trabalhos iniciais Henao (2009), Henao *et al.* (2010) e Henao *et al.* (2011), várias melhorias em métodos de krigagem para Dados Funcionais foram sugeridas, contudo ainda existem lacunas que acreditamos que precisam ser superadas e que são descritas a seguir.

Propriedades Assintóticas

Devido ao caráter recente e inovador dos métodos de krigagem para Dados Funcionais descritos no Capítulo 3 há apenas uma checagem via simulação das propriedades assintóticas, principalmente a consistência. Ou seja, é necessário demonstrar que, sob a condição de estacionaridade fraca pontual e isotropia [1], as três abordagens de krigagem propostas nesse trabalho são, por exemplo, consistentes.

Abordagem Bayesiana

Uma possibilidade não explorada nessa tese é o uso da abordagem Bayesiana. Em Geoestatística, há alguns trabalhos usando abordagem Bayesiana. Omre e Halvorsen (1989) apresentaram os métodos de Krigagem Ordinária e Krigagem Universal no contexto de uma abordagem Bayesiana. Pilz e Spöck (2008) propuseram um método de Krigagem denominado Krigagem Trans-Gaussiana Bayesiana para quando a suposição de que temos um campo aleatório Gaussiano e estrutura de covariância conhecida não é razoável. Diggle e Ribeiro Jr (2002) adotaram o paradigma Bayesiano

[1] Vide a introdução do Capítulo 3 para a definição de estacionaridade fraca pontual e isotropia para uma amostra funcional georeferenciada.

de estimação para derivar distribuições preditivas condicionais nos dados e apresentaram um algoritmo para amostrar dessas distribuições preditivas. Tais amostras podem ser usadas para predizer valores da variável sob estudo ou de qualquer função linear ou não linear como, por exemplo, a probabilidade dessa variável exceder um limite. De Oliveira *et al.* (1997) estenderam a teoria de krigagem do contexto de campo aleatório Gaussiano para campo aleatório transformado, em que uma transformação nos dados é realizada para moderar o afastamento da normalidade e a transformação pertence a uma família paramétrica de transformações. Fuentes (2001) introduziu um novo método de interpolação espacial para processos não estacionários. Nesse método, o campo é representado localmente como um campo aleatório isotrópico e estacionário, mas os parâmetros podem variar no espaço e uma abordagem Bayesiana é proposta para incorporar a incerteza na estimação dos parâmetros da covariância. Na ausência de um amplo estudo comparando a performance dos métodos de Krigagem, Moyeed e Papritz (2002) estudaram várias abordagens de comparação, incluindo abordagens Bayesianas, em um conjunto de dados de concentração de cobalto (Co) e cobre (Cu) em 2649 localidades em *Scottish Border*, região administrativa da Escócia - Reino Unido. Brown *et al.* (1994) desenvolveram um método de Krigagem para campos aleatórios multivariados usando abordagem Bayesiana, em que obtiveram a distribuição a *posteriori* para vetores não observados e para os parâmetros do modelo. A ideia seria adaptar esses métodos de Krigagem para Dados Funcionais, de maneira análoga aos trabalhos de Ramón G. Heano no final da década de 2000 em Henao (2009); Henao *et al.* (2010, 2011).

Outros Modelos

Um terceiro modelo de Krigagem para Dados Funcionais foi proposto no final da década 2000 em Henao (2009) e em Nerini *et al.* (2010) e ele não foi estudado nem implementado usando ondaletas nem sua performance foi analisada. Mais precisamente, dado uma amostra funcional $\chi_{\boldsymbol{s}_1}(t), \ldots, \chi_{\boldsymbol{s}_n}(t)$ isotrópica e pontualmente fracamente estacionária conforme descrito na introdução do Capítulo 3, uma estimativa para a curva $\chi_{\boldsymbol{s}_0}(t)$ em um ponto não monitorado $\boldsymbol{s}_0$ é a solução do problema de otimização não linear descrito por

$$\hat{\chi}_{\boldsymbol{s}_0}(t) = \sum_{i=1}^{n} \int_0^1 \lambda_i(t,u)\chi_{\boldsymbol{s}_i}(u)du, \quad \forall t \in [0,1] \text{ sujeito a } \mathrm{E}\{\hat{\chi}_{\boldsymbol{s}_0}(t)\} = \chi_{\boldsymbol{s}_0}(t), \quad \forall t \in [0,1]$$

em que $\lambda(.,.) : [0,1] \to \mathbb{R}$ e $\chi_{\boldsymbol{s}_i} : [0,1] \to \mathbb{R}$ precisam ser aproximados usando uma base de funções e B-Splines foram usadas. Além disso, notamos que nesse projeto optamos pelo uso de estimadores não paramétricos para o semivariograma, mas uma alternativa possível é usar estimadores paramétricos conforme descrito em Banerjee *et al.* (2014); Genton e Kleiber (2015). Adicionalmente, podemos substituir a aproximação numérica de integrais por retângulos descrita na Seção 3.2 por métodos mais eficientes e precisos, como, por exemplo, uma aproximação por trapézios.

Implementação

Um dos subprodutos desse projeto de doutorado é a *Toolbox* do MATLAB denominada *krigingFDA*. Contudo, para o seu uso além de ter uma licença válida do MATLAB, é preciso ter as seguintes *Toolbox* instaladas

– *Mapping Toolbox*

– *Optimization Toolbox*

– *Statistics and Machine Learning Toolbox*

– *Econometrics Toolbox*

o que pode invializar o uso das implementações descritas no Apêndice A para pequenas empresas ou centros de pesquisa emergentes. Neste contexto, uma "tradução" da *Toolbox krigingFDA* para plataformas livres como Octave (John W. Eaton e Wehbring, 2015) e R R Core Team (2015) é desejável. O Octave tem uma sintaxe próxima do MATLAB, sendo considerada uma versão livre do mesmo, mas seu uso é pouco disseminado entre a comunidade estatística. O R, por outro lado, tem uma sintaxe bastante distante do MATLAB, mas seu uso é muito mais disseminado para análises estatísticas.

Amostras Funcionais de Grande Dimensão

Um ponto que devemos destacar é que todos métodos de Krigagem para Dados Fucionais descritos no Capítulo 3 são combinações lineares de curvas ou de coeficientes da projeção das curvas em um espaço de aproximação. E quando o número de elementos da amostra funcional cresce, o número de parâmetros que precisamos estimar também aumenta. Na verdade, métodos de krigagem envolvem inversão de matrizes cuja complexidade numérica é da ordem de pelo menos $O(n^3)$. Não fomos capazes de encontrar soluções na literatura para esse tema no contexto de Dados Funcionais, mas em Geoestatística existem algumas propostas e Sun *et al.* (2012) faz uma revisão dos métodos de Geoestatística quando a amostra é grande. Merecem destaque o trabalho de Banerjee *et al.* (2008), denominado processo preditivo espacial, em que o processo espacial original é projetado em um subespaço gerado pelas realizações do processo original em conjunto pré-especificado de localidades e a proposta de Cressie e Johannesson (2008) em que os autores consideraram uma classe de matrizes de variância-covariância Σ de ordem $n \times n$ tal que a inversa Σ^{-1} pode ser obtida invertendo matrizes de ordem $r \times r$ com r fixo e $r < n$, e, consequentemente, a complexidade numérica dos métodos de Krigagem decai de $O(n^3)$ para $O(nr^2)$, além disso essa abordagem é capaz de tratar simultanemanente com o tamanho da amostra e a heterogeneidade espacial.

Apêndice A

Documentação da *Toolbox krigingFDA* do MATLAB

Neste apêndice, documentamos a *Toolbox* do MATLAB desenvolvida durante este projeto e denominada *krigingFDA*.

A *Toolbox krigingFDA* tem quatorze funções:

`FieldKriging_Haar` Esta função implementa o método Krigagem Funcional por Campo usando ondaleta Haar.

`chi_Haar` Esta função calcula a curva estimada usando ondaleta Haar.

`FieldHatchi_Haar` Esta função calcula a curva $\chi_{\boldsymbol{s}_0}(t)$ no instante t no ponto não monitorado $\boldsymbol{s}_0$ para a ondaleta Haar no modelo de Krigagem Funcioanl por Campo.

`FieldKriging_dbN` Esta função implementa o médodo de Krigagem Funcional por Campo usando ondaleta da família Daubechies.

`chi_dbN` Esta função calcula curva estimada usando ondaleta Daubechies

`FieldHatchi_dbN` Esta função calcula a curva $\chi_{\boldsymbol{s}_0}(t)$ no instante t no ponto não monitorado $\boldsymbol{s}_0$ para a família Daubechies no modelo de Krigagem Funcional por Campo.

`FunctionalKriging` Esta função é uma implementação para o método Krigagem Ordinária Funcional usando ondaletas Haar.

`PointwiseFuncKriging` Implementação do método de Krigagem Tempo-Variante Funcional usando ondaletas Haar.

`coef_Haar` Calcula os coeficientes de uma curva suavizada usando ondaletas Haar

`coef_dbN` Calcula os coeficientes de uma curva suavizada usando ondaletas Daubechies

`FuncKriging_dbN` Implementação do método de Krigagem Ordinária Funcional usando ondaletas Daubechies.

`hatchi_s0` Calcula a curva estimada $\chi_{\boldsymbol{s}_0}(t)$ no instante t para o modelo de Krigagem Ordinária Funcional para ondaletas Daubechies.

`PointwiseFuncKriging_dbN` Implementação do método de Krigagem Tempo-Variante Funcional para ondaletas Daubechies.

`hatchi` Calcula a curva estimada $\chi_{\boldsymbol{s}_0}(t)$ no instante t para o modelo de Krigagem Tempo-Variante Funcional para ondaletas Daubechies.

A descrição de cada uma dessas funções é descrita em uma seção organizada em quatro partes: objetivo, sintaxe, descrição e código fonte.

A.1 `FieldKriging_Haar`

Objetivo

Esta função implementa o método descrito na Seção 3.3 em que desejamos encontrar o vetor de coeficientes $\boldsymbol{a}_0$ da projeção da curva $\chi_{\boldsymbol{s}_0}(t)$ em um ponto não monitorado $\boldsymbol{s}_0$ usando ondaletas Haar.

Sintaxe

```
[a0, mL, mcoef]=FieldKriging_Haar(mdata, lat, long, latData, longData, J, tol);
```

Descrição

A função `FieldKriging_Haar` tem sete argumentos de entrada:

i. `mdata` é um matriz em que a i-ésima coluna é a série temporal $\chi_{\boldsymbol{s}_i}(t_1), \ldots, \chi_{\boldsymbol{s}_i}(t_n)$ observada da curva $\chi_{\boldsymbol{s}_i}(t)$

ii. `lat` é a latitude do ponto não monitorado $\boldsymbol{s}_0$

iii. `long` é a longitude do ponto não monitorado $\boldsymbol{s}_0$

iv. `latData` é um vetor coluna contendo as latitudes dos pontos $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$ das curvas em nossa amostra funcional

v. `longData` é um vetor coluna contendo as longitudes dos pontos $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$ das curvas em nossa amostra funcional

vi. `J` é o nível de aproximação na análise de multirresolução

vii. `tol` é o nível de tolerância para o cálculo do semivariograma conforme explicado no Capítulo 3. Esse argumento é opcional e seu valor padrão é 0.1

e tem três saídas

a. `a0` é um vetor coluna de coeficientes da projeção da curva $\chi_{\boldsymbol{s}_0}(t)$ no ponto não monitorado $\boldsymbol{s}_0$ usando método de krigagem introduzido na Seção 3.3

b. `mL` é um matriz em que a i-ésima coluna contém a diagonal da matriz Λ_i

c. `mcoef` é uma matriz em que a i-ésima coluna armazena os coeficientes da curva $\chi_{\boldsymbol{s}_i}(t)$ no ponto $\boldsymbol{s}_i$

Código Fonte

Listing A.1: *Arquivo* `FieldKriging_Haar.m`.

```
function [ a0, mL, mcoef ] = FieldKriging_Haar( mdata, lat, long, latData,...
longData, J, tol)
%FieldKriging_Haar Interpolation of random multivariate created by the
% coefficients of curves in the functional dataset. This function is
% specific for Haar basis.
%Input:
%   1)mdata: matrix data with mdata(:,i) = (y_1, \dots, y_n)^\top. The
%   first dimension is related to time and the second dimension is related
%   to space.
%   2)lat: latitude of the spot in interest
%   3)long: longitude of the spot in interest
%   4)latData: column vector with latitude from the sampled curves.
%   5)longData: column vector with longitude from the sampled curves.
%   6)J: approximation level
%   7)tol: tolerance bandwidth used in the semivariogram estimation. This
%   parameter is optional and the default value is 0.1.
%Output:
%   1)a0: estimated coefficients of curve at s0
%   2)mL: diagonal of matrices in the model
%   3)mcoef: coefficients matrix with mcoef(i,:) contains the parameters
%   associated with the inth curve in the functional sample.


    %Optional argument. Tolerance bandwidth used in the semivariogram
    %estimation.
    if nargin == 6
        tol = 0.1;
    end

    %points number in the space dimension
    n = size(mdata, 2);

    %points number in the time dimension
    nTime = 2^ceil(log2( size(mdata, 1) ));

    %basis size
    nbasis = 2^J;

    tic;
    %coefficient matrix
    mcoef = NaN(n, nbasis);
    for i = 1:size(mdata, 2)
        y = mdata(:,i);
        y = [y(:);zeros(nTime - size(y, 1),1)];
        mcoef(i,:) = coefHaar(y, J);
    end

    %Distance matrix
    mdist = NaN(n);
    for i = 1:n
        mdist(i,:) = distance('gc', latData(i), longData(i), latData, longData);
    end

    %Filling the independent matrix
```

```
    A = zeros((n+1) * (nbasis));
    for i = 1:(n-1)
        for  j = (i+1):n
            rows = ((i-1) * nbasis + 1): (i * nbasis);
            cols = ((j-1) * nbasis + 1): (j * nbasis);
            A(rows, cols) = G(mcoef, mdist, mdist(i,j), tol) .* eye(nbasis);
            A(cols, rows) = A(rows, cols);
        end
    end
    cols = (nbasis * n+ 1):(nbasis * (n + 1));
    for i = 1:n
        rows = ((i-1) * nbasis + 1) : (i * nbasis);
        A(rows,cols) = eye(nbasis);
        A(cols, rows) = eye(nbasis);
    end

    %Filling the independent matrix
    b = NaN( (n+1) * nbasis, 1);
    %last elements
    rows = ( n * nbasis + 1 ): ((n+1)*nbasis);
    b(rows) =  ones(nbasis, 1);
    %column vector of (||s0-s1||, \dots, ||s0-sn||)^\top
    vdist = distance('gc', lat, long, latData, longData);
    for i = 1:n
        rows = ((i-1) * nbasis + 1): (i * nbasis);
        b(rows) = diag( G(mcoef, mdist, vdist(i), tol) );
    end
    tempo = toc;
    disp(['Filling the coefficients matrix consumed ', num2str(tempo), ' seconds.']);

    %Solving the linear system
    x = lsqr(A, b, 0.000001, 200000);
    mL = reshape(x(1:(n*nbasis)),nbasis, n);

    %Calculating a_0
    a0 = zeros(nbasis, 1);
    for i = 1:n
        a0 = a0 + diag( mL(:,i) ) * mcoef(i,:)';
    end

end

function [ mG ] = G( mcoef, mdist, h, tol)
%G This function computes the semivariogram for a multivariate random
%field.
%Input
%   1)mcoef: mcoef: coefficients matrix with mcoef(i,:) contains the parameters
%   associated with the inth curve in the functional sample.
%   2) mdist: distance matris with mdist(i, j) = ||s_i - s_j||
%   3)h: distance target
%   4)tol: tolerance bandwidth. Optional argument and default value is
%   0.01.
%Output:
%   1)mG: estiamted semivariogram at h

    %Optional argument. Tolerance bandwidth.
    if nargin == 3
        tol = 0.1;
    end
```

```
    %searching for points with ||s_i - s_0|| \in (h - \epsilon, h +
    %\epsilon)
    [row, col] = find( (mdist > (h - tol)) & (mdist < (h + tol)));
    %checking with index is empty...
    while numel(row) == 0
        disp(['There are no sampled points with distance ', num2str(h),...
        ' inside the tolerance ', num2str(tol)]);
        tol = tol + 0.001;
        [row, col] = find( (mdist > (h - tol)) & (mdist < (h + tol)));
    end


    %output
    mG = zeros(size(mcoef, 2));
    for i = 1:numel(row)
        mG = mG + (mcoef(row(i),:) - mcoef(col(i),:))' *...
        (mcoef(row(i),:) - mcoef(col(i),:));
    end
    mG = mG / (2 *  numel(col));

end
```

A.2 chi_Haar

Objetivo

Esta função calcula o valor da curva estimada usando ondaletas Haar instante t dado os coeficientes calculados pela função `coef_Haar`.

Sintaxe

```
f  = chiHaar( t, J, c )
```

Descrição

A função `chi_Haar` tem três argumentos:

i. `t` é um vetor coluna de instantes para calcularmos o valor da curva

ii. `J` é o nível de aproximação

iii. `c` é um vetor coluna contendo os coeficientes estimados usando a função `coef_Haar`

e retorna apenas um vetor coluna contendo os valores de curva $\chi(t)$ nos elementos do vetor coluna `t`.

Código fonte

```
function [ f ] = chiHaar( t, J, c )
%chiHaar This function computes the smoothed curve using the Haar basis
%given the values t, the approximation level J and the coefficients vector.
%Input:
```

```
%   1)t: column vector of sample values
%   2)J: approximation level
%   3)c: Haar coefficients column vector
%Output:
%   1)f: column vector of estimated values with the same dimension of t.

    %vectorizing t and c
    t = t(:);
    c = c(:);

    f = NaN(size(t));
    for i = 1:length(t)
        if floor(t(i) * 2^J) < 2^J
            k = floor(t(i) * 2^J) + 1;
        else
            k = floor(t(i) * 2^J);
        end
        f(i) = 2^(J/2) * c(k);
    end

end
```

A.3 FieldHatchi_Haar

Objetivo

Esta função calcula a curva $\chi_{\boldsymbol{s}_0}(t)$ em t no ponto não monitorado $\boldsymbol{s}_0$ dado a diagonal das matrizes $\Lambda_i, i = 1, \ldots, n$ e o coeficientes das curvas $\chi_{\boldsymbol{s}_i}(t), i = 1, \ldots, n$ usando ondaletas Haar conforme descrito na Seção 3.3.

Sintaxe

```
[ f ] = FieldHatchi_Haar( t, mL, mcoef );
```

Descrição

A função `FieldHatchi_Haar` tem três argumentos de entrada

i. `t` é um vetor coluna de pontos em que desejamos calcular o valor estimado da curva $\chi_{\boldsymbol{s}_0}(t)$ no ponto $\boldsymbol{s}_0$

ii. `mL` é uma matriz em que a i-ésima coluna contém a diagonal da matriz Λ_i do modelo de krigagem descrito na Seção 3.3

iii. `mcoef` é uma matriz em que a i-ésima coluna é preenchida com os coeficientes da curva $\chi_{\boldsymbol{s}_i}(t)$

e retorna um vetor coluna `f` $= (\chi_{\boldsymbol{s}_0}(t_1), \ldots, \chi_{\boldsymbol{s}_0}(t_m))^\top$ com o mesmo número linhas de `t`.

Código Fonte

Listing A.2: *Arquivo* `FieldHatchi_Haar.m`.

```
function [ f ] = FieldHatchi_Haar( t, mL, mcoef )
%FieldHatchi_Haar This function computes \hat\chi_{s0}(t) = (\sum_{i=1}^{n}
%\Lambda_i a_{si})^\top \bm{\phi}.
%Input:
%   t: column vector (t_1, \dots, t_m)^\top
%   mL: matrix where the ith column contains diag(\Lambda_i)
%   mcoef: matrix A with size(A) = [2^J + 2N -2, nSpatial] (where nSpatial
%   is the number of spatial points in the sample) with the ith column
%   contains the coefficient of \hat\chi(t)
%Output:
%   f: column vector with the estimates (\hat\chi_{so}(t_1), \dots,
%   \hat\chi_{s0}(t_m))^\top

    %approximation level
    J = log2(size(mcoef, 1));

    %coefficients of chi_{s0}(t)
    a0 = zeros(2^J, 1);
    for i = 1:size(mcoef, 2)
        a0 = a0 + diag(mL(:,i)) * mcoef(:,i);
    end

    f = chiHaar( t, J, a0 );
end
```

A.4 `FieldKriging_dbN`

Objetivo

Esta função calcula a curva $\chi_{\boldsymbol{s}_0}(t)$ em t no ponto não monitorado $\boldsymbol{s}_0$ dado a diagonal das matrizes $\Lambda_i, i = 1, \ldots, n$ e o coeficientes das curvas $\chi_{\boldsymbol{s}_i}(t), i = 1, \ldots, n$ usando ondaletas Daubechies conforme descrito na Seção 3.3.

Sintaxe

```
[a0, mL, mcoef]=FieldKriging_dbN(mdata, lat, long, latData, longData, Iter, J, N, tol);
```

Descrição

A função `FieldKriging_dbN` tem nove argumentos de entrada:

i. `mdata` é um matriz em que a i-ésima coluna é a série temporal $\chi_{\boldsymbol{s}_i}(t_1), \ldots, \chi_{\boldsymbol{s}_i}(t_n)$ observada da curva $\chi_{\boldsymbol{s}_i}(t)$

ii. `lat` é a latitude do ponto não monitorado $\boldsymbol{s}_0$

iii. `long` é a longitude do ponto não monitorado $\boldsymbol{s}_0$

iv. `latData` é um vetor coluna contendo as latitudes dos pontos $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$ das curvas em nossa amostra funcional

v. `longData` é um vetor coluna contendo as longitudes dos pontos $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$ das curvas em nossa amostra funcional

vi. `Iter` é uma número inteiro positivo tal que $\frac{1}{2^{Iter}}$ precisão numérica usada nas aproximações de integrais envolvendo a função escala

vii. `J` é o nível de aproximação na análise de multirresolução

viii. `N` é um número inteiro positivo que indexa a ordem na família Daubechies

ix. `tol` é o nível de tolerância para o cálculo do semivariograma conforme explicado no Capítulo 3. Esse argumento é opcional e seu valor padrão é 0.1

e retorna três argumentos

a. a_0 é o vetor colunas contendo os coeficientes estimados da curva $\chi_{\boldsymbol{s}_0}(t)$

b. `mL` é uma matriz em que a i-ésima coluna contém a diagonal principal do matriz Θ_i do modelo Krigagem Funcional por Campo

c. `mcoef` é uma matriz cuja i-ésima coluna contém os coeficientes da curva $\chi_{\boldsymbol{s}_i}(t)$

Código Fonte

Listing A.3: *Arquivo* `FieldKriging_dbN.m`.

```
function [ a0, mL, mcoef ] = FieldKriging_dbN( mdata, lat, long, latData, longData,...
                                                Iter, J, N, tol)
%model1 Interpolation of random multivariate created by the coefficients of
%curves in the functional dataset.
%Input:
%   1)mdata: matrix data with mdata(:,i) = (y_1, \dots, y_n)^\top. The
%   first dimension is related to time and the second dimension is related
%   to space.
%   2)lat: latitude of the spot in interest
%   3)long: longitude of the spot in interest
%   4)latData: column vector with latitude from the sampled curves.
%   5)longData: column vector with longitude from the sampled curves.
%   6)Iter: order of dyadic points \phi at values \dfrac{l}{2^Iter}
%   7)J: approximation level
%   8)N: daubechies family's order
%   9)tol: tolerance bandwidth used in the semivariogram estimation.
%Output:
%   1)a0: estimated coefficients of curve at s0
%   2)mL: diagonal of matrices in the model
%   3)mcoef: coefficients matrix with mcoef(i,:) contains the parameters
%   associated with the inth curve in the functional sample.

    %Optional argument. Tolerance bandwidth used in the semivariogram
    %estimation.
    if nargin == 8
        tol = 0.1;
    end

    tic;
    %points number in the space dimension
    n = size(mdata, 2);

    %points number in the time dimension
```

```
nTime = 2^ceil(log2( size(mdata, 1) ));

%basis size
nbasis = 2^J+2*N-2;

%coefficient matrix
mcoef = NaN(n, nbasis);
for i = 1:size(mdata, 2)
    y = mdata(:,i);
    y = [y(:);zeros(nTime - size(y, 1),1)];
    mcoef(i,:) = coef_dbN(y, Iter, J, N);
end

%Distance matrix
mdist = NaN(n);
for i = 1:n
    mdist(i,:) = distance('gc', latData(i), longData(i), latData, longData);
end

%Filling the independent matrix
A = zeros((n+1) * (nbasis));
for i = 1:(n-1)
    for  j = (i+1):n
        %             disp(['i = ', num2str(i), ' j = ', num2str(j)]);
        rows = ((i-1) * nbasis + 1): (i * nbasis);
        cols = ((j-1) * nbasis + 1): (j * nbasis);
        A(rows, cols) = G(mcoef, mdist, mdist(i,j), tol) .* eye(nbasis);
        A(cols, rows) = A(rows, cols);
    end
end
cols = (nbasis * n+ 1):(nbasis * (n + 1));
for i = 1:n
rows = ((i-1) * nbasis + 1) : (i * nbasis);
A(rows,cols) = eye(nbasis);
A(cols, rows) = eye(nbasis);
end

%Filling the independent matrix
b = NaN( (n+1) * nbasis, 1);
%last elements
rows = ( n * nbasis + 1 ): ((n+1)*nbasis);
b(rows) =  ones(nbasis, 1);
%column vector of (||s0-s1||, \dots, ||s0-sn||)^\top
vdist = distance('gc', lat, long, latData, longData);
for i = 1:n
    %         disp(['i = ', num2str(i)]);
    rows = ((i-1) * nbasis + 1): (i * nbasis);
    b(rows) = diag( G(mcoef, mdist, vdist(i), tol) );
end
tempo = toc;
disp(['Filling the coefficients matrix consumed ', num2str(tempo), ' seconds.']);

%Solving the linear system
x = lsqr(A, b, 0.000001, 200000);
mL = reshape(x(1:(n*nbasis)),nbasis, n);

%Calculating a_0
a0 = zeros(nbasis, 1);
for i = 1:n
```

```
        a0 = a0 + diag( mL(:,i) ) * mcoef(i,:)';
    end
end

function [ mG ] = G( mcoef, mdist, h, tol)
%G This function computes the semivariogram for a multivariate random
%field.
%Input
%   1)mcoef: mcoef: coefficients matrix with mcoef(i,:) contains the parameters
%   associated with the inth curve in the functional sample.
%   2) mdist: distance matris with mdist(i, j) = ||s_i - s_j||
%   3)h: distance target
%   4)tol: tolerance bandwidth. Optional argument and default value is
%   0.01.
%Output:
%   1)mG: estiamted semivariogram at h

    %Optional argument. Tolerance bandwidth.
    if nargin == 3
        tol = 0.1;
    end

    %searching for points with ||s_i - s_0|| \in (h - \epsilon, h +
    %\epsilon)
    [row, col] = find( (mdist > (h - tol)) & (mdist < (h + tol)));
    %checking with index is empty...
    while numel(row) == 0
        disp(['There are no sampled points with distance ', num2str(h),...
        ' inside the tolerance ', num2str(tol)]);
        tol = tol + 0.001;
        [row, col] = find( (mdist > (h - tol)) & (mdist < (h + tol)));
    end


    %output
    mG = zeros(size(mcoef, 2));
    for i = 1:numel(row)
        mG = mG + (mcoef(row(i),:) - mcoef(col(i),:))' *...
                            (mcoef(row(i),:) - mcoef(col(i),:));
    end
    mG = mG / (2 *  numel(col));

end
```

A.5 chi_dbN

Objetivo

Esta função calcula o valor da curva estimada em um vetor coluna de pontos em $[0, 1]$ dado os coeficientes calculados pela função `coef_dbN`.

Sintaxe

```
f  = chi_dbN( t, Iter, J, N, c );
```

Descrição -

A função `chi_dbN` tem cinco argumentos de entrada

i. `t` é um vetor coluna de pontos $(t_1, \ldots, t_m)^\top$ com $t_j \in [0, 1]$

ii. `Iter` é um número inteiro positivo em $\frac{1}{2^{Iter}}$ é a precisão da aproximação numérica de integrais envolvendo a função escala

iii. `J` é o nível de aproximação

iv. `N` é um número inteiro indicando a ordem na família Daubechies

v. `c` é um vetor coluna contendo os coeficientes estimados pela função `coef_dbN`

e retorna um vetor coluna `f` $= (\chi(t_1), \ldots, \chi(t_m))^\top$ de mesma dimensão do vetor coluna `t`.

Código Fonte

Listing A.4: *Arquivo* `chi_dbN.m`

```
function [ f ] = chi_dbN( t, Iter, J, N, c )
% chi_t: Calculate \hat\phi(t) given the coefficients and t'
% Input arguments:]
%   1)t - column vector of sample values
%   2)Iter - order of dyadic points \phi at values \dfrac{l}{2^Iter}
%   3)J - approximation level
%   4)N - daubechies family's order
%   5)c - coefficients
% Output:
%   f - coefficients

    [phi, ~, xval] = wavefun(['db', num2str(N)], Iter);

    f = chi_t(t(:), xval(:), phi(:), J, N, c(:));
end
```

Observe que a função `chi_dbN` descrita em *Listing* A.4 usa o *MEX file* imprimido em *Listing* A.5. Vide o site oficial do MATLAB para informação sobre como compilar *MEX file* em cada sistema operacional.

Listing A.5: *Arquivo chi_t.c*

```
/*chi_t: Calculate \hat\phi(t) given the coefficients and t'
Input arguments:]
1)t - sample values
2)xval - dyadic points
3)phi - phi value at dyadic points
4)J - approximation level
5)N - daubechies family's order
6)c - coefficients
Output:
chi - coefficients
*/
```

```
#include <stdio.h>
#include <math.h>
#include "mex.h"
#include "matrix.h"

/*global variables*/
double* xval; /*xval have dyadic points*/
double* phi; /* phi have the value at dyadic points xval*/
int nDyadic; /*xval length*/

/*Calculate phi at t*/
double phi_dbN(double t);

void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
{
    double* t; /*column vector*/
    double J=0; /*level approximation*/
    double N=0; /* order at Daubechies family*/
    double j, i; /*loop variables*/
    double* f; /* output*/
    double* c; /*coefficients*/

    if (nrhs != 6)
    {
    mexErrMsgTxt("six inputs required.");
    }

    t = mxGetPr(prhs[0]);
    xval  = mxGetPr(prhs[1]);
    phi = mxGetPr(prhs[2]);

    /*xval with dyadic points*/
    if ( mxGetM(prhs[1]) < 2 || mxGetN(prhs[1]) != 1)
    {
    mexErrMsgTxt("Invalid xval.");
    }

    /*phi avaliated at xval*/
    if (mxGetM(prhs[2]) < 2 || mxGetN(prhs[2]) != 1)
    {
    mexErrMsgTxt("Invalid phi.");
    }

    /*xval and phi must have same size*/
    if (mxGetM(prhs[1]) != mxGetM(prhs[2]))
    {
    mexErrMsgTxt("Phi and xval must have same dimensions.");
    }

    /*t must be a column vector*/
    if (mxGetN(prhs[0]) != 1)
    {
    mexErrMsgTxt("y need be a column vector.");
    }

    /*coefficients be a column vector */
    if ( mxGetN(prhs[5]) != 1)
    {
    mexErrMsgTxt("Coefficients need be a column vector.");
```

```
    }

    c = mxGetPr(prhs[5]);

    J = mxGetScalar(prhs[3]);
    N = mxGetScalar(prhs[4]);

    /*Set the output pointer to the output matrix*/
    plhs[0] = mxCreateDoubleMatrix(mxGetM(prhs[0]), mxGetN(prhs[0]), mxREAL);

    /*get a pointer */
    f = mxGetPr(plhs[0]);

    /*number of dyadic points*/
    nDyadic = mxGetM(prhs[1]);

    for(i = 1; i <= mxGetM(prhs[0]); i++)
    {
        *(f + (int)i - 1)= 0;
        for(j = 1; j <= pow(2, J) + 2 * N - 2; j++)
        {
            if((pow(2,J) * *(t + (int)i -1) - (j-(2*N-1)) >= 0) &&
                              (pow(2,J) * *(t + (int)i -1) - (j-(2*N-1)) <= 2*N-1))
            {
                *(f + (int)i -1) = *(f + (int)i -1) +
                                   + *(c + (int)j - 1) * pow(2,J/2) * phi_dbN(pow(2, J) *
                                   * *(t +(int)i -1) - (j-(2*N-1)));
            }
        }
    }
}

double phi_dbN(double t)
{
    double cont = 0;

    while(*(xval + (int)cont) < t && cont < nDyadic)
    {
        cont = cont + 1;
    }

    return *(phi +(int)cont);
}
```

A.6 FieldHatchi_dbN

Objetivo

Esta função computa o valor da curva $\chi_{\boldsymbol{s}_0}(t)$ em uma localização não monitorada no vetor coluna de pontos `t` dado a diagonal das matrizes Λ_i e os coeficientes das curvas da amostra funcional $\chi_{\boldsymbol{s}_i}(t), i = 1, \ldots, n$.

Sintaxe

```
output_s0  = FieldHatchi_dbN( t, mLambda, mcoef, Iter, N);
```

Descrição

A função `FieldHatchi_dbN` tem cinco argumentos

i. `t` é um vetor coluna $(t_1, \ldots, t_m)^\top$ com $t_j \in [0,1], j = 1, \ldots, m$

ii. `mLambda` é uma matriz em que a i-ésima coluna é a diagonal da matriz Λ_i descrita no modelo de krigagem da Seção 3.3

iii. `mcoef` é a matriz de coeficientes estimados das curvas $\chi_{\boldsymbol{s}_i}(t), \ldots, \chi_{\boldsymbol{s}_n}(t)$ da amostra funcional

iv. `Iter` é um número inteiro positivo em que $\frac{1}{2^{Iter}}$ é a precisão usada na aproximação numérica de integrais envolvendo a função escala $\phi(x)$

e retorna um vetor coluna `output_s0` $= (\chi_{\boldsymbol{s}_0}(t_1), \ldots, \chi_{\boldsymbol{s}_0}(t_m))^\top$ de mesma dimensão do vetor `t`.

Código Fonte

Listing A.6: *Arquivo* `FieldHatchi_dbN`.

```
function [ output_s0 ] = FieldHatchi_dbN( t, mLambda, mcoef, Iter, N)
%FieldHatchi_s0 This function computes \hat\chi_{s0}(t) = (\sum_{i=1}^{n}
%\Lambda_i a_{si})^\top \bm{\phi}.
%Input:
%   t: column vector (t_1, \dots, t_m)^\top
%   mLambda: matrix where the ith column contains diag(\Lambda_i)
%   mcoef: matrix A with size(A) = [2^J + 2N -2, nSpatial] (where nSpatial
%   is the number of spatial points in the sample) with the ith column
%   contains the coefficient of \hat\chi(t)
%   Iter: order of dyadic points \phi at values \dfrac{1}{2^Iter}
%   N: order in the family Daubechies
%Output:
%   output_s0: column vector with the estimates (\hat\chi_{so}(t_1), \dots,
%   \hat\chi_{s0}(t_m))^\top

    %Approximation level
    J = log2(size(mcoef, 1) - 2 * N + 2);

    %estimated coefficient a_{s0}
    a0 = zeros(size(mcoef(:,1)));
    for i = 1:size(mcoef, 2)
        a0 = a0 + diag(mLambda(:,i)) * mcoef(:,i);
    end

    output_s0 = chi_dbN(t(:), Iter, J, N, a0);
end
```

A.7 `FunctionalKriging`

Objetivo

Esta função é a implementação do método de Krigagem Ordinária Funcional (vide Seção 3.1 para maiores detalhes) usando ondaletas Haar.

Sintaxe

```
[ lambda, mcoef, pontos, fEst] = FunctionalKriging( lat, long, latData,...
                                                longData, mdata,J, lplot, m);
```

Descrição

A função `FunctionalKriging` tem oito argumentos de entrada

i. `lat` é a latitude do ponto não monitorado $\boldsymbol{s}_0$

ii. `long` é a longitude ponto não monitorado $\boldsymbol{s}_0$

iii. `latData` é um vetor coluna contendo a latitude dos pontos $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$ de nossa amostra funcional

iv. `longData` é um vetor coluna contendo a longitude dos pontos $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$ de nossa amostra funcional

v. `mdata` é uma matriz em que a i-ésima coluna contém a série temporal observada no ponto $\boldsymbol{s}_i$ $(\chi_{\boldsymbol{s}_i}(t_1), \ldots, \chi_{\boldsymbol{s}_i}(t_m))^\top$. Se m não é potência de 2, completamos com zeros a série até que $m^\star$ seja potência de 2, isto é, a série temporal usada pelo programa é dada por $(\chi_{\boldsymbol{s}_i}(t_1), \ldots, \chi_{\boldsymbol{s}_i}(t_m), \chi_{\boldsymbol{s}_i}(t_{m+1}), \ldots$ com $m^\star = \lceil \log_2(n) \rceil$ e $\chi_{\boldsymbol{s}_i}(t_j) = 0$ para $j = m+1, \ldots, m^\star$

vi. `J` é o nível de aproximação.

vii. `lplot` é uma variável lógica. Se `true`, imprime dois gráficos: um contendo a curva estimada e um mostrando os valores estimados dos parâmetros $\lambda_1, \ldots, \lambda_n$. Esta variável é opcional e seu valor padrão é `true`.

viii. `m` é uma variável inteira positiva. Ela estabelece quantos pontos serão usado para fazer o gráfico da curva estimada, caso a variável `lplot` seja verdadeira. Esta variável é opcional e seu valor padrão é 1000.

e retorna quatro matrizes

a. `lambda` é um vetor coluna $(\lambda_1, \ldots, \lambda_n)^\top$ contendo os parâmetros estimados do modelo de Krigagem Ordinária Funcional.

b. `mcoef` é uma matriz contendo os coeficientes da curvas da amostra funcional usando ondaletas. A i-ésima linha contém os coeficientes da projeção da curva $\chi_{\boldsymbol{s}_i}(t)$ no ponto $\boldsymbol{s}_i$ no espaço de aproximação V_J.

c. `pontos` é um vetor coluna $(t_1, \ldots, t_m)^\top$ contendo as abscissas usadas para produzir o gráfico da curva estimada caso `lplot` seja `true`.

d. é um vetor coluna $(\chi_{\boldsymbol{s}_0}(t_1), \ldots, \chi_{\boldsymbol{s}_0}(t_m))^\top$ contendo as ordenadas usadas para gerar o gráfico da curva suavizada caso `lplot` tenha valor `true`.

Código Fonte

Listing A.7: *Arquivo* `FunctionalKriging.m`.

```
function [ lambda, mcoef, pontos, fEst] = FunctionalKriging( lat, long, latData,...
                                                    longData, mdata,J, lplot, m)
%FunctionalKriging This function estimates lambda_1, ..., lambda_n
%coeficients of functional kriging predictor. Moreover, it plots two
%graphs: lambda_i, i=1,...,n versus distance and estimated curve. In this
%function, we supose the curve has the interval [0,1] as the domain.
%   Input arguments
%
%   1)lat: latitude of location we wish estimate the curve
%   2)long: longitude of location we wish estimate the curve
%   3)latData: column vector with the latitude from our data
%   4)longData: column vector with the longitude from our data
%   5)J: approximaion level
%   6)mdata: matriz with the data from all points in our data
%   7)lplot: logical argument. For default, receive true. If lplot = true, we
%   plot the graphs, else we don't plot the graphs
%   8)m: size of avaluating vector points
%
%   Output arguments
%
%   1)lambda: model coefficients from functional kriging predictor
%   mcoef: cofficients of all curves in our data

    if nargin == 7
        lplot = true;
        m = 1000;
    elseif nargin == 6
        m = 1000;
    end

    nIter = length(latData);


    %Calculating distance
    distancia = NaN(nIter);
    for i = 1:nIter
        distancia(i,:) = distance('gc', latData, longData, latData(i), ...
                                    longData(i));
    end

    %coefficients matrix
    mcoef = NaN(nIter, 2^J);

    %Calculating the coefficients
    for i = 1:nIter
        mcoef(i,:) = coef(mdata(:,i),J);
    end

    tic;
    %Estimating lambda_1, ..., lambda_n
    A = NaN(nIter+1);
    b = NaN(nIter+1,1);

    %last row of A
```

```
    A(nIter+1,:) = [ones(1, nIter),0];

    %last column of A
    A(:, nIter+1) = [ones(nIter,1);0];

    %Last row of b
    b(nIter+1) = 1;

    for i = 1:nIter

        for k = (i+1):nIter
            A(i,k) = variograma(distancia(i,k), distancia, mcoef);
            A(k,i) = A(i,k);
        end

        A(i,i) = 0;


        h = distance('gc',lat, long, latData(i), longData(i));
        b(i) = variograma(h, distancia,mcoef);
    end


    %Solving the linear system
    X = linsolve(A, b);

    %Selecting only lambda
    lambda = X(1:nIter);
    tempo = toc;
    disp(['Filling the matrix demanded ', num2str(tempo), ' seconds.']);

    %Graph of resulting function
    %     m = 10000;
    pontos = linspace(0,1,m);
    fEst = NaN(size(pontos));
    for i = 1:(m-1)
        k = floor(pontos(i) * 2^J) + 1;
        fEst(i) = 2^(J/2) * mcoef(:,k)' * lambda(:);
    end
    fEst(m) = 2^(J/2) * mcoef(:,2^J)' * lambda(:);

    if lplot == true
        %lambda's graph
        figure;
        labels = sprintfc('%d', dCoorde.id);
        eixoX = distance('gc',lat, long, latData, longData);
        plot(eixoX, abs(lambda), 'k*');
        xlabel('Distance ||s_i - s_0||');
        ylabel('|\lambda_i|');
        text(eixoX, abs(lambda), labels, 'VerticalAlignment','bottom', ...
        'HorizontalAlignment','left');
        %estimated curve
        figure;
        plot(pontos, fEst, 'r-');
    end
end

function valor = variograma( h, distancia,mcoef,e )
%variograma estimate the variograma using the coefficients of chi_{s_i}(t)
```

```
%The second argument is optional and has default value 0.1

    %e argument is  setted to 0.1 by default
    if nargin == 3
        e = 0.1;
    end



    %find columns and rows with ||s_i - s_j|| \in (h-e, h+e)
    [row, col] = find((distancia < h + e) & (distancia > h - e));
    while numel(row) == 0
        disp(['Nobody with the distance ', num2str(h),...
        ' with the tolerance ', num2str(e)]);
        e = e + 0.001;
        [row, col] = find((distancia < h + e) & (distancia > h - e));
    end

    %finally estimating \hat\gamma
    valor = 0;
    for i = 1:numel(row)
        co1 = mcoef(row(i),:);
        co2 = mcoef(col(i),:);
        valor = valor + sum((co1-co2).^2);
    end
    valor = valor / (2 * numel(row));
end

function c = coef(y, J)
%coef estimate the coefficientes of \chi_{s_i}(t) using analysis of
%wavelets Haar
%J: approximation level (J < n)
%y: data

    %finding n such
    n = ceil(log2(max(size(y))));

    %Vectorizing the data
    y = y(:);

    %filling vector with zeros
    if length(y) < 2^n
        yNew = [y(:);zeros(2^n - max(size(y)),1)];
    else
        yNew = y;
    end


    %Matriz to save the estimated coefficients
    c = NaN(2^J, 1);

    %Loop esimating the coefficients following the line of Todd Ogden
    for k = 0:(2^J-1)
        upper = (k+1) * 2^n / 2^J;% - 1;
        lower = k * 2^n / 2^J + 1;
        valor = 0;
        for i = lower:upper
            valor = valor + yNew(i);
        end
```

```
        valor = valor * 2^(J/2) / 2^n;
        c(k+1) = valor;
    end
end
```

A.8 hatchi_so_haar

Objetivo

Esta função tem o objetivo de calcular a estimativa da curva $\chi_{\boldsymbol{s}_0}(t)$ em um ponto $t \in [0,1]$ usando os coeficientes $\lambda_1, \ldots, \lambda_n$ obtidos pelo médoto de interpolação espacial denominado Krigagem Ordinária Funcional e os coeficientes de aproximação das curvas $\chi_{\boldsymbol{s}_i}(t), i = 1, \ldots, n$ de nossa amostra funcional.

Sintaxe

```
fEst = hatchi_s0_haar(pontos, mcoef, lambda, J)
```

Descrição

Esta função tem quatro argumentos

i. `pontos` é um vetor coluna $(t_1, \ldots, t_m)^\top$ com $t_j \in [0,1], j = 1, \ldots, m$ em que desejamos calcular $\chi_{\boldsymbol{s}_0}(t_j), j = 1, \ldots, m$

ii. `mcoef` é uma matriz em que a i-ésima coluna contém os coeficientes de aproximação da curva $\chi_{\boldsymbol{s}_i}(t)$ de nossa amostra funcional

iii. `lambda` é um vetor coluna $(\lambda_1, \ldots, \lambda_n)^\top$ contendo os parâmetros estimados do modelo de Krigagem Ordinária Funcional

iv. `J` é o nível de aproximação

e retorna como saída um vetor coluna $(\chi_{\boldsymbol{s}_0}(t_1), \ldots, \chi_{\boldsymbol{s}_0}(t_m))^\top$ de mesma dimensão do vetor `t`.

Código Fonte

Listing A.8: *Arquivo* `hatchi_s0_haar.m`

```
function fEst = hatchi_s0_haar(pontos, mcoef, lambda, J)
%hatchi_s0_haar This function computes the estimate curve
%given the parameter from Functional Ordinary Kriging and
%coefficients matrix of functional sample
%Input:
%   1)pontos: a column vector of points to compute chi_{s_0}(t)
%   2)mcoef: a matrix where the ith column is the coefficients
%   from ith curve
%   3)lambda: column vector with the parameter of Functional Ordinary Kriging
%   4)J: approximation level
%Output:
%   1)fEst: column vector with the estimate curva at s_0 at the column vector pontos
```

```
    fEst = NaN(size(pontos));
    m = length(pontos);
    for k = 1:m
        if floor(pontos(k)*2^J) < 2^J
            pos = floor(pontos(k) * 2^J) + 1;
        else
            pos = floor(pontos(k) * 2^J);
        end
        fEst(k) = 2^(J/2) * mcoef(:,pos)' * lambda(:);
    end


end
```

A.9 PointwiseFuncKriging

Objetivo

Esta função é uma implementação do método de Krigagem Tempo-Variante Funcional (vide Seção 3.2 para maiores detalhes deste modelo) para dados funcionais usando ondaletas Haar.

Sintaxe

```
[ mb, mcoef, pontos, f] = PointwiseFuncKriging( lat, long, latData, longData,...
                                                mdata, J, lplot, m);
```

Descrição

A função `PointwiseFunctKringing` tem oito argumentos de entrada

i. `lat` é a latitude do ponto não monitorado $\boldsymbol{s}_0$

ii. `long` é a longitude ponto não monitorado $\boldsymbol{s}_0$

iii. `latData` é um vetor coluna contendo a latitude dos pontos $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$ de nossa amostra funcional

iv. `longData` é um vetor coluna contendo a longitude dos pontos $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$ de nossa amostra funcional

v. `mdata` é uma matriz em que a i-ésima coluna contém a série temporal observada no ponto $\boldsymbol{s}_i$ $(\chi_{\boldsymbol{s}_i}(t_1), \ldots, \chi_{\boldsymbol{s}_i}(t_m))^\top$. Se m não é potência de 2, completamos com zeros a série até que $m^\star$ seja potência de 2, isto é, a série temporal usada pelo programa é dada por $(\chi_{\boldsymbol{s}_i}(t_1), \ldots, \chi_{\boldsymbol{s}_i}(t_m), \chi_{\boldsymbol{s}_i}(t_{m+1})$ com $m^\star = \lceil \log_2(n) \rceil$ e $\chi_{\boldsymbol{s}_i}(t_j) = 0$ para $j = m+1, \ldots, m^\star$.

vi. `J` é o nível de aproximação.

vii. `lplot` é uma variável lógica. Se `true`, imprime dois gráficos: um contendo a curva estimada e um mostrando as funções estimadas $\lambda_1(t), \ldots, \lambda_n(t)$. Esta variável é opcional e seu valor padrão é `true`.

viii. `m` é uma variável inteira positiva. Ela estabelece quantos pontos serão usado para fazer o gráfico da curva estimada, caso a variável `lplot` seja verdadeira. Esta variável é opcional e seu valor padrão é 1000.

e retorno quatro saídas

a. `mb` é uma matriz `mb` em que a i-ésima coluna contém o parâmetro estimado $\boldsymbol{b}_i$

b. `mcoef` é uma matriz `mcoef` em que a i-ésima coluna contém os coeficientes da projeção da curva $\chi_{\boldsymbol{s}_i}(t)$ no espaço de aproximação V_J

c. `pontos` é um vetor coluna $(t_1, \ldots, t_m)^\top$ que é composto por m pontos igualmente espaçados entre 0 e 1

d. `f` é um vetor coluna $(\chi_{\boldsymbol{s}_0}(t_1), \ldots, \chi_{\boldsymbol{s}_0}(t_m))^\top$ compreendendo o valores estimados da curva $\chi_{\boldsymbol{s}_0}(t)$ no ponto não monitorado $\boldsymbol{s}_0$ para cada entrada do vetor coluna `pontos`

Listing A.9: *Arquivo* `PointwiseFuncKriging.m`.

```
function [ mb, mcoef, pontos, f] = PointwiseFuncKriging( lat, long, latData, longData,...
                                                    mdata, J, lplot, m)
%PointwiseFuncKriging This function estimates the coefficients b_0, ...,
%b_{2^J-1} coefficients of functions lambda_1, ..., lambda_n in the the
%method of pointwise functional kriging estimator. Moreover, optionally
%it plots the estimated curve and the functions lambda_1, ..., lambda_n
%   Detailed explanation goes here
%   Input arguments
%
%   lat: latitude of location we wish estimate the curve
%   long: longitude of location we wish estimate the curve
%   latData: column vector with the latitude from our data
%   longData: column vector with the longitude from our data
%   J: approximation level
%   mdata: matriz with the data from all points in our data
%   lplot: logical argument. For default, receive true. If lplot = true, we
%   plot the graphs, else we don't plot the graphs
%   Output arguments
%
%   mb: matriz with coefficients of estimated functions lambda_1, ...,
%   lambda_n
%   mcoef: cofficients of all curves in our data

    if nargin == 6
        lplot = true;
        m = 10000;
    elseif nargin == 7
        m = 10000;
    end

    nIter = max(size(latData));

    %Creating the struct dCoord
    dCoorde = struct('id',1:nIter, 'latitude', latData,...
    'longitude', longData);

    %Creating the struct dData
    dData1 = struct();
    for i = 1:nIter
        dData1.(['X',num2str(dCoorde.id(i))]) = mdata(:,i);
    end
```

```
%Calculating distance
distancia = NaN(nIter);
for i = 1:nIter
    distancia(i,:) = distance('gc', latData, longData, latData(i), ...
                              longData(i));
end

%coefficients matrix
mcoef = NaN(2^J, nIter);

%Calculating the coefficients
for i = 1:nIter
    mcoef(:,i) = coef(dData1.(['X',num2str(dCoorde.id(i))]),J);
end


%Calculating the matrices Q and J
matQ = NaN(2^J * (nIter+1));
matJ = NaN(2^J * (nIter+1), 1);

%Tolerance to estimation of the cross covariance matrix
faixa = 0.1;

%inner part of Q matrix
for k = 1:nIter
    for l = 1:nIter
        matQ(((k-1)*2^J+1):(k*2^J), ((l-1)*2^J+1):(l*2^J)) = ...
                                    Q(mcoef,distancia,k,l, faixa);
    end
end

%zero part
k = nIter+ 1;
l = nIter+ 1;
matQ(((k-1)*2^J+1):(k*2^J), ((l-1)*2^J+1):(l*2^J)) = zeros(2^J);

%last line
k = nIter+ 1;
for l = 1:nIter
    matQ(((k-1)*2^J+1):(k*2^J), ((l-1)*2^J+1):(l*2^J)) = eye(2^J);
end

%last column
l = nIter+ 1;
for k = 1:nIter
    matQ(((k-1)*2^J+1):(k*2^J), ((l-1)*2^J+1):(l*2^J)) = eye(2^J);
end

%Building the matrix J
for k = 1:nIter
    matJ(((k-1)*2^J+1):(k*2^J), 1) =...
    mJ(dCoorde,mcoef,distancia, k, lat, long, faixa);
end
%last lines
k = nIter + 1;
matJ(((k-1)*2^J+1):(k*2^J), 1) = 2^(-J / 2) * ones(2^J, 1) ;

%Solution to beta: minimize <(matJ - matQ * beta); (matJ - matQ * beta)>
beta = lsqr(matQ, matJ, 0.00001, 20000);
```

```
    %     clc;

    %Matriz de coeficientes de lambda
    mb = reshape(beta(1:(nIter * 2^J)), size(mcoef));


    %Grid value to build the graph
    pontos = linspace(0,1,m);
    f = zeros(size(pontos));
    for k = 1:m
        if floor(pontos(k)*2^J) < 2^J
            pos = floor(pontos(k) * 2^J) + 1;
        else
            pos = floor(pontos(k) * 2^J);
        end
        phi = zeros(2^J, 1);
        phi(pos) = 2^(J/2);
        f(k) = sum(diag(mcoef' * (phi * phi') * mb));
    end

    if lplot == true
        figure;
        hold on;
        plot(pontos, f, 'r-');
        title('Estimated curve.');
        xlim([0 max(pontos)]);
        xlabel('Time');
        ylabel('\chi(t)');
        hold off;

        figure;
        hold on;
        for i = 1:nIter
            f = zeros(size(pontos));
            for k = 1:m
                f(k) = funcaoEstimada(mb(:,i), pontos(k));
            end
            plot(pontos, f, 'k-');
        end
        xlabel('Time');
        ylabel('\lambda_i(t)');
        title('Estimate \lambda_i(t)');
        xlim([0 max(pontos)]);
        hold off;
    end
end

function c = coef(y, J)
%coef estimate the coefficientes of \chi_{s_i}(t) using analysis of
%wavelets Haar
%J: approximation level (J < n)
%y: data

    %finding n such
    n = ceil(log2(max(size(y))));

    %filling vector with zeros
    if max(size(y)) < 2^n
        yNew = [y(:);zeros(2^n - max(size(y)),1)];
```

```
    else
        yNew = y;
    end


    %Matriz to save the estimated coefficients
    c = NaN(2^J, 1);

    %Loop esimating the coefficients following the line of Todd Ogden
    for k = 0:(2^J-1)
        upper = (k+1) * 2^n / 2^J;% - 1;
        lower = k * 2^n / 2^J + 1;
        valor = 0;
        for i = lower:upper
            valor = valor + yNew(i);
        end
        valor = valor * 2^(J/2) / 2^n;
        c(k+1) = valor;
    end
end

function valor = EmpCrossVar(A, distancia, h, e )
%EmpCrossVar Calculate the empirical cross-covariance
%The arguments of this function are
%A: matrix of coefficients
%distancia: matrix of distance of sampled points in space
%h: distance ||s_i - s_j||
%e: tolerance

    %default value for the error
    if nargin < 4
        e = 0.01;
    end

    %Number of sampled points in the space
    n = max(size(distancia));

    %find lines and columns such that ||s_i - s_j|| \in (h-e, h+e)
    index = find((distancia < h + e) & (distancia > h - e));

    %checking with index is empty...
    while numel(index) == 0
        disp(['There are no sampled points with distance ', num2str(h),...
        ' inside the tolerance ', num2str(e)]);
        e = e + 0.001;
        index = find((distancia < h + e) & (distancia > h - e));
    end

    row = mod(index, n);
    row(row == 0) = n;
    col = ceil(index / n);

    %estimating hat gamma
    valor = zeros(max(size(A(:,1))));
    for i = 1:max(size(row))
        valor = valor + (A(:,row(i)) - A(:,col(i))) * (A(:,row(i)) - A(:,col(i)))';
    end
    valor = valor / (2 * max(size(row)));
```

```
end

function valor = Q(A, distancia, i, j, e )
%Q Calculate the auxiliaries matrices Q_{ij}
%i, j: line and column
%A: matrix of curve coefficients
%e: tolerance

    if nargin < 5
        e = 0.01;
    end


    %distance
    h = distancia(i, j);


    %Calculate the empirical cross covariance
    sigma = EmpCrossVar(A, distancia,h, e);

    %Level J
    J = log2(max(size(sigma)));

    valor = 2^J .* sigma .* eye(2^J);
end

function valor = mJ(dCoordG, A, distancia, i, l, c, e )
%mJ calculate the auxiliary matrix J

    if nargin < 7
        e = 0.01;
    end

    %Distancia
    h = distance('gc', dCoordG.latitude(i), dCoordG.longitude(i),...
    l, c);


    %Calcula a Sigma
    sigma = EmpCrossVar(A, distancia,h, e);

    %Approximation level J
    J = log2(max(size(sigma)));

    valor = 2^(J/2) * diag(sigma);
end

function valor = funcaoEstimada(c, u)
%funcaoEstimada: calculate a function at y with approximated at V_j with c
%coefficients c
%We are suposing our coefficients stay in 0, ..., 2^J - 1
%c: estimated coefficients
%u: point in [0,1]
%The returned value is \hat{f}(u)

    a = c(:);
    j = log2(max(size(a)));
    k = floor(u * 2^j);
    indicadora = zeros(1,2^j);
```

```
    if u < 1
        indicadora(k+1) = 1;
    else
        indicadora(2^j) = 1;
    end
    valor = indicadora * a * 2^(j/2);
end
```

A.10 FTVK_hatchi_s0_haar

Objetivo

Objetivo dessa função é estimar o valor da curva $\chi_{\boldsymbol{s}_0}(t)$ em um ponto não monitorado $\boldsymbol{s}_0$ para $t \in [0,1]$ usando o método Krigagem Tempo-Variante Funcional. Mais precisamente, estimamos $\chi_{\boldsymbol{s}_0}(t)$ em $t \in [0,1]$ por

$$\chi_{\boldsymbol{s}_0}(t) = \sum_{i=1}^{n} \hat{\lambda}_i(t)\hat{\chi}_{\boldsymbol{s}_i}(t)$$

em que $\hat{\lambda}_i(t) = \sum_k b_{i,k}^J \phi_{J,k}(t)$ e $\hat{\chi}_{\boldsymbol{s}_i}(t) = \sum_k a_{i,k}^J \phi_{J,k}(t)$ (para maiores detalhes vide Seção 3.2).

Sintaxe

```
fEst = FTVK_hatchi_s0_haar(pontos, mcoef, mb, J)
```

Descrição

Esta função tem quatro argumentos

i. `pontos` é um vetor coluna $(t_1, \ldots, t_m)^\top$ com $t_j \in [0,1], j = 1, \ldots, m$ em que desejamos calcular $\chi_{\boldsymbol{s}_0}(t_j), j = 1, \ldots, m$

ii. `mcoef` é uma matriz em que a i-ésima coluna contém os coeficientes de aproximação da curva $\chi_{\boldsymbol{s}_i}(t)$ de nossa amostra funcional

iii. `mb` é uma matriz em que a i-ésima coluna contém os coeficientes estimados de $\lambda_i(t)$, isto é, a i-ésima é constituda por $b_{i,-M}, \ldots, b_{i,M}$

iv. `J` é o nível de aproximação

e retorna como saída um vetor coluna $(\chi_{\boldsymbol{s}_0}(t_1), \ldots, \chi_{\boldsymbol{s}_0}(t_m))^\top$ de mesma dimensão do vetor `t`.

Código Fonte

Listing A.10: *Arquivo* `FTVK_hatchi_s0_haar.m`

```
 function fEst = FTVK_hatchi_s0_haar(pontos, mcoef, mb, J)
%hatchi_s0_haar This function computes the estimate curve
%given the parameter from Functional Time-Varying Kriging and
```

```
%coefficients matrix of functional sample
%Input:
%   1)pontos: a column vector of points to compute chi_{s_0}(t)
%   2)mcoef: a matrix where the ith column is the coefficients
%   from ith curve
%   3)mb: matrix where the ith column is the parameter vector
%   from the Function Time-Varying Kriging
%   4)J: approximation level
%Output:
%   1)fEst: column vector with the estimate curva at s_0 at the column vector pontos

    fEst = zeros(size(pontos));
    m = length(pontos);
    for k = 1:m
        if floor(pontos(k)*2^J) < 2^J
            pos = floor(pontos(k) * 2^J) + 1;
        else
            pos = floor(pontos(k) * 2^J);
        end
        phi = zeros(2^J, 1);
        phi(pos) = 2^(J/2);
        fEst(k) = sum(diag(mcoef' * (phi * phi') * mb));
    end


end
```

A.11 coef_Haar

Objetivo

Dado o vetor coluna $\mathtt{y} = (y_1, \ldots, y_n)^\top$, esta função retorna os coeficientes da aproximação $\hat{f}$ usando regressão não paramétrica e ondaletas Haar.

Sintaxe

```
c = coefHaar(y, J);
```

Descrição

A função `coefHaar` tem dois argumentos de entrada

i. `y` é um vetor coluna contendo a série temporal $y_1, \ldots, y_n$ em que desejamos suavizar usando regressão não paramétrica

ii. `J` é o nível de aproximação na análise de multirresolução

e retorna um vetor coluna `c` com $2^{\mathtt{J}}$ linhas que são os coeficientes estimados da função $\hat{f}$ obtidos usando regressão não paramétrica conforme Ogden (1997).

Código Fonte

Listing A.11: *Arquivo* `coefHaar.m`.

```
function c = coefHaar(y, J)
%coef estimate the coefficientes of \chi_{s_i}(t) using analysis of
%wavelets Haar
%J: approximation level (J < n)
%y: data

    %finding n such
    n = ceil(log2(length(y)));

    %Vectorizing the data
    y = y(:);

    %filling vector with zeros
    if length(y) < 2^n
        yNew = [y(:);zeros(2^n - max(size(y)),1)];
    else
        yNew = y;
    end


    %Matriz to save the estimated coefficients
    c = NaN(2^J, 1);

    %Loop esimating the coefficients following the line of Todd Ogden
    for k = 0:(2^J-1)
        upper = (k+1) * 2^n / 2^J;% - 1;
        lower = k * 2^n / 2^J + 1;
        valor = 0;
        for i = lower:upper
            valor = valor + yNew(i);
        end
        valor = valor * 2^(J/2) / 2^n;
        c(k+1) = valor;
    end
end
```

A.12 `coef_dbN`

Objetivo

Esta função calcula os coeficientes da função $\hat{f}(x) = \sum_k c_k \phi_{J,k}(t)$ obtida pela regressão paramétrica usando ondaletas Daubechies dada por

$$y_i = f\left(\frac{i}{n}\right) + \epsilon_i, \quad i = 1, \ldots, n.$$

Sintaxe

```
c  = coef_dbN( y, Iter, J, N )
```

Descrição

A função `coef_dbN` tem quatro argumentos de entrada

i. `y` é um vetor coluna composta da série temporal $y_1, \ldots, y_n$ que desejamos suavizar

ii. `Iter` é um número inteiro positivo inteiro em que $\frac{1}{2^{Iter}}$ é a precisão usada nas integrações numéricas

iii. `J` é o nível de aproximação

iv. `N` é um número inteiro positivo indexando a família Daubechies

e retorna um vetor coluna $\mathtt{c} = (c_{-2N+2}, \ldots, c_{2^J-1})^\top$ com $2^\mathtt{J} + 2N - 2$ elementos tal que $\hat{f}(x) = \sum_{k=-2N+2}^{2^J-1} c_k 2^{\frac{J}{2}} \phi(2^J x - k)$ em que $\phi(\cdot)$ é função escala para ondaleta Daubechies de ordem N.

Código Fonte

Listing A.12: *Arquivo* `coef_dbN`

```
 function [ c ] = coef_dbN( y, Iter, J, N )
% cef_dbN: Calculate coefficients Daubechies in a nonparametrical regression
% y = f(x) + epsilon.
% Input arguments:
%   1)y - sample values
%   2)Iter - order of dyadic points \phi at values \dfrac{l}{2^Iter}
%   3)J - approximation level
%   4)N - daubechies family's order
% Output:
%   c - coefficients


    [phi, ~, xval] = wavefun(['db',num2str(N)], Iter);
    c = coef_dbN_mx(y(:), xval(:), phi(:), J, N);
end
```

Note que a função `coef_dbN` cujo código está descrito em *Listing* A.12 usa o *MEX file* descrito em *Listing* A.13.

Listing A.13: *Arquivo* `coef_dbN_mx.c`.

```
 /*cef_dbN: Calculate coefficients Daubechies in a non parametrics regression
y = f(x) + epsilon.
Input arguments:]
    1)y - sample values
    2)xval - dyadic points
    3)phi - phi value at dyadic points
    4)J - approximation level
    5)N - daubechies family's order
Output:
    c - coefficients
*/

#include <stdio.h>
#include <math.h>
#include "mex.h"
#include "matrix.h"

/*global variables*/
```

```
double* xval; /*xval have dyadic points*/
double* phi; /* phi have the value at dyadic points xval*/
int nDyadic; /*xval length*/

/*integration function*/
double integration(double a, double b);

/*log2 function*/
double Log2(double n) ;

/*maximum value between two numbers*/
double max2(double a, double b);
/*double max(double first, double second);*/

/*minimum value between two numbers*/
double min2(double a, double b);
/*double min(double first, double secont);*/

void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
 {
     double* y; /*response vector*/
     double J; /*level approximation*/
     double N; /* order at Daubechies family*/
     double k, i; /*loop variables*/
     double* z; /*auxiliary matrix*/
     double p2J;
     double lower, upper, l, u;
     double n; /*number of samples points*/

     /*We neew 5 inputs*/
    if ( nrhs != 5)
    {
        mexErrMsgTxt("Five inputs required.");
    }

    y = mxGetPr(prhs[0]);
    xval  = mxGetPr(prhs[1]);
    phi = mxGetPr(prhs[2]);


    /*we need, at least, a column vector with two elements*/
    if ( mxGetM(prhs[1]) < 2 || mxGetN(prhs[1]) != 1)
    {
        mexErrMsgTxt("Invalid xval.");
    }

    if (mxGetM(prhs[2]) < 2 || mxGetN(prhs[2]) != 1)
    {
        mexErrMsgTxt("Invalid phi.");
    }

    if (mxGetM(prhs[1]) != mxGetM(prhs[2]))
    {
        mexErrMsgTxt("Phi and xval must have same dimensions.");
    }

    if (mxGetN(prhs[0]) != 1)
    {
        mexErrMsgTxt("y need be a column vector.");
```

```
    }

    J = mxGetScalar(prhs[3]);
    N = mxGetScalar(prhs[4]);

    /*number of dyadic points*/
    nDyadic = mxGetM(prhs[2]);

    /*number of sampled points*/
    n = mxGetM(prhs[0]);

    /*2^J*/
    p2J =  pow(2, J) ;

    /*Set the output pointer to the output matrix*/
    plhs[0] = mxCreateDoubleMatrix((int)pow(2, J) + 2 * N - 2, 1, mxREAL);

    /*get a pointer */
    z = mxGetPr(plhs[0]);


    for(k = (-2*N+2); k <= (p2J - 1); k++)
    {
        lower = max2( (k * n) / p2J + 1, 1);
        upper = min2( ((k + 2 * N - 1) * n) / p2J, n);
        *(z+(int)k+2*(int)N-2) = 0;
        for(i = lower; i <= upper; i++)
        {
            l =  (i-1) * p2J /  n - k;
            u = i * p2J / n - k;
            *(z+(int)k+2*(int)N-2) = *(z+(int)k+2*(int)N-2) +
                      +*(y + (int)i - 1) * integration(l, u) * pow(2, (-J/2)) ;
        }
    }


 }

 double integration(double a, double b)
{
    int lower=0, upper=0; /*integration limits*/
    int i; /*loop variable*/
    double res = 0; /*integration result*/

    /*finding lower*/
    for( i = 0; i < nDyadic; i++)
    {
        if(*(xval + i) == a)
        {
            lower = i;
            break;
        }
    }

    /*finding upper*/
    for( i = lower; i < nDyadic; i++)
    {
        if(*(xval + i) == b)
        {
```

```
            upper = i;
            break;
        }
    }


    for(i = lower; i <= upper; i++)
    {
        res = res + *(phi + i);
    }

    res = res / pow(2,  Log2( 1 / (*(xval + 1) - *(xval + 0)) )  ) ;
    return res;
}

double Log2( double n )
{
    /* log(n)/log(2) is log2.  */
    return log( n ) / log( 2 );
}

double max2(double a, double b)
{
    if (a >= b)
    {
        return a;
    }
    else
    {
        return b;
    }
}

double min2(double a, double b)
{
    if (a <= b)
    {
        return a;
    }
    else
    {
        return b;
    }
}
```

A.13 FuncKriging_dbN

Objetivo

Esta função implmenenta o método de Krigagem Tempo-Variante Funcional usando ondaletas da família Daubechies.

Sintaxe

```
[ lambda, mcoef] = FuncKriging_dbN( lat, long, latData, longData, mdata,...
                    J, N, Iter, lplot);
```

Descrição

Esta função tem nove argumentos de entrada

i. `lat` é a latitude do ponto não monitorado $\boldsymbol{s}_0$

ii. `long` é a longitude do ponto não monitorado $\boldsymbol{s}_0$

iii. `latData` é um vetor coluna contendo as latitudes dos pontos $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$ das curvas em nossa amostra funcional

iv. `longData` é um vetor coluna contendo as longitudes dos pontos $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$ das curvas em nossa amostra funcional

v. `mdata` é um matriz em que a i-ésima coluna é a série temporal $\chi_{\boldsymbol{s}_i}(t_1), \ldots, \chi_{\boldsymbol{s}_i}(t_n)$ observada da curva $\chi_{\boldsymbol{s}_i}(t)$

vi. `J` é o nível de aproximação na análise de multirresolução

vii. `N` é um número inteiro positivo que indexa a ordem na família Daubechies

viii. `Iter` é uma número inteiro positivo tal que $\dfrac{1}{2^{Iter}}$ precisão numérica de integrais envolvendo a função escala $\phi(x)$. Este argumento é opcional e seu valor padrão é 10.

ix. `lplot` uma variável lógica. Se `true`, a função produz um gráfico com a curva estimada. Esta argumento é opcional e seu valor padrão é `true`.

e retorna duas saídas

a. `lambda` é um vetor coluna composto dos parâmetros estimados $\lambda_1, \ldots, \lambda_n$ no modelo de Krigagem Ordinária Funcional

b. `mcoef` é uma matriz em que a i-ésima coluna contém os coeficientes estimados da curva $\chi_{\boldsymbol{s}_i}(t)$

Código Fonte

Listing A.14: *Arquivo* `FuncKriging_dbn.m`.

```
function [ lambda, mcoef] = FuncKriging_dbN( lat, long, latData, longData,...
    mdata, J, N, Iter, lplot)
%FuncKriging_dbN Calculate coefficients for Daubechies family and the
%parameters in the Functional Kriging methodology
%Input:
%   lat: latitude of location we wish estimate the curve
%   long: longitude of location we wish estimate the curve
%   latData: column vector with the latitude from our data
%   longData: column vector with the longitude from our data
%   J: approximation level
%   N: order in the family Daubechies
%   Iter: order of dyadic points \phi at values \dfrac{1}{2^Iter}
%   mdata: matrix with the data from all points in our data (data by station is on column)
%   lplot: logical argument. For default, receive true. If lplot = true, we
%   plot the graphs, else we don't plot the graphs
%Output arguments
```

```
%   mb: matriz with coefficients of estimated functions lambda_1, ...,
%   lambda_n
%   mcoef: cofficients of all curves in our data

    if nargin == 8
        lplot = true;
    elseif nargin == 7
        lplot = true;
        Iter = 10;
    elseif nargin < 7
        error('Insuficient number of inputs. Vide help for more information.');
    end

    %approximation of phi, xval e psi
    [phi, ~, xval] = wavefun(['db', num2str(N)], Iter);

    %number of sampled spatial points
    sp = size(latData, 1);

    mdist = NaN(sp);%matriz of distance between two spatial sampled point
    vdist = NaN(sp, 1);%column vector betw spatial sampled point and non
                       %sampled point

    for i = 1:sp
        mdist(:,i) = distance('gc', latData(i), longData(i), latData, longData);
        vdist(i) = distance('gc', lat, long, latData(i), longData(i));
    end

    %length of time series
    nTempo = size(mdata, 1);

    %next near number potency of 2
    n = 2^ceil(log2(nTempo));

    %coefficients matrix
    mcoef = NaN(2^J+2*N-2, sp);
    for i = 1:sp
        y = mdata(:,i);
        y = [y(:);zeros(n - size(y, 1),1)];
        mcoef(:,i) = coef_dbN_mx(y(:), xval(:), phi(:), J, N);
    end

    x = lambdaC_dbN_mex(mcoef, mdist, vdist, J, N);
    lambda = x(1:sp);

    if lplot == true
        figure;
        m = 1000;
        t = linspace(0, 1, m)';
        fp = hatchi_t(t(:), xval(:), phi(:), J, N, mcoef, lambda(:));
        plot(t, fp, 'r-');
    end

end
```

Note que a função `FuncKriging_dbN`, cujo código está descrito em *Listing* A.14, usa a função `lambdaC_dbN_mex` que calcula os parâmetros $\lambda_1, \ldots, \lambda_n$ no modelo de Krigagem Ordinária Funcional e foi implementa usando *MEX file* cujo código fonte está transcrito em *Listing* A.15. Ademais, para produzir o gráfico da curva ajustada usamos a função `hatchi_t_mx`

que foi programada usando *MEX file* e seu código fonte está impresso no *Listing* A.17.

Listing A.15: *Arquivo* `lambdaC_dbN_mex.c`.

```
 /*
lambdaC: calculate \lambda_1, \dots, \lambda_n using the functional kriging predictor
Inputs:
   1)mcoef: matrix with rows representing coefficients of sampled spots
   2)mdist: matrix A = (a_{ij}) with a_{ij} = distance(s_i, s_j) where
   s_i, s_j are sampled spots
   3)vdist: column vector V = (v_1, \dots, v_n)^T with v_j =
   distance(s_0, s_j) where s_j is a sampled spot
   4)J: approximation level
   5)N: order in the daubechies family
 output:
    1)lambda: estimated coefficients of functional kriging predictor
*/

#include <stdio.h>
#include <math.h>
#include "mex.h"
#include "matrix.h"
#include "blas.h"
#include "lapack.h"
#include <stdlib.h>

//global variables
double J; //approximation level
double N; //order in the daubechies family
double* mdist; //matrix A = (a_{ij}) with a_{ij} = distance(s_i, s_j)
            //where s_i, s_j are sampled spots
int r_mdist; //number of rows and columns of distance matrix.
              //This is a square matrix
double* mcoef; //matrix with estimated coefficients's curves at sampled spot
int r_mcoef, c_mcoef; //number of rows and columns of mcoef matrix
double* vdist; //column vector V = (v_1, \dots, v_n)^T with v_j =
            //distance(s_0, s_j) where s_j is a sampled spot
int n_vdist; //vdist length

//function which calculate the estimated variogram
double variogram(double h,  double e);

void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
{
    double* A;
    double* A1;
    double* b;
    int i,j;
    int dimA;
    //inputs for dgesv
    size_t m,n,p;      /* matrix dimensions */
    mwSignedIndex *iPivot;   /* inputs to DGESV */
    mxArray *Awork, *mxPivot;
    mwSignedIndex info, dims[2];
    //we need 5 inputs arguments
    if (nrhs != 5)
    {
        mexErrMsgTxt("Five inputs required.");
    }
```

```
mcoef = mxGetPr(prhs[0]);
mdist = mxGetPr(prhs[1]);
vdist = mxGetPr(prhs[2]);

r_mcoef = mxGetM(prhs[0]);
c_mcoef = mxGetN(prhs[0]);
r_mdist = mxGetM(prhs[1]);
n_vdist = mxGetM(prhs[2]);

J = mxGetScalar(prhs[3]);
N = mxGetScalar(prhs[4]);

//Number of columns of mcoef must be equal number of rowns and columns of mdist
if (r_mdist != c_mcoef)
{
    mexErrMsgTxt("Matrices for coefficients and distances have wrong dimensions.\n")
}

//Checking number of mcoef rows
if (r_mcoef != (int)pow(2,J) + 2 * (int)N - 2)
{
    mexErrMsgTxt("Number of mcoef rows must be 2^J + 2 * N -2.\n");
}

//mdist must sqaure matrix
if (mxGetM(prhs[1]) != mxGetN(prhs[1]))
{
    mexErrMsgTxt("Square matrix required for distances between sampled spots.\n");
}


//vdist have to be a column vector
if (mxGetN(prhs[2]) != 1)
{
    mexErrMsgTxt("Column vector distance required.\n");
}

//checking if we have sufficient elements at vdist
if (n_vdist != r_mdist)
{
    mexErrMsgTxt("Column vector of distances disagree with matrix distance.\n");
}

//Creating matrix A and b

Awork = mxCreateDoubleMatrix(r_mdist+1, r_mdist+1, mxREAL);
plhs[0] = mxCreateDoubleMatrix(r_mdist+1, 1, mxREAL);
A = mxGetPr(Awork);
b = mxGetPr(plhs[0]);
dimA = r_mdist+1;

//filling the last column of A
for (i = 0;  i < (dimA - 1); i++)
{
    A[(dimA - 1)*dimA + i] = 1;
}
A[(dimA-1)*dimA + (dimA-1)] = 0;
```

```
    //filling the last row of A
    for (i =0; i < (dimA-1);i++)
    {
        A[i * dimA + dimA-1] = 1;
    }

    //fillings last row of b
    b[dimA-1] = 1;

    //full filling A

    for(i = 0; i < (dimA-2); i++)
    {
        for(j = i; j < (dimA-1); j++)
        {
            A[i + j * dimA] = variogram(mdist[i + j * r_mdist], 0.1);
            A[j + i * dimA] = A[i + j * dimA];
        }
    }
    for (i = 0; i < dimA; i++)
    {
        A[i + i * dimA] = 0;
    }

    //full filling b
    for (i = 0; i < (dimA-1); i++)
    {
        b[i] = variogram(vdist[i],0.1);
    }

    //

    //Solving the linear system
    m = mxGetN(Awork);
    n = mxGetN(plhs[0]);
    p = mxGetM(plhs[0]);

    dims[0] = m;
    dims[1] = p;
    mxPivot = mxCreateNumericArray(2, dims, mxINT32_CLASS, mxREAL);
    iPivot = (mwSignedIndex*)mxGetData(mxPivot);

    /* Call LAPACK */
    dgesv(&m,&n,A,&m,iPivot,b,&p,&info);

    //Error if info != 0
    if (info != 0)
    {
        mexErrMsgTxt("Houston, we have a problem.\n");
    }

}

double variogram(double h,  double e)
{
    int cont;
    int i, j, k;
    double res;
    double tol;
```

```
    cont = 0;
    tol = e;
    while (cont == 0)
    {
        res = 0;
        for(i = 0; i < r_mdist; i++)
        {
            for(j = 0; j < r_mdist; j++)
            {
                if ((mdist[i*r_mdist+j] < h + tol) && (mdist[i*r_mdist+j] > h - tol))
                {
                    for(k = 0; k < r_mcoef; k++)
                    {
                        res = res + (mcoef[i*r_mcoef+k] - mcoef[j*r_mcoef+k]) *
                                  *(mcoef[i*r_mcoef+k] - mcoef[j*r_mcoef+k]);
                    }
                    cont = cont + 1;
                }
            }
        }

        //If we don't catch nobody
        if (cont == 0)
        {
            printf("Nobody at neighbour.\n");
            tol = tol + 0.001;
        }
        else //If we catch somebody
        {
            res = res / (2 * cont);
        }
    }
    return res;
}
```

A.14 `hatchi_s0`

Objetivo

Esta função calcula o valor da curva $\chi_{\boldsymbol{s}_0}(t)$ no ponto t dado os parâmetros $\lambda_1, \ldots, \lambda_n$ no modelo de Krigagem Ordinária Funcinal.

Sintaxe

```
f  = hatchi_s0( t, Iter, J, N, mcoef, lambda );
```

Descrição

A função `hatchi_s0` tem seis argumentos de entrada

i. `t` é um vetor coluna $(t_1, \ldots, t_m)^\top$ que desejamos calcular $\chi_{\boldsymbol{s}_0}(t_j), j = 1, \ldots, m$

ii. `Iter` é um número inteiro positivo tal que $\frac{1}{2^{Iter}}$ é a precisão utilizada nos métodos de aproximação para integrais envolvendo a função escala $\phi(x)$

iii. `J` é o nível de aproximação

iv. `N` é um número inteiro positivo indexando a ordem na família Daubechies

v. `mcoef` é uma matriz cuja i-ésima coluna é composta pelos coeficientes da aproximação da curva $\chi_{\boldsymbol{s}_i}(t)$

vi. `lambda` é um vetor coluna constituido pelos parâmetros $\lambda_1, \ldots, \lambda_n$ no modelo de Krigagem Ordinária Funcional

e retorna um vetor coluna `f` $= (\chi_{\boldsymbol{s}_0}(t_1), \ldots, \chi_{\boldsymbol{s}_0}(t_m))^\top$ com o mesmo número de colunas de `t`.

Código Fonte

Listing A.16: *Arquivo* `hatchi_s0.m`.

```
 function [ f ] = hatchi_s0( t, Iter, J, N, mcoef, lambda )
% hatchi_s0: calculate \hat{\chi}_{s_0}(t) given mcoef e \lambda_1, \dots, \lambda_{sp} of
% Inputs:
%    1)t: column vector of values in [0,1]
%    2)Iter: order of dyadic points \phi at values \dfrac{l}{2^Iter}
%    3)J: approximation level
%    4)N: order in the daubechies family
%    5)mcoef:matrix with columns representing coefficients of sampled spots
%    6)lambda: coefficients estimated of functional kriging predictor
%  output:
%   1)chi_{s_0}: estimated chi_{s_0} evaluated at t

    [phi, psi, xval] = wavefun(['db',num2str(N)], Iter);

    f = hatchi_t_mx(t(:), xval(:), phi(:), J, N, mcoef, lambda);
end
```

Note que a função `hatchi_s0` usa a função `hatchi_t_mx` que foi implementada usando *MEX file* cujo código `hatchi_t_mx.c` fonte é transcrito em *Listing* A.17.

Listing A.17: *Arquivo* `hatchi_t_mx.c`.

```
 /*
hatchi_t: calculate \hat{\chi}_{s_0}(t) given mcoef e \lambda_1, \dots, \lambda_{sp}
of functional kriging predictor
Inputs:
   1)t: column vector of values in [0,1]
   2)xval: column vector with dyadic points
   3)phi: \phi evaluated at xval
   4)J: approximation level
   5)N: order in the daubechies family
   6)mcoef:matrix with columns representing coefficients of sampled spots
   7)lambda: coefficients estimated of functional kriging predictor
 output:
    1)chi_{s_0}: estimated chi_{s_0} evaluated at t
*/

#include <stdio.h>
#include <math.h>
#include "mex.h"
```

```
#include "matrix.h"
#include <stdlib.h>

double* xval;
double* phi;
double J;
double N;
double* mcoef;
int nDyadic;

//estimated curve
double chi_t (double x, int col);

//scale function
double phi_dbN(double t);

void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
{
    int sp; //number of sampled points
    double* t; //column vector of points in [0,1]
    double* lambda; //estimated coefficients of functional kriging predictor
    double* f; //estimated \hat{\chi}_{s_0}
    int nt; //number of elements of t
    int i, j, k; //loop variable

    //checking the inputs number
    if ( nrhs != 7)
    {
        mexErrMsgTxt("Seven inputs required.\n");
    }

    t = mxGetPr(prhs[0]);
    nt = mxGetM(prhs[0]);


    if (mxGetN(prhs[0]) != 1)
    {
        mexErrMsgTxt("First argument must be a column vector.\n");
    }

    xval = mxGetPr(prhs[1]);
    phi = mxGetPr(prhs[2]);
    nDyadic = mxGetM(prhs[1]);

    if (mxGetM(prhs[1]) != mxGetM(prhs[2]) ||
                mxGetN(prhs[1]) != 1 || mxGetN(prhs[2]) != 1)
    {
        mexErrMsgTxt("Second e third arguments have to be column
                      vectors with equal dimensions.\n");
    }

    J = mxGetScalar(prhs[3]);
    N = mxGetScalar(prhs[4]);

    //coefficients matrix
    mcoef = mxGetPr(prhs[5]);
    sp = mxGetN(prhs[5]);
    if (mxGetM(prhs[5]) != (int)pow(2, J) + 2 * (int)N -2)
    {
```

```
        mexErrMsgTxt("Fifth arguments must be a columns with
                        2^J - 2 * n - 2 columns.\n");
    }

    //column vector of estimated lambda
    lambda = mxGetPr(prhs[6]);

    if (mxGetN(prhs[6]) != 1)
    {
        mexErrMsgTxt("Sixth argument must be a column vector.\n");
    }

    if (mxGetM(prhs[6]) != mxGetN(prhs[5]))
    {
        mexErrMsgTxt("Dimensions of fifth and sixth argument don't agree.\n");
    }

    plhs[0] = mxCreateDoubleMatrix(nt, 1, mxREAL);
    f = mxGetPr(plhs[0]);

    //*f = 0;
    //f[1] = 1;


    for (i = 0; i < nt; i++)
    {
        f[i] = 0;
        for (j = 0; j < sp; j++)
        {
            f[i] = f[i] + lambda[j] * chi_t(t[i], j);
        }
    }
}

//estimated chi_t
double chi_t (double x, int col)
{
    double  j; //loop variable
    double res = 0;
    int linhas; // row numbers of mcoef

    linhas = (int)pow(2,J) + 2 * (int)N  - 2;
    for(j = 1; j <= pow(2, J) + 2 * N - 2; j++)
    {
        if((pow(2,J) * x - (j-(2*N-1)) >= 0) &&
                (pow(2,J) * x - (j-(2*N-1)) <= 2*N-1))
        {
            res = res + mcoef[col * linhas + (int)j - 1] *
                    *pow(2,J/2) * phi_dbN(pow(2, J) * x - (j-(2*N-1)));
        }
    }
    return res;
}

// scale function
 double phi_dbN(double t)
 {
     int cont = 0;
```

```
    while(xval[cont] < t && cont < nDyadic)
    {
        cont = cont + 1;
    }

    return phi[cont];
}
```

A.15 PointwiseFuncKriging_dbN

Objetivo

Dada uma amostra funcional $\chi_{\boldsymbol{s}_1}(t), \ldots, \chi_{\boldsymbol{s}_n}(t)$, esta função estima uma curva $\chi_{\boldsymbol{s}_0}(t)$ em um ponto $\boldsymbol{s}_0 \notin \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n\}$ usando o modelo de Krigagem Tempo-Variante Funcional conforme descrito na Seção 3.2 para ondaletas da família Daubechies.

Sintaxe

```
[ mb, mcoef,pontos, fp ] = PointwiseFuncKriging_dbN( lat, long, latData, longData,...
                                    mdata, J, N, Iter, lplot, m);
```

Descrição

A função `PointwiseFuncKriging_dbN` tem dez argumentos de entrada

i. `lat` é a latitude do ponto não monitorado $\boldsymbol{s}_0$

ii. `long` é a longitude do ponto não monitorado $\boldsymbol{s}_0$

iii. `latData` é um vetor coluna contendo as latitudes dos pontos $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$ das curvas em nossa amostra funcional

iv. `longData` é um vetor coluna contendo as longitudes dos pontos $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$ das curvas em nossa amostra funcional

v. `mdata` é um matriz em que a i-ésima coluna é a série temporal $\chi_{\boldsymbol{s}_i}(t_1), \ldots, \chi_{\boldsymbol{s}_i}(t_n)$ observada da curva $\chi_{\boldsymbol{s}_i}(t)$

vi. `J` é o nível de aproximação na análise de multirresolução

vii. `N` é um número inteiro positivo que indexa a ordem na família Daubechies

viii. `Iter` é uma número inteiro positivo tal que $\frac{1}{2^{Iter}}$ precisão numéricas para aproximação de integrais envolvendo a função escala $\phi(x)$. Este argumento é opcional e seu valor padrão é 10.

ix. `lplot` é uma variável lógica. Se `true`, a função produz um gráfico com a curva estimada. Esta argumento é opcional e seu valor padrão é `true`.

x. `m` é um número inteiro positivo usado para determinar o número de pontos usados nas abscissas para construir o gráfico da curva estimada caso `lplot` seja `true`

e retorna quatro saídas

a. `mb` é uma matriz cuja i-ésima coluna contém o vetor de parâmetros $\boldsymbol{b}_i$ do modelo de Krigagem Tempo-Variante Funcional descrito na Seção 3.2

b. `mcoef` é uma matriz cuja i-ésima coluna contém os coeficientes da curva $\chi_{\boldsymbol{s}_i}(t)$

c. `pontos` é um vetor coluna $(t_1, \ldots, t_m)^\top$ com `m` pontos entre 0 e 1

d. `fp` é vetor coluna $(\chi_{\boldsymbol{s}_0}(t_1), \ldots, \chi_{\boldsymbol{s}_0}(t))^\top$ com o mesmo número de linhas de `t` contendo as estimativas da curva $\chi_{\boldsymbol{s}_0}(t)$ no ponto não monitorado $\boldsymbol{s}_0$ pelo método de Krigagem Tempo-Variante Funcional

Código Fonte

Listing A.18: *Arquivo* `PointwiseFuncKriging_dbN`.

```
function [ mb, mcoef,pontos, fp ] = PointwiseFuncKriging_dbN( lat, long,...
    latData, longData, mdata, J, N, Iter, lplot, m)
%FuncKriging_dbN Calculate coefficients for Daubechies family and the
%parameters in the Pointwise Functional Kriging methodology
%Input:
%   lat: latitude of location we wish estimate the curve
%   long: longitude of location we wish estimate the curve
%   latData: column vector with the latitude from our data
%   longData: column vector with the longitude from our data
%   J: approximation level
%   N: order in the family Daubechies
%   Iter: order of dyadic points \phi at values \dfrac{l}{2^Iter}
%   mdata: matriz with the data from all points in our data
%   lplot: logical argument. For default, receive true. If lplot = true, we
%   plot the graphs, else we don't plot the graphs
%Output arguments
%   mb: matriz with coefficients of estimated functions lambda_1, ...,
%   lambda_n
%   mcoef: cofficients of all curves in our data

    if nargin == 9
        m = 1000;
    elseif nargin == 8
        m = 1000;
        lplot = true;
    elseif nargin == 7
        m = 1000;
        lplot = true;
        Iter = 10;
    elseif nargin < 7
        error('Insuficient number of inputs. Vide help for more information.');
    end

    %approximation of phi, xval e psi
    [phi, ~, xval] = wavefun(['db', num2str(N)], Iter);

    %number of sampled spatial points
    sp = size(latData, 1);

    mdist = NaN(sp);%matriz of distance between two spatial sampled point
    vdist = NaN(sp, 1);%column vector betw spatial sampled point and non
```

```
                        %sampled point

    for i = 1:sp
        mdist(:,i) = distance('gc', latData(i), longData(i), latData, longData);
        vdist(i) = distance('gc', lat, long, latData(i), longData(i));
    end

    %length of time series
    nTempo = size(mdata, 1);

    %next near number potency of 2
    n = 2^ceil(log2(nTempo));

    %coefficients matrix
    mcoef = NaN(2^J+2*N-2, sp);
    for i = 1:sp
        y = mdata(:,i);
        y = [y(:);zeros(n - size(y, 1),1)];
        mcoef(:,i) = coef_dbN_mx(y(:), xval(:), phi(:), J, N);
    end

    [mmQ, mmJ] = mQ_mx(mcoef, mdist, xval', phi',N, J, vdist);

    ntam = 2^J + 2*N - 2;

    beta1 = lsqr(mmQ, mmJ, 0.001, 2000);

    mb = reshape(beta1(1:(sp * ntam)), [ntam,sp]);
    %m = 1000;
    pontos = linspace(0,1,m)';
    fp = zeros(size(pontos));

    for i = 1:sp
        fp = fp + chi_t(pontos,xval',phi',J, N, mb(:,i)) .*...
         chi_t(pontos,xval',phi',J, N, mcoef(:,i));
    end

    if lplot == true
        figure;
        plot(pontos, fp,'r-');
    end
end
```

A função `PointwiseFuncKriging_dbN` usa a função `mQ_mx` que foi implementada usando *MEX file* cujo código fonte é transcrito a seguir.

Listing A.19: *Arquivo mQ_mx.c.*

```
 /*mQ_mex: Calculate matrix of coefficients in pointwise functional kriging predictor
    1)mcoef - coefficients matrix
    2)mdist - distance matrix
    3)xval - column vector of dyadic points
    4)phi  - scale function at dyadic points
    5)N - order at daubechies family
    6)vdist - distance between sampled points and estimating point
Output:
    output - coefficient matrix
    matB - independent matrix
*/
```

```
#include <stdio.h>
#include <math.h>
#include "mex.h"
#include "matrix.h"

/*global variables*/
double* mcoef; /* coefficients matrix */
double* mdist; /*distance matrix*/
double* vdist; /*distance matrix (between sampled points and estimating point)*/
double* phi; /* value of phi at dyadic points*/
double* xval; /*dyadic points*/
int nDyadic; /*number of dyadic points*/
int N; /*order at Daubechies family*/
int J; /* approximation level*/
int sp; /*number of spatial sampled points*/

/*Calculate cross covariance*/
void EmpCross(double h, double* res);

/*Integrate \phi_{pklq}*/
double IntPhi_pklq(int p, int k, int l, int q);

/*auxiliary function to find c_k*/
double integration(int i);

/*Integrate \phi_{jkl}*/
double IntPhi_jkl(int j, int k, int l);

/*daubechies scale function*/
double dbN(double x);

/*phi_{pklq} function*/
double dbN_pklq(double y, int p, int k, int l, int q);

/*phi_{jkl} function*/
double dbN_jkl(double x, int j, int k, int l);

/*maximum of five integers*/
int maximo(int i1, int i2, int i3, int i4, int i5);

/*maximum of three integers*/
int max3(int i1, int i2, int i3);

/*minimum of three integers*/
int min3(int i1, int i2, int i3);

void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
{
    double* output;/*output variable*/
    double* matB; /* independent matrix*/
    int i, j, row, col, ntam, ndim; /*loop variable*/
    int p, k, l, q; /*test variable*/
    int p2J;
    mxArray* Awork;
    double* mVar;
    int ak, bk, al, bl, aq, bq;

    /*four inputs required*/
```

```
if (nrhs != 7)
{
    mexErrMsgTxt("Five inputs required.\n");
}

/*mdist is a square matrix*/
if (mxGetM(prhs[1]) != mxGetN(prhs[1]))
{
    mexErrMsgTxt("Square matrix squared.\n");
}

/*mcoef must agree with mdist*/
if (mxGetN(prhs[0]) != mxGetM(prhs[1]))
{
    mexErrMsgTxt("Columns number of coefficients matrix have to has
                the same row number of distance matrix.\n");
}

/*vector of dyadic points must be a column vector*/
if (mxGetN(prhs[2]) != 1)
{
    mexErrMsgTxt("Vector of dyadic points has to be a column vector.\n");
}

/*vector of phi must be a column vector*/
if (mxGetN(prhs[3]) != 1)
{
    mexErrMsgTxt("Vector of scale function evaluated at dyadic
                points has to be a column vector.\n");
}

N = mxGetScalar(prhs[4]);
J = mxGetScalar(prhs[5]);

/*checking tbe dimension of coefficients matrix*/
if (mxGetM(prhs[0]) != (int)pow(2, (double)J) +2*N - 2)
{
    printf("teste = %d J = %d.\n", (int)pow(2, (double)J) +2*N - 2, J);
    mexErrMsgTxt("Coefficient matrix has to have 2^J + 2*N - 2 rows.\n");
}

/*xval and phi have to has the same length*/
if (mxGetM(prhs[2]) != mxGetM(prhs[3]))
{
    mexErrMsgTxt("xval and phi must have the same length.\n");
}

/*column vector of distance between sampled points and estimating point*/
if (mxGetN(prhs[6]) != 1)
{
    mexErrMsgTxt("Sixth argument have to be a column vector of distance.\n");
}
else if (mxGetM(prhs[6]) != mxGetN(prhs[0]))
{
    mexErrMsgTxt("Matrix of distances between with sampled and
                  estimating points with wrong dimensions.\n");
}

/*vector of y must be a column vector*/
```

```
mcoef = mxGetPr(prhs[0]);
mdist = mxGetPr(prhs[1]);
xval = mxGetPr(prhs[2]);
phi = mxGetPr(prhs[3]);
vdist = mxGetPr(prhs[6]);

sp = mxGetN(prhs[0]);
nDyadic = mxGetM(prhs[2]);

ntam = (int)pow(2, (double)J) + 2 * N -2;
ndim = ntam * (sp + 1);
p2J = (int)pow(2, (double)J);

plhs[0] = mxCreateDoubleMatrix(ndim,ndim, mxREAL);
output = mxGetPr(plhs[0]);
plhs[1] = mxCreateDoubleMatrix(ndim,1, mxREAL);
matB = mxGetPr(plhs[1]);

Awork = mxCreateDoubleMatrix(ntam, ntam, mxREAL);
mVar = mxGetPr(Awork);

for (i = 0; i < ndim; i++)
{
    for (j = 0;j < ndim; j++)
    {
        output[j * ndim + i] = 0;
    }
}

/*last columns*/
for (i = 1; i <= sp; i++)
{
    k = 0;
    l = 0;
    for (p = ((i-1)*ntam + 1); p <= i*ntam; p++)
    {

        for (q = (ndim-ntam+1); q <= ndim; q++)
        {
            if (k == l)
            {
                output[(q-1)* ndim + (p-1)] = 1;
            }
            else
            {
                output[(q-1) * ndim + (p-1)] = 0;
            }
            l++;
        }
        l = 0;
        k++;
    }
}

/*last rows*/
for (i = 1; i <= sp;i++)
{
    k = 0;
```

```
        l = 0;
        for (p = (ndim-ntam+1); p <= ndim; p++)
        {
            for (q = ((i-1)*ntam + 1); q <= (i*ntam); q++)
            {

                if (k == l)
                {
                    output[(q-1)*ndim + (p-1)] = 1;
                }
                else
                {
                    output[(q-1)*ndim + (p-1)] = 0;
                }
                l++;
            }
            l = 0;
            k++;
        }
    }

    /*last row and last column*/

    /*inside part*/
    for (i = 1; i <= sp; i++)
    {
        for (j = 1; j <= sp; j++)
        {
            EmpCross(mdist[(j-1)*sp + (i-1)], mVar);
            for (p = (-2*N+2);p <= (p2J-1); p++)
            {
                aq = max3(-2*N+2, -2*N+2+p, -2*N+2);
                bq = min3(2*N-2+p,2*N-2+p,p2J-1);
                for (q = aq;q <= bq; q++)
                {
                    printf("Matrix Q: i = %d, j = %d, p = %d,
                              q = %d N = %d.\n",i,j,p,q, N);
                    row = (i-1)*ntam + (p+2*N-1) -1;
                    col = (j-1)*ntam + (q+2*N-1) -1;
                    ak = max3(-2*N +2, -2*N +2 + p, -2*N +2 + q);
                    bk = min3(p2J-1,2*N-2+p, 2*N-2+q);
                    for (k = ak;k <= bk; k++)
                    {
                        al = max3(-2*N+2,-2*N+2,-2*N+2+k);
                        bl = min3(p2J-1, 2*N-2+k, 2*N-2+k);
                        for (l = al;l <= bl; l++)
                        {
                                output[col*ndim + row] = output[col*ndim + row] +
                                    mVar[(q+2*N-2)*ntam +
                                    (p+2*N-2)] *IntPhi_pklq(p,k,l,q);
                        }
                    }
                }
            }
        }
    }

    /*filling independent matrix matB*/
    for (i = 1; i <= sp; i++)
```

```
    {
        EmpCross(vdist[i-1], mVar);
        for (j = -2*N+2;j <= p2J-1; j++)
        {
            printf("matrix b: i = %d j = %d N = %d.\n", i, j, N);
            matB[(i-1)*ntam + (j+2*N-2)] = 0;
            ak = max3(-2*N+2+j, -2*N+2, -2*N + 2);
            bk = min3(p2J-1, 2*N-2+j, 2*N-2+j);
            for (k = ak; k <= bk; k++)
            {
                al = max3(-2*N+2, -2*N+2+k,-2*N+2+j);
                bl = min3(p2J-1,2*N-2+j, 2*N-2+k);
                for (l = al; l<= bl; l++)
                {
                    matB[(i-1)*ntam + (j+2*N-2)] = matB[(i-1)*ntam + (j+2*N-2)] +
                    mVar[(k+2*N-2)*ntam + (l+2*N-2)]*IntPhi_jkl(j,k,l);
                }
            }
        }
    }

    /*c_k*/
    for (j = -2*N+2;j <= p2J-1; j++)
    {
        matB[sp*ntam + (j+2*N-2)] = integration(j);
    }

}

void EmpCross(double h, double* res)
{
    int cont; /*check if we get somebody*/
    int i, j, k, l; /*loop variable*/
    int Mcoef; /*row number*/
    double e; /*tolerance*/

    Mcoef = (int)pow(2, (double)J) + 2 * N - 2;

    for (k = 0; k < Mcoef; k++)
    {
        for (l = 0; l < Mcoef; l++)
        {
            res[l*Mcoef + k] = 0;
        }
    }

    e = 0.1; /*tolerance*/
    cont = 0;
    while (cont == 0)
    {
        for (i = 0; i < sp; i++)
        {
            for (j = 0; j < sp; j++)
            {
                if ((mdist[j*sp+i] < h + e) && (mdist[j*sp+i] > h - e))
                {
                    cont++;
                    for (k = 0; k < Mcoef; k++)
                    {
```

```
                        for (l = 0; l < Mcoef; l++)
                        {
                            res[l*Mcoef + k] = res[l*Mcoef + k] +
                            (mcoef[i*Mcoef+l] - mcoef[j*Mcoef+l]) *
                            (mcoef[i*Mcoef+k] - mcoef[j*Mcoef+k]);
                        }
                    }
                }
            }
        }

        /*If we don't catch nobody*/
        if (cont == 0)
        {
            printf("Nobody at neighbour.\n");
            e = e + 0.001;
        }
        else /*If we catch somebody*/
        {
            for (k = 0; k < Mcoef; k++)
            {
                for (l = 0; l < Mcoef; l++)
                {
                    res[l*Mcoef + k] = res[l*Mcoef + k] / (2 * cont) ;
                }
            }
        }
    }
}

double IntPhi_pklq(int p, int k, int l, int q)
{
    double tol, x, valor;

    tol = xval[1] - xval[0];
    x = 0;
    valor = 0;
    while (x <= 1)
    {
        valor = valor + dbN_pklq(x, p, k, l, q);
        x = x + tol;
    }
    valor = valor * tol * pow(2, 2 * (double)J);
    return valor;
}

double IntPhi_jkl(int j, int k, int l)
{
    double tol, x, valor;

    tol = xval[1] - xval[0];
    x = 0;
    valor = 0;
    while (x <= 1)
    {
        valor = valor + dbN_jkl(x, j, k, l);
        x = x + tol;
    }
    valor = valor * tol * pow(2, (3 * (double)J)/2);
```

```
    return valor;
}

double integration(int i)
{
    int a, b;
    double valor = 0;

    if ((1 > (double)i / pow(2, (double)J)) &&
                ((double)(2* N + i - 1)  / pow(2, (double)J) > 0))
    {
        a = 0;
        while (xval[a] < max3(-i,0,0))
        {
            a++;
        }
        b = a;
        while (xval[b] < min3(2*N-1,2*N-1,(int)pow(2,(double)J)-i))
        {
            b++;
        }
        for (i = a; i <= b; i++)
        {
            valor = valor + phi[i];
        }
        valor = valor * (xval[1] - xval[0]) * pow(2, -((double)J)/2);
    }
    return valor;
}

double dbN_pklq(double x, int p, int k, int l, int q)
{
    double res;

    res = dbN(pow(2,(double)J) * x - (
                double)p) * dbN(pow(2,(double)J) * x - (double)k);
    res = res * dbN(pow(2,(double)J) * x - (double)l) *
                          dbN(pow(2,(double)J) * x - (double)q);

    return res;
}

double dbN_jkl(double x, int j, int k, int l)
{
    double res;

    res = dbN(pow(2,(double)J) * x - (double)j) * dbN(pow(2,(double)J) * x
                - (double)k) * dbN(pow(2,(double)J) * x - (double)l);

    return res;
}

double dbN(double x)
{
    double res;
    int cont;


    if ((x >= 0) && (x <= 2*(double)N - 1))
```

```
    {
        cont = 0;
        while (xval[cont] < x)
        {
            cont++;
        }
        res = phi[cont];
    }
    else
    {
        res = 0;
    }

    return res;
}

int maximo(int i1, int i2, int i3, int i4, int i5)
{
    int res;

    res = i1;

    if (i2 > res)
    {
        res = i2;
    }
    if(i3 > res)
    {
        res = i3;
    }
    if (i4 > res)
    {
        res = i4;
    }
    if (i5 > res)
    {
        res = i5;
    }
    return res;
}

int max3(int i1, int i2, int i3)
{
    int res;

    res = i1;

    if (i2 > res)
    {
        res = i2;
    }
    if (i3 > res)
    {
        res = i3;
    }
    return res;
}

int min3(int i1, int i2, int i3)
```

```
{
    int res;

    res = i1;

    if (i2 < res)
    {
        res = i2;
    }
    if (i3 < res)
    {
        res = i3;
    }
    return res;
}
```

A.16 hatchi

Objetivo

Esta função computa o valor da curva $\chi_{\boldsymbol{s}_0}(t)$ no ponto t, quando os vetores $\boldsymbol{b}_i$ do modelo de Krigagem Tempo-Variante Funcional são conhecido ou estimados e base de funções utilizadas são as ondaletas da família Daubechies.

Sintaxe

```
fp  = hatchi(t, Iter, J, N, mcoef, mb)
```

Descrição

A função `hatchi` tem seis argumentos de entrada

i. `t` é um vetor coluna $(t_1, \ldots, t_m)^\top$ com $t_j \in [0, 1]$

ii. `Iter` é um número inteiro positivo em que $\frac{1}{2^{Iter}}$ é a precisão utilizada nas aproximações numéricas de integrais envolvendo a função escala $\phi(x)$

iii. `J` é o nível de aproximação

iv. `N` é um número inteiro positivo indexando a família Daubechies

v. `mcoef` é uma matriz em que a i-ésima coluna contém os coeficientes da curva $\chi_{\boldsymbol{s}_i}(t)$

vi. `mb` é uma matriz em que a i-ésima coluna consiste do vetor $\boldsymbol{b}_i$ que é o vetor de coeficientes da função $\lambda_i(t)$ no modelod de Krigagem Tempo-Variante Funcional

e retorna um vetor coluna $(\chi_{\boldsymbol{s}_0}(t_1), \ldots, \chi_{\boldsymbol{s}_0}(t_m))^\top$ com o mesmo número de linhas do vetor `t`.

Código Fonte

Listing A.20: *Arquivo* `hatchi.c`.

```
 function [ fp ] = hatchi(t, Iter, J, N, mcoef, mb)
% hatchi_t: calculate \hat{\chi}_{s_0}(t) given mcoef e \lambda_1, \dots,
%\lambda_{sp} of functional kriging predictor
% Inputs:
%    1)t: column vector of values in [0,1]
%    2)Iter: order of dyadic points \phi at values \dfrac{l}{2^Iter}
%    3)J: approximation level
%    4)N: order in the daubechies family
%    5)mcoef:matrix with columns representing coefficients of sampled spots
%    6)mb: coefficients matrix estimated of functional kriging predictor
%  output:
%   1)chi_{s_0}: estimated chi_{s_0} evaluated at t

    [phi, ~, xval] = wavefun(['db',num2str(N)], Iter);

    fp = zeros(size(t));

    n = size(mcoef, 2); %number of points sampled in the space dimension

    for i = 1:n
        fp = fp + chi_t(t,xval(:),phi(:),J, N, mb(:,i)) .*...
             chi_t(t,xval(:),phi(:),J, N, mcoef(:,i));
    end
end
```

Apêndice B

Estações Meteorológicas – Províncias Marítimas Canadenses

Na Tabela B.1, mostramos as informações referentes as 82 estaçoes meteorológicas selecionadas nas províncias marítimas. Incluímos nessa tabela quatro informações

ID Número usado pela agência canadense *Environment and Climate Change Canada* (ECCC) para identificar as estações meteorológicas;

Nome Nome da estação;

Latitude Latitude da localização da estação;

Longitude Longitude da localização da estação.

ID	Nome	latitude	longitude
6106	ACADIA FOREST EXP ST	45.99	-66.36
6108	ALMA	45.60	-64.95
6109	AROOSTOOK	46.71	-67.72
6915	BAS CARAQUET	47.80	-64.83
6916	BATHURST A	47.63	-65.75
6119	BERTRAND	47.75	-65.07
6137	CHARLO A	47.98	-66.33
6146	COLESON COVE	45.15	-66.20
6150	DOAKTOWN	46.55	-66.14
6157	FREDERICTON A	45.87	-66.53
10981	FREDERICTON AQUATIC CENTRE C/S	45.96	-66.65
6160	GAGETOWN 2	45.78	-66.15
6170	HARCOURT	46.50	-65.27
27482	HARVEY STATION	45.73	-67.02
6175	HAUT SHIPPAGAN	47.75	-64.77
26970	HAVELOCK	46.00	-65.32
6180	HOYT BLISSVILLE	45.60	-66.57
6181	JUNIPER	46.55	-67.17
26968	KOUCHIBOUGUAC CS	46.77	-65.01
6194	MACTAQUAC PROV PARK	45.95	-66.90
6197	MAPLETON	46.18	-67.23
6140	MIRAMICHI A	47.01	-65.47

6206	MONCTON	46.10	-64.79
6207	MONCTON A	46.11	-64.68
6208	MOUNT CARLETON	47.42	-66.93
6212	NAUWIGEWAUK	45.47	-65.90
6213	NEPISIGUIT FALLS	47.40	-65.78
6215	NICTAU	47.23	-67.15
6219	PARKINDALE	45.87	-65.07
6220	PENNFIELD	45.10	-66.73
6232	REXTON	46.67	-64.87
6241	SACKVILLE	45.85	-64.38
6250	SAINT JOHN A	45.32	-65.89
6256	ST LEONARD A	47.16	-67.83
6268	SUSSEX	45.72	-65.53
6280	UPSALQUITCH LAKE	47.46	-66.42
6920	WOLFE LAKE CS	45.67	-65.15
6921	AVONDALE	45.02	-64.12
27602	BERRYS BAY	43.66	-65.26
6308	BRIDGEWATER	44.40	-64.55
6318	CHARLESVILLE	43.58	-65.78
27600	CHETICAMP CS	46.65	-60.95
6329	COLLEGEVILLE	45.48	-62.02
6334	DEBERT	45.42	-63.42
6336	DEMING	45.22	-61.18
6354	GREENWOOD A	44.98	-64.92
6357	HALIFAX CITADEL	44.65	-63.58
6358	HALIFAX STANFIELD INT'L A	44.88	-63.50
6370	JACKSON	45.58	-63.83
6923	KEJIMKUJIK 1	44.40	-65.20
27742	LAKE MAJOR	44.72	-63.48
6383	LIVERPOOL BIG FALLS	44.13	-64.93
6388	LOUISBOURG	45.90	-60.00
6393	LYONS BROOK	45.66	-62.80
6409	MIDDLE MUSQUODOBOIT	45.07	-63.10
6407	MIDDLEBORO	45.77	-63.57
6414	NAPPAN CDA	45.77	-64.25
6428	PARRSBORO	45.40	-64.33
6435	POCKWOCK LAKE	44.77	-63.83
6437	POINT ACONI	46.32	-60.33
6447	PUGWASH	45.84	-63.66
6465	SHEARWATER A	44.63	-63.50
26969	SOUTH MOUNTAIN	45.02	-64.68
27282	SOUTH SIDE HARBOUR	45.62	-61.90
6473	SPRINGFIELD	44.67	-64.85
6456	ST MARGARET'S BAY	44.70	-63.90
6484	SUMMERVILLE	45.12	-64.18
6486	SYDNEY A	46.17	-60.05
6927	TATAMAGOUCHE	45.68	-63.23
27868	TRENTON MUNICIPAL A	45.61	-62.62
6497	WATERVILLE CAMBRIDGE	45.05	-64.65

6512	WINDSOR MARTOCK	44.93	-64.17
6514	WRECK COVE BROOK	46.53	-60.45
6516	YARMOUTH A	43.83	-66.09
6519	ALBERTON	46.85	-64.02
6520	ALLISTON	46.07	-62.60
6522	BANGOR	46.35	-62.68
6929	ELMWOOD	46.25	-63.33
6538	LONG RIVER	46.50	-63.55
6536	MONTICELLO	46.47	-62.47
6540	O'LEARY	46.70	-64.26
6931	VICTORIA	46.22	-63.49

Tabela B.1: *Tabela com as informações das estações meteorológicas selecionadas nas províncias marítimas canadenses.*

Apêndice C

Estações Meteorológicas – Região Metropolitana de São Paulo

Na Tabela C.1, mostramos as informações referentes as 14 estaçoes meteorológicas selecionadas na região metropolitana de São Paulo mantidas pela CETESB (Companhia Ambiental do Estado De São Paulo). Incluímos nessa tabela quatro informações

ID Número usado pela CETESB para identificar as estações meteorológicas;

Nome Nome da estação;

Latitude Latitude da localização da estação;

Longitude Longitude da localização da estação.

Tabela C.1: *Tabela com as informações das estações meteorológicas da CETESB selecionadas na Região Metropolitana de São Paulo.*

Nome	ID	Latitude	Longitude
Santana	63	-23.5151	-46.6293
Santo Amaro	64	-23.6546	-46.7096
Pq Dom Pedro II	72	-23.544	-46.6277
Congonhas	73	-23.6158	-46.6633
Mooca	85	-23.5477	-46.6031
São Caetano	86	-23.6182	-46.5563
Diadema	92	-23.6881	-46.6133
Nossa Senhora do Ó	96	-23.4798	-46.6924
Osasco	120	-23.5266	-46.7923
Sto André - Pço Municipal	254	-23.655	-46.5316
Carapicuiba	263	-23.5301	-46.8357
Guarulhos - Pço Municipal	264	-23.4559	-46.5184
Itaim Paulista	266	-23.5017	-46.4204
Pte Remédios	270	-23.5092	-46.7095

Referências Bibliográficas

Banerjee *et al.*(**2008**) Sudipto Banerjee, Alan E Gelfand, Andrew O Finley e Huiyan Sang. Gaussian predictive process models for large spatial data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(4):825–848. Citado na pág. 66

Banerjee *et al.*(**2014**) Sudipto Banerjee, Bradley P Carlin e Alan E Gelfand. *Hierarchical modeling and analysis for spatial data*. Crc Press. Citado na pág. 7, 22, 41, 42, 65

Boggess e Narcowich(2009) Albert Boggess e Francis J Narcowich. *A first course in wavelets with Fourier analysis*. John Wiley & Sons. Citado na pág. 12, 35

Bohorquez *et al.*(**2015**) Martha Bohorquez, Ramón Giraldo e Jorge Mateu. Optimal sampling for spatial prediction of functional data. *Statistical Methods & Applications*, páginas 1–16. Citado na pág. 2

Brown *et al.*(**1994**) Philip J Brown, Nhu D Le e James V Zidek. Multivariate spatial interpolation and exposure to air pollutants. *Canadian Journal of Statistics*, 22(4):489–509. Citado na pág. 64

Caballero *et al.*(**2013**) William Caballero, Ramón Giraldo e Jorge Mateu. A universal kriging approach for spatial functional data. *Stochastic Environmental Research and Risk Assessment*, 27(7):1553–1563. Citado na pág. 3

Chiles e Delfiner(2009) Jean-Paul Chiles e Pierre Delfiner. *Geostatistics: modeling spatial uncertainty*, volume 497. John Wiley & Sons. Citado na pág. 7

Chui(2014) Charles K Chui. *An introduction to wavelets*, volume 1. Academic press. Citado na pág. 13

Cressie e Johannesson(2008) Noel Cressie e Gardar Johannesson. Fixed rank kriging for very large spatial data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):209–226. Citado na pág. 66

Cressie e Cassie(1993) Noel AC Cressie e Noel A Cassie. *Statistics for spatial data*, volume 900. Wiley New York. Citado na pág. 7

Daubechies *et al.*(**1992**) Ingrid Daubechies *et al. Ten lectures on wavelets*, volume 61. SIAM. Citado na pág. 12, 16

De Oliveira *et al.***(1997)** Victor De Oliveira, Benjamin Kedem e David A Short. Bayesian prediction of transformed gaussian random fields. *Journal of the American Statistical Association*, 92 (440):1422–1433. Citado na pág. 64

de Souza e Dias(2010) Camila P. Estevam de Souza e Ronaldo Dias. *Introdução à análise de dados funcionais*. Associação Brasileira de Estatística, São Paulo - SP. Citado na pág. 5

Delicado *et al.***(2010)** Pedro Delicado, Ramón Giraldo, C Comas e Jorge Mateu. Statistics for spatial functional data: Some recent contributions. *Environmetrics*, 21(3–4):224–239. Citado na pág. 2

Diggle e Ribeiro(2007) Peter Diggle e Paulo Justiniano Ribeiro. *Model-based geostatistics*. Springer Science & Business Media. Citado na pág. 7

Diggle e Ribeiro Jr(2002) Peter J Diggle e Paulo J Ribeiro Jr. Bayesian inference in gaussian model-based geostatistics. *Geographical and Environmental Modelling*, 6(2):129–146. Citado na pág. 63

Ecker(2003) Mark D Ecker. Geostatistics: past, present and future. *Environmetrics. Developed under the auspices of UNESCO. Encyclopedia of Life Support Systems (EOLSS). Ed. by AH El-Shaarawi and J. Jureckova, Eolss Publishers, Oxford, UK. http://www. eolss. net*. Citado na pág. 1

Ferraty e Vieu(2006) Frédéric Ferraty e Philippe Vieu. *Nonparametric functional data analysis: theory and practice*. Springer Science & Business Media. Citado na pág. 2, 5

Frederick Pearson(1990) II Frederick Pearson. *Map ProjectionsTheory and Applications*. CRC press. Citado na pág. 7

Fuentes(2001) Montserrat Fuentes. A high frequency kriging approach for non-stationary environmental processes. *Environmetrics*, 12(5):469–483. Citado na pág. 64

Genton e Kleiber(2015) Marc G. Genton e William Kleiber. Cross-covariance functions for multivariate geostatistics. *Statistical Science*, 30(2):147–163. Citado na pág. 31, 65

Goovaerts(1997) Pierre Goovaerts. *Geostatistics for Natural Resources Evaluation*. Oxford University Press. Citado na pág. 1

Goulard e Voltz(1992) Michel Goulard e Marc Voltz. Linear coregionalization model: tools for estimation and choice of cross-variogram matrix. *Mathematical Geology*, 24(3):269–286. Citado na pág. 31, 35

Haar(1909) Kármán Th. v. Haar, A. Zur theorie der spannungszustände in plastischen und sandartigen medien. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, 1909:204–218. Citado na pág. 15

Haining(2003) Robert Patrick Haining. *Spatial data analysis*. Cambridge University Press Cambridge. Citado na pág. 7

Härdle *et al.*(1998) Wolfgang Härdle, Gerard Kerkyacharian, Alexander Tsybakov e Dominique Picard. *Wavelets, approximation, and statistical applications*. Springer. Citado na pág. 14, 25

Henao(2009) Ramon Giraldo Henao. *Geostatistical analysis of functional data*. Tese de Doutorado, Universitat Politècnica de Catalunya, Barcelona. Citado na pág. 2, 45, 63, 64

Henao *et al.*(2010) Ramon Giraldo Henao, Pedro Delicado e Jorge Mateu. Continuous time-varying kriging for spatial prediction of functional data: An environmental application. *Journal of agricultural, biological, and environmental statistics*, 15(1):66–82. Citado na pág. 2, 3, 27, 28, 35, 45, 56, 63, 64

Henao *et al.*(2011) Ramon Giraldo Henao, Pedro Delicado e Jorge Mateu. Ordinary kriging for function-valued spatial data. *Environmental and Ecological Statistics*, 18(3):411–426. Citado na pág. 2, 3, 22, 25, 45, 63, 64

Hengl *et al.*(2007) Tomislav Hengl, Gerard BM Heuvelink e David G Rossiter. About regression-kriging: from equations to case studies. *Computers & Geosciences*, 33(10):1301–1315. Citado na pág. 2

Hevesi *et al.*(1992) Joseph A. Hevesi, Jonathan D. Istok e Alan L. Flint. Precipitation estimation in mountainous terrain using multivariate geostatistics. part i: Structural analysis. *Journal of Applied Meteorology*, 31(7):661–676. Citado na pág. 1

Horváth e Kokoszka(2012) Lajos Horváth e Piotr Kokoszka. *Inference for functional data with applications*, volume 200. Springer Science & Business Media. Citado na pág. 2, 5

Ignaccolo *et al.*(2014) Rosaria Ignaccolo, Jorge Mateu e Ramon Giraldo. Kriging with external drift for functional data for air quality monitoring. *Stochastic Environmental Research and Risk Assessment*, 28(5):1171–1186. Citado na pág. 2

Jerrett *et al.*(2005) Michael Jerrett, Altaf Arain, Pavlos Kanaroglou, Bernardo Beckerman, Dimitri Potoglou, Talar Sahsuvaroglu, Jason Morrison e Chris Giovis. A review and evaluation of intraurban air pollution exposure models. *Journal of Exposure Science and Environmental Epidemiology*, 15(2):185–204. Citado na pág. 1

John W. Eaton e Wehbring(2015) Sören Hauberg John W. Eaton, David Bateman e Rik Wehbring. *GNU Octave version 4.0.0 manual: a high-level interactive language for numerical computations*. URL http://www.gnu.org/software/octave/doc/interpreter. Citado na pág. 65

Journel e Huijbregts(1978) A. G. Journel e Ch. J. Huijbregts. *Mining Geostatistics*. Academic Press, New York. Citado na pág. 1

Krige(1951) Danie Gerhardus Krige. A statistical aproach to some mine valuation and allied problems in the witwartersland. Dissertação de Mestrado, University of the Witwatersland, Johannesburg. Citado na pág. 2

Mallat(1989a) Stephane G Mallat. Multiresolution approximations and wavelet orthonormal bases of $L^2(\mathbb{R})$. *Transactions of the American Mathematical Society*, 315(1):69–87. Citado na pág. 18

Mallat(1989b) Stephane G Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693. Citado na pág. 18

Matheron(1963) Georges Matheron. Principles of geostatistics. *Economic geology*, 58(8):1246–1266. Citado na pág. 2

Meyer e Ryan(1993) Y Meyer e RD Ryan. Wavelets: Algorithms and applications. *SIAM, Philadelphia, PA*. Citado na pág. 12

Meyer(1985) Yves Meyer. Principe d'incertitude, bases hilbertiennes et algebres d'operateurs. *Séminaire Bourbaki*, 28:209–223. Citado na pág. 13

Meyer e Salinger(1995) Yves Meyer e David H Salinger. *Wavelets and operators*, volume 1. Cambridge University Press. Citado na pág. 13

Minnitt e Assibey-Bonsu(2014) R.C.A. Minnitt e W. Assibey-Bonsu. Professor Danie Gerhardus Krige. *Journal of the Southern African Institute of Mining and Metallurgy*, 114:vii–xi. Citado na pág. 1

Morettin(2014) Pedro Alberto Morettin. *Ondas e Ondaletas: Da Análise de Fourier à Análise de Ondaletas de Séries Temporais*. Edusp – Editora da Universidade de São Paulo, São Paulo. Citado na pág. 13, 16

Moyeed e Papritz(2002) Rana A Moyeed e Andreas Papritz. An empirical comparison of kriging methods for nonlinear spatial point prediction. *Mathematical Geology*, 34(4):365–386. Citado na pág. 64

Nerini *et al.***(2010)** David Nerini, Pascal Monestiez e Claude Manté. Cokriging for spatial functional data. *Journal of Multivariate Analysis*, 101(2):409–418. Citado na pág. 2, 64

Ogden(1997) Todd Ogden. *Essential wavelets for statistical applications and data analysis*. Springer. Citado na pág. 25, 93

Omre e Halvorsen(1989) Henning Omre e Kjetil B Halvorsen. The bayesian bridge between simple and universal kriging. *Mathematical Geology*, 21(7):767–786. Citado na pág. 63

Ostro *et al.***(1999)** Bart D Ostro, Susan Hurley e Michael J Lipsett. Air pollution and daily mortality in the coachella valley, california: a study of pm10 dominated by coarse particles. *Environmental research*, 81(3):231–238. Citado na pág. 59

Pilz e Spöck(2008) Jürgen Pilz e Gunter Spöck. Why do we need and how should we implement bayesian kriging methods. *Stochastic Environmental Research and Risk Assessment*, 22(5): 621–632. Citado na pág. 63

Pope III e Dockery(2006) C Arden Pope III e Douglas W Dockery. Health effects of fine particulate air pollution: lines that connect. *Journal of the air & waste management association*, 56 (6):709–742. Citado na pág. 59

Pope III *et al.***(1991)** C Arden Pope III, Douglas W Dockery, John D Spengler e Mark E Raizenne. Respiratory health and pm10 pollution: a daily time series analysis. *American Review of Respiratory Disease*, 144(3_pt_1):668–674. Citado na pág. 59

R Core Team(2015) R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015. URL https://www.R-project.org/. Citado na pág. 65

Ramsay e Dalzell(1991) J. O. Ramsay e C. J. Dalzell. Some tools for functional data analysis. *Journal of the Royal Statistical Society. Series B (Methodological)*, 53(3):539–572. Citado na pág. 2

Ramsay(2006) James O Ramsay. *Functional data analysis*. Wiley Online Library. Citado na pág. 2, 5, 56

Ramsay e Silverman(2002) James O Ramsay e Bernard W Silverman. *Applied functional data analysis: methods and case studies*, volume 77. Springer New York. Citado na pág. 2, 5, 6

Ramsay *et al.***(2009)** James O. Ramsay, Giles Hooker e Spencer Graves. *Functional Data Analysis with R and MATLAB*. Springer. Citado na pág. 2, 5

Reyes *et al.***(2015)** Adriana Reyes, Ramón Giraldo e Jorge Mateu. Residual kriging for functional spatial prediction of salinity curves. *Communications in Statistics-Theory and Methods*, 44(4): 798–809. Citado na pág. 2

Salazar *et al.***(2015)** Elías Salazar, Ramón Giraldo e Emilio Porcu. Spatial prediction for infinite-dimensional compositional data. *Stochastic Environmental Research and Risk Assessment*, 29 (7). Citado na pág. 2

Schwartz *et al.***(1993)** Joel Schwartz, Daniel Slater, Timothy V Larson, William E Pierson e Jane Q Koenig. Particulate air pollution and hospital emergency room visits for asthma in seattle. *American Review of Respiratory Disease*, 147(4):826–831. Citado na pág. 59

Seaton *et al.***(1995)** Anthony Seaton, D Godden, W MacNee e K Donaldson. Particulate air pollution and acute health effects. *The Lancet*, 345(8943):176–178. Citado na pág. 59

Stein e Corsten(1991) A Stein e LCA Corsten. Universal kriging and cokriging as a regression procedure. *Biometrics*, 47(2):575–587. Citado na pág. 3

Stein(2012) Michael L Stein. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media. Citado na pág. 7

Styan(1973) George PH Styan. *Hadamard products and multivariate statistical analysis*, volume 6. Elsevier. Citado na pág. 38, 39

Sun *et al.*(**2012**) Ying Sun, Bo Li e Marc G Genton. Geostatistics for large datasets. Em *Advances and challenges in space-time modelling of natural events*, páginas 55–77. Springer. Citado na pág. 66

Vidakovic(2009) Brani Vidakovic. *Statistical modeling by wavelets*, volume 503. John Wiley & Sons. Citado na pág. 25

Wackernagel(2003) Hans Wackernagel. *Multivariate geostatistics*. Springer. Citado na pág. 35