UNIVERSIDADE ESTADUAL DE CAMPINAS

INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E COMPUTAÇÃO CIENTÍFICA

DEPARTAMENTO DE ESTATÍSTICA

SELEÇÃO ADAPTATIVA DE FUNÇÕES DE BASE EM ANÁLISE DE DADOS FUNCIONAIS VIA PENALIZAÇÃO ESTOCÁSTICA

Cézar Augusto de Freitas Anselmo

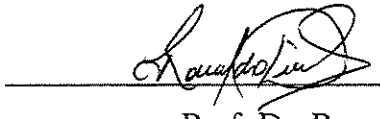
Orientador: Prof. Dr. Ronaldo Dias

- Campinas, Março de 2003 -

SELEÇÃO ADAPTATIVA DE FUNÇÕES DE BASE EM ANÁLISE DE DADOS FUNCIONAIS VIA PENALIZAÇÃO ESTOCÁSTICA

Este exemplar corresponde à redação final da dissertação devidamente corrigida e defendida por Cézar Augusto de Freitas Anselmo e aprovada pela comissão julgadora.

Campinas, 07 de março de 2003.



Prof. Dr. Ronaldo Dias

Orientador

Banca Examinadora

1. Ronaldo Dias
2. Jorge Alberto Achcar
3. Nancy Lopes Garcia

Dissertação apresentada ao Instituto de Matemática, Estatística e Computação Científica da UNICAMP como requisito parcial para obtenção do Título de MESTRE em ESTATÍSTICA.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Anselmo, Cézar Augusto de Freitas

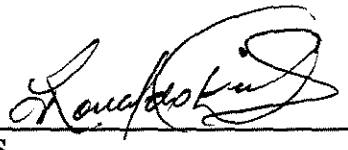
An82s Seleção adaptativa de funções de base em análise de dados funcionais via penalização estocástica / Cézar Augusto de Freitas Anselmo - Campinas, [S.P. :s.n.], 2003.

Orientador : Ronaldo Dias

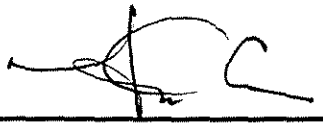
Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Estatística não-paramétrica. 2. Análise funcional. 3. Spline, teoria do. I. Dias, Ronaldo. II. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. III. Título.

Dissertação de Mestrado defendida em 07 de março de 2003 e aprovada pela Banca Examinadora composta pelos Profs. Drs.



Prof (a). Dr (a). RONALDO DIAS



Prof (a). Dr (a). JORGE ALBERTO ACHCAR

Prof (a). Dr (a). NANCY LOPES GARCIA

Em memória de Pietra (Pringa) Nikolievitch Rasputina.

Agradecimentos

- Em primeiro lugar, agradeço a Deus, pela Sua maravilhosa criação e por ter permitido que eu chegasse até aqui.
- Aos meus pais, Lessa e Paulo Anselmo, por não terem medido esforços na tarefa de me educar.
- À Yvette (Popoita) Alves Tynus, por apoiar-me nas minhas decisões, mesmo sem saber aonde elas nos levariam.
- Aos meus irmãos, Augusta e Leonardo Anselmo, por sua dedicação e preocupação nos momentos de necessidade.
- Ao meu orientador, Prof. Ronaldo Dias, por acreditar em minha competência, presenteando-me com seu tão querido projeto científico.
- Aos Profs. Filidor Edilfonso Vilca Labra e Manoel Raimundo de Sena Junior, pelo importante apoio na fase inicial de minha morada em Campinas.
- Às Profas. Maria Cristina Falcão Raposo e Claudia Regina Oliveira de Paiva Lima, pela constante preocupação com meu bem estar e pelas muitas orientações em nossas discussões científicas.
- Ao Prof. Francisco de Assis Tenório de Carvalho, alicerce importante na construção de meu caráter de pesquisador científico.
- Aos primos Antônio Saldoildo Freitas Tenório e Antônio Saldoildo Freitas Tenório Filho, de intelectualidades admiráveis, por saber que sempre poderei contar com sua ajuda.
- Ao Prof. Aluísio de Souza Pinheiro, pelo atendimento nos momentos de dúvidas e esclarecimentos pessoais.
- Aos Profs. Luiz Koodi Hotta, Nancy Lopes Garcia, Hildete Prisco Pinheiro e Hervé Jean François Guiol, pelos valorosos conselhos e dicas ao longo das disciplinas ministradas.
- Ao Prof. Francisco Cribari-Neto, sinônimo de disciplina e competência científicas, por suas dicas computacionais.

- Aos amigos de jornada Benilton de Sá Carvalho e Artur Iuri Alves de Sousa, pelo companheirismo, apoio e conselhos pessoais e acadêmicos, sem os quais seria mais difícil trilhar esse caminho.

- Aos amigos da UFPE, em especial Tarci, Tati, Déa, Serafinho, Alexandre, Eduardo, Marcelo, Rodrigo, Carlos, Denis, George, Eufra, Heráclito, Rô, Renato, Wando, JVM, Oscar, Tiago, Prof. Silvio e Renata, e aos colegas de mestrado, Rose, Roger e Raquel, pelos bons momentos compartilhados.

- À querida Laih-la Silva Freitas, dádiva e luz no meu caminho.

- Ao Prof. Sérgio Furtado dos Reis, por instigar discussões produtivas, em nossos seminários conjuntos.

- Ao Prof. José Plínio de Oliveira Santos, por proporcionar-me momentos de introspecção e vislumbre em seus seminários de Teoria Combinatória dos Números.

- A Guilherme Laurentino de Lima Filho e Gisele Fiorelli, pelas sugestões e correções ortográficas.

- Aos companheiros da República BeerMania, Hiawatha, Autista, Morbiolo, Embrulho, Mossias, Cedrix, Adriano e Gisele Bortoleto, pela acolhida nesses anos.

- Ao amigo Reginaldo, homem de boa conduta, pelos momentos de desabafo.

- Aos companheiros da Dança e do Ciclismo, pelos momentos de entretenimento, indispensáveis ao ser humano.

- Aos demais funcionários do IMECC – UNICAMP, em especial a Tânia M. M. Trinchinato e Maria Aparecida (Cidinha) Miccerino de Almeida, por sua ajuda no que foi necessário para finalização dessa dissertação.

- Aos familiares que estiveram atentos à minha vida nesse período.

- À FAPESP, pelo apoio financeiro.

Muitíssimo obrigado.

Resumo

A área da análise não-paramétrica de dados responsável pelos casos em que esses se constituem de uma coleção de curvas é denominada Análise de Dados Funcionais. O crescente interesse nesse estudo tem motivado o surgimento de pesquisas nessa direção.

Esse trabalho propõe um novo método em Análise de Dados Funcionais. Os dados $(x_i)_{i=1..m}$ consistem de um conjunto de m curvas inter-relacionadas que, individualmente, podem ser consideradas como combinação linear de funções apropriadamente escolhidas e que geram uma base para o espaço de funções para as curvas em questão. Dessa forma, a técnica de mínimos quadrados penalizados é utilizada conjuntamente com o conceito de penalização estocástica, onde se introduz uma variável aleatória Bernoulli associada a cada coeficiente da expansão linear.

A explicação por considerar o problema como funcional ao invés de multivariado reside na suposição de que x_i é alguma função bem comportada e na possibilidade de estimar funcionais, como derivadas e integrais. Questões como custo computacional e plausibilidade das suposições do modelo em geral foram de interesse primário. As simulações iniciais foram feitas através do *software* R (http://cran.r-project.org) e posteriormente, por motivos de comparação e velocidade, foi realizada sua implementação em Ox (http://www.nuff.ox.ac.uk/users/Doornik). Através delas, utilizou-se uma variante do algoritmo SEM para estimação dos parâmetros do modelo.

Em todas as simulações esteve implícito que o argumento sobre o qual são feitas as medidas, t, é unidimensional e que as curvas são registradas e alinhadas em algum sentido, de modo que propriedades importantes de cada curva ocorram aproximadamente sobre os mesmos valores de t. O método mostrou-se adaptativo, de maneira que capturou as nuances importantes de cada curva, conseguindo resumi-las na estimativa final que, nas simulações, demonstrou ser uma excelente aproximação da curva verdadeira. Adicionalmente, os algoritmos produzidos perfizeram um custo computacional relativamente baixo quando utilizados nas simulações do modelo.

Palavras chave: análise de dados funcionais, seleção adaptativa, penalização estocástica, funções de base, algoritmo SEM, estatísticas funcionais, medidas resumo, *splines*, análise não-paramétrica de dados.

Abstract

The subject of nonparametric data analysis that deals with curve-shaped data is named Functional Data Analysis. This sub-area of nonparametrics has attracted an increasing interest of many researchers.

In this work we present a new method in Functional Data Analysis. The data $(x_i)_{i=1..m}$ consist of a collection of m curves interrelated that, individually, can be considered as a linear combination of functions appropriately chosen. These functions generate a basis to function space where one believes the curves live. In this way, penalized minimum squares method is jointly used with the idea of stochastic penalization by inserting a Bernoulli random variable linked to each coefficient of the curves linear expansion.

The reason to consider this problem as functional instead multivariate lies on supposition that x_i is a smooth function. Computational aspects and plausibility of suppositions were of primary interest. First simulations were performed in R software (http://cran.r-project.org). By reasons of comparison and velocity, we also implemented a version in Ox (http://www.nuff.ox.ac.uk/users/Doornik). We implemented a variant version of SEM algorithm to calculate parameters estimates.

All the simulations regarded that independent variable, t, is unidimensional and the curves are registered and aligned in some sense, assuring the curves features occur about the same t value. The simulations results show that the method is adaptive, able to recover important features of curves and the final curve estimate can be shown to be an excellent approximation to the true curve. In addition, the idealized algorithms achieved a relatively low computational cost in simulations of the model.

Keywords: functional data analysis, adaptive selection, stochastic penalization, basis functions, SEM algorithm, functional statistics, summary statistics, splines, non-parametric data analysis.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

1 *Introdução*

1.1 Análise de Dados Funcionais

Dados de inúmeras áreas surgem através de um processo naturalmente descrito como funcional. A Análise de Dados Funcionais[1] é um conjunto de técnicas que podem ser usadas para estudar a variação de funções em uma amostra bem como de suas derivadas. Tais funções podem ser obtidas de um processo de medida *online*, no qual temos as medidas avaliadas continuamente, ou ser resultado de um processo de suavização aplicado a dados discretos (Ramsay, 2001a).

O propósito maior é, sem dúvida, explicar a variação dentro das funções e também entre as mesmas. Mas a ADF não é apenas uma simples extensão das técnicas para dados não funcionais. Se por um lado há equivalentes funcionais das medidas descritivas usuais, como média, variância e correlação, bem como métodos tais quais análise de componentes principais, correlação canônica e regressão, por outro há uso de informação presente nas derivadas das funções sob estudo, uso esse incompleto ou mesmo inexistente nas técnicas não funcionais. Na ADF é a análise diferencial principal responsável por usar de maneira eficaz tal informação.

No entanto, quando se fala da variabilidade das funções em ADF, vêm à tona dois tipos de variação: variação de amplitude e variação de fase. A variação de amplitude é a maior responsável por dar ao processo sob estudo o comportamento que se deseja explicar, isto é, ela dá o padrão coletivo a cada indivíduo ou função. Dessa forma, se a função em estudo apresenta dois picos distintos em seu comportamento em todos ou na maioria dos indivíduos, é de se esperar que todo o conjunto apresente esse

[1] Nesse texto abreviada por ADF.

comportamento em maior ou menor escala. A variação de amplitude fornece, então, o padrão de comportamento do conjunto de dados. Uma possibilidade de variação de amplitude muito simples é exibida na Figura 1.1, enquanto que a Figura 1.2 exibe uma forma de variação de amplitude que dificulta a estimação. Em todas as simulações dessa seção foi usado um ruído de baixa variância para ilustrar melhor cada curva individualmente.

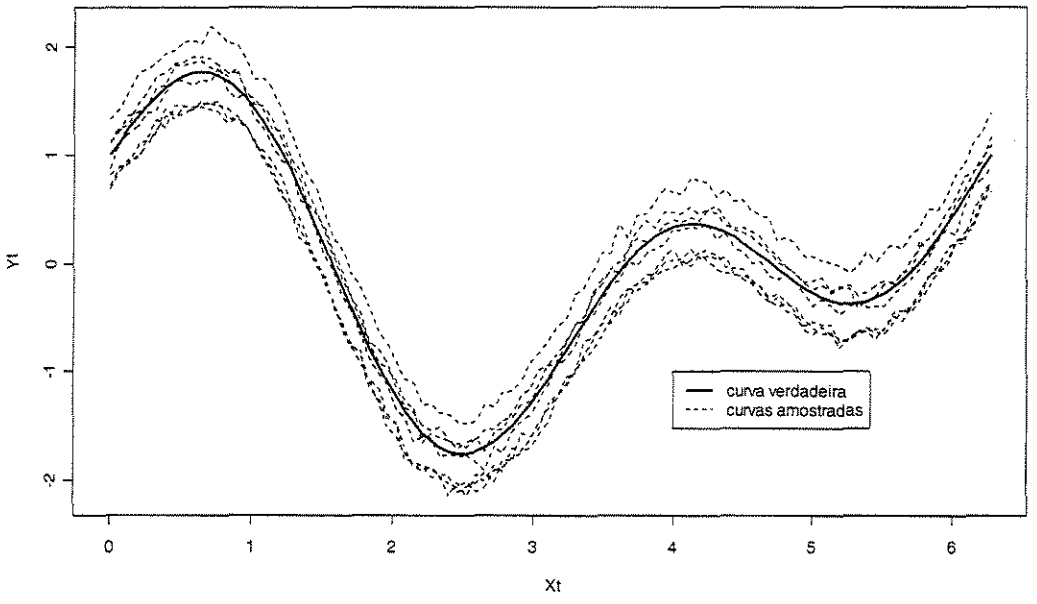


Figura 1.1: Curvas simuladas sob efeito simples de variação de amplitude adicionadas de ruído de variância pequena.

A variação de fase, por outro lado, pode mascarar o padrão de comportamento do conjunto de funções. Ela é um processo que desalinha as funções umas das outras, tornando difícil até mesmo com poucas funções detectar o comportamento padrão do conjunto, conforme pode ser visto na Figura 1.3. Quando os dois tipos de variação estão presentes – caso mais comum – técnicas de estimação mais elaboradas têm que ser usadas. A Figura 1.4 exibe um caso assim.

Observe como a variação de fase encobre o padrão por trás do processo, prejudicando, dessa forma, a estimação. A média seccional[2] é uma estatística descritiva

[2] Média aritmética posicional dos valores observados de todas as curvas, obtida para cada posição da variável independente.

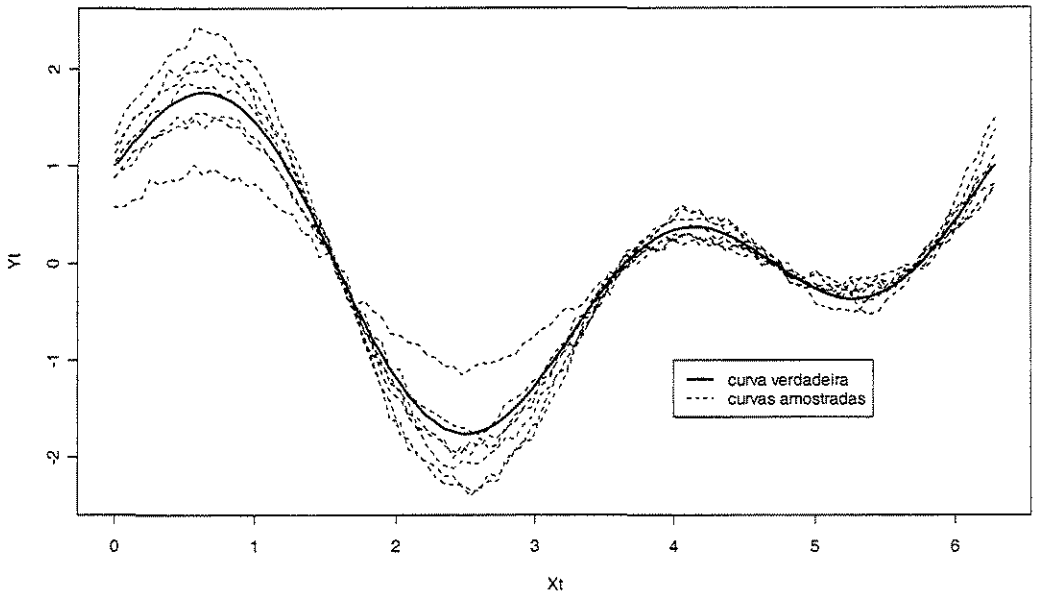


Figura 1.2: Curvas simuladas sob efeito moderado de variação de amplitude adicionadas de ruído de variância pequena.

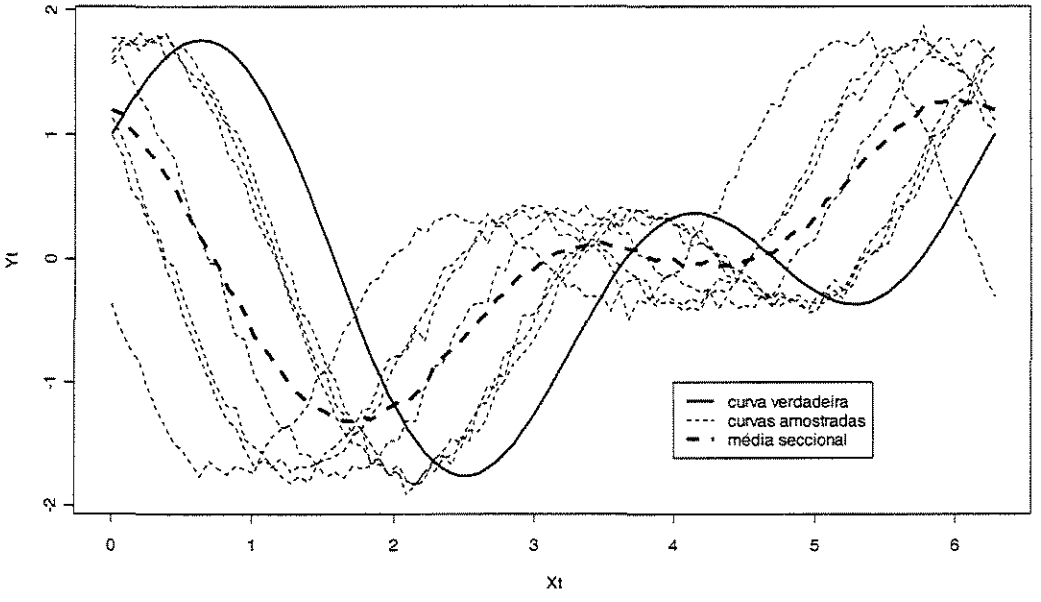


Figura 1.3: Curvas simuladas sob efeito de variação de fase adicionadas de ruído de variância pequena.

funcional que poderia dar uma idéia do processo que gerou as curvas. No entanto, devido ao desalinhamento entre as curvas, essa idéia é subestimada em toda sua extensão. Esse tipo de problema é comum em ADF. Em processos naturais, cada indivíduo pode apresentar uma mesma oscilação um pouco antes ou um pouco depois, possivelmente em intensidades diferentes. Como exemplo, pode-se citar o comportamento da altura de uma criança que, em geral, é acelerado na infância e desacelera próximo à idade adulta. Crianças diferentes possuem aceleração/ desaceleração em escalas e formas diferentes (variação de amplitude) que começam em instantes diferentes de sua idade (variação de fase).

Portanto, o primeiro passo em ADF é alinhar as curvas no eixo da variável independente, em geral o tempo. Essa técnica, chamada de registro, pode ser encontrada na literatura e uma boa referência é Ramsay (1998, 2001a). Após o registro, podemos então obter a média das estimativas – média estrutural[3]. A Figura 1.5 exibe as curvas da Figura 1.4 alinhadas por um tipo de registro. Nesse exemplo a média estrutural condiz mais com o processo verdadeiro do que a média seccional. Na verdade, a média seccional informa mal a respeito do processo que gerou as curvas, enquanto que a média estrutural é praticamente uma sombra da curva verdadeira, apresentando apenas um *bump*[4] mais moderado no primeiro quarto do intervalo de estimação. Nessa dissertação, a menos que mencionado diferente, sempre estará suposto que as curvas já estão alinhadas, de modo que propriedades importantes do comportamento de cada curva ocorram aproximadamente no mesmo local ao longo das mesmas.

[3] Essa é a denominação que alguns autores dão à média seccional avaliada após o registro.
[4] Porção da curva que apresenta uma oscilação mais forte, como vales e montanhas na superfície de um planeta.

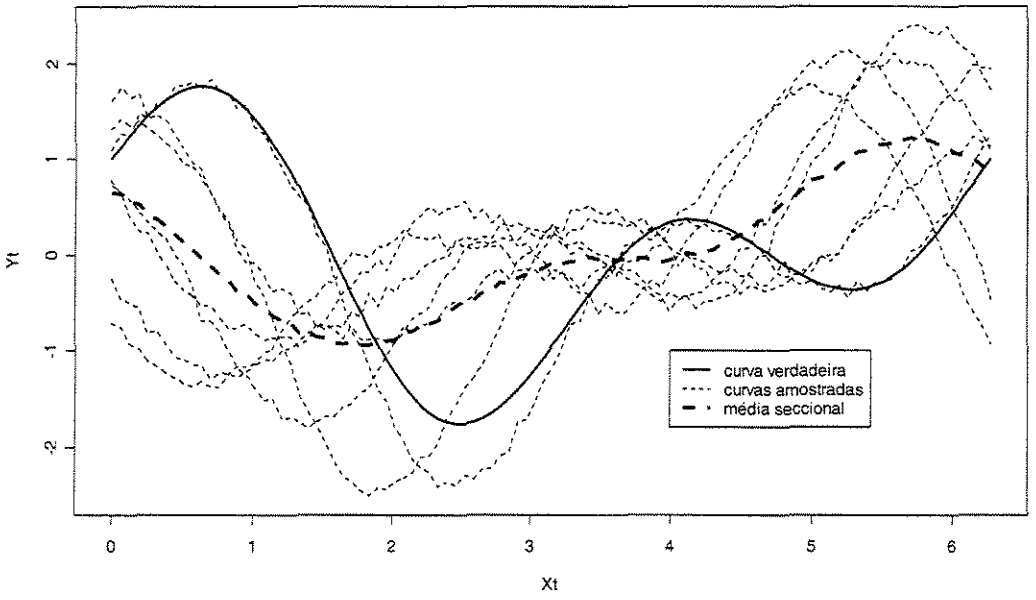


Figura 1.4: Curvas simuladas sob efeito de variação de fase e de amplitude adicionadas de ruído de variância pequena.

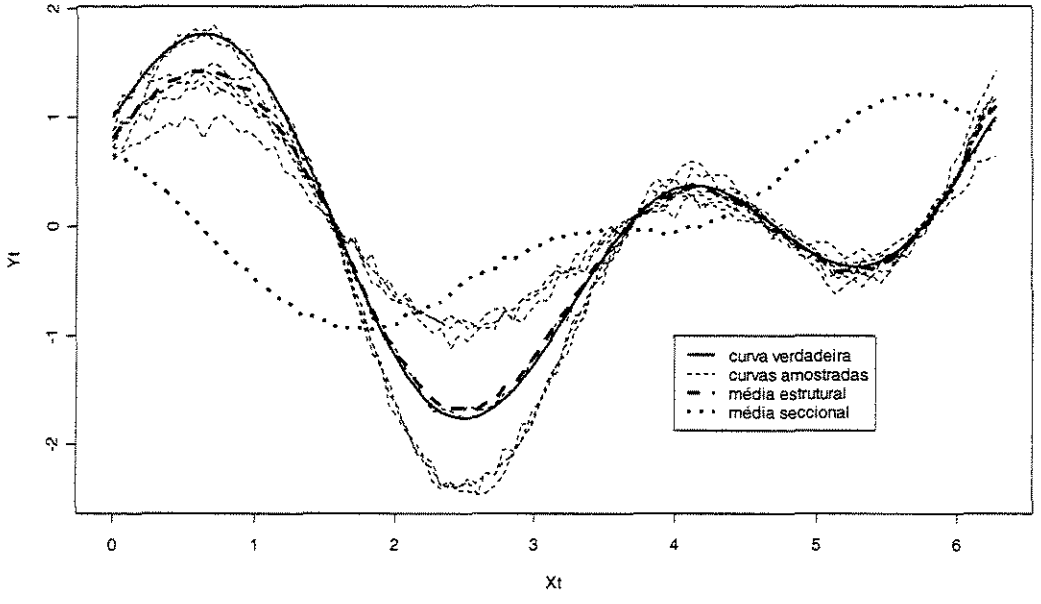


Figura 1.5: Curvas simuladas após registro.

1.2 Notação geral

Essa seção faz uma breve revisão da notação utilizada na maioria das técnicas existentes na literatura e que será adotada aqui.

Uma diferença preliminar é o conceito de informação funcional como sendo um dado único ao invés de um conjunto de dados. Esta seção abrange a notação necessária para facilitar o entendimento dessa idéia. Aqui também será vista a extensão para o caso funcional de conceitos de álgebra linear que serão úteis.

Por simplicidade de notação – pois fundamentalmente não há diferença – o subíndice i referir-se-á a cada unidade funcional ou vetor, o subíndice j indicará a posição nesses elementos e, a menos que indicado, não será reservado nenhum uso especial para subíndices k e l.

Como na literatura geral da Estatística, matrizes serão denotadas por maiúsculas em negrito, como $\mathbf{X}$. No entanto, vetores, escalares e funções serão simplesmente escritos como itálico. Isso quer dizer que x pode referir-se tanto a um escalar quanto a um vetor com elementos x_j ou a uma função com valores $x(t)$. Se x é um vetor ou uma função, seus elementos x_j ou $x(t)$ são geralmente escalares, mas podem se referir a um vetor ou função. diag(x) indicará uma matriz diagonal cujos elementos $d_{jj} = x_j$. Os subíndices $_{z,}$ e $_{,z}$, como em $\mathbf{X}_{z,}$ (ou $\mathbf{X}_{,z}$), indicam a matriz que se obtém quando se retiram as linhas (ou colunas) de $\mathbf{X}$ cujos equivalentes no vetor z são nulos. Por exemplo, se $z_1 = z_3 = 0$, $\mathbf{X}_{,z}$ é a matriz $\mathbf{X}$ sem a primeira e a terceira colunas.

Para indicar a transposição de um vetor ou matriz, será utilizado o superíndice ^{T}, como em $\mathbf{X}^T$. A notação para a v-ésima derivada de x será $D^v x$.

As implementações computacionais de rotinas serão citadas como em rotina().

1.2.1 Espaço de funções

Será visto nessa seção um pouco da notação para certas classes de funções às quais deseja-se que a função estimada pertença por satisfazer certo conjunto de restrições.

Há um espaço de funções onde se acredita que vivem todas as funções de interesse para serem consideradas respostas plausíveis aos problemas propostos. Ele é o conjunto de todas funções de quadrado integrável num certo intervalo $[a,b]$ e será denotado por $\mathcal{L}_2[a,b]$. Quando não for relevante mencionar o intervalo de integração, usar-se-á simplesmente $\mathcal{L}_2$.

O espaço $\mathcal{L}_2[a,b]$ é uma coleção muito rica de funções, mas para certos propósitos ele possui certos inconvenientes. Por exemplo, para x_1 e x_2 serem consideradas idênticas em $\mathcal{L}_2[a,b]$, basta que $\|x_1 - x_2\| = 0$, ou seja, x_1 e x_2 são iguais se elas diferem em um conjunto de Lebesgue de medida nula. $\|x\|$ indica a norma de x e será revisada na seção seguinte. Conseqüentemente, avaliar um elemento de $\mathcal{L}_2[a,b]$ em algum ponto de $[a,b]$ não é uma operação bem definida (Eubank, 1988), trazendo certos problemas no contexto em que estamos trabalhando. Como a maioria das necessidades que se seguirão envolve funções suaves sob certo sentido (Seção 1.3), podemos considerar funções que atendam certas condições de continuidade e diferenciabilidade.

Nesse caso uma classe de funções apropriada é $C^v[a,b]$, a classe de funções v vezes continuamente diferenciáveis em $[a,b]$. Em certos casos exigir que uma função f pertença a $C^v[a,b]$ será desnecessário, sendo suficiente que $D^v f$ seja de quadrado integrável em $[a,b]$. Seja $\mathcal{W}_2^v[a,b]$ o conjunto de todas funções v-1 vezes absolutamente continuamente diferenciáveis e com $D^v f \in \mathcal{L}_2[a,b]$, isto é, $\mathcal{W}_2^v[a,b]$ é o espaço de Sobolev de ordem v.

Consideraremos assim apenas funções em $\mathcal{W}_2^v$ ou em C^v, eliminando a ambigüidade de se calcular valores em certos elementos de interesse. Por exemplo, se x_1 e $x_2 \in C^0[a,b]$, $\|x_1 - x_2\| = 0 \Rightarrow x_1 \equiv x_2$. Além disso, o fato de que $\mathcal{W}_2^v$ e C^v são subespaços de $\mathcal{L}_2$ é útil, pois garante que eles herdem as propriedades de $\mathcal{L}_2$. Nesse caso, como trabalharemos com funções de base, precisamos de uma propriedade de $\mathcal{L}_2$ que garanta que toda função pode ser escrita como combinação linear de bases – há uma propriedade análoga para espaços euclidianos p-dimensionais. Já que estamos lidando

com espaços vetoriais infinito-dimensionais, forneceremos alguns resultados adicionais para funções de base na Seção 1.2.4.

1.2.2 Produto interno e norma

O conhecido produto interno da álgebra linear, denotado por $< x_1, x_2>$, representa a operação ${x_1}^T x_2$ e tem as seguintes propriedades:

- $<x_1, x_2> = <x_2, x_1>$ para todo x_1 e x_2;
- $<x, x> \geq 0$ para todo x e a desigualdade é estrita somente se $x \neq 0$;
- $<ax_1 + bx_2, x_3> = a<x_1, x_3> + b<x_2, x_3>$, para todo x_1, x_2 e x_3 vetores, a e b escalares.

Tais propriedades são denominadas simetria, não-negatividade e bi-linearidade e seu significado é imediato.

Da definição conclui-se que ${x_1}^T x_2 = \sum_j x_{1j} x_{2j}$. Note que essa não é a característica mais atraente do produto interno, pelo menos não mais que as três propriedades listadas anteriormente. É útil saber, por exemplo, que $x^T Z y$ é positiva definida se Z é uma matriz positiva definida cuja ordem garante a multiplicação (costuma-se denotar $Z > 0$ para Z positiva definida) e muitas outras propriedades aparecem em áreas como regressão e análise multivariada.

Para duas funções $x_1, x_2 \in \mathcal{L}_2$ o produto interno para funções é definido por $\int x_1(t) x_2(t)\, dt$ e também goza das propriedades enunciadas para o equivalente vetorial.

Para motivar, note que as propriedades do produto interno criam uma idéia interessante de associação entre o par de elementos no qual ele opera. Defina $\|x\|^2 = <x, x>$, de modo que $\|x\|$, a norma de x, é a raiz quadrada do produto interno de x consigo mesmo. Para vetores n-dimensionais a norma representa o tamanho do vetor enquanto que para funções é a conhecida norma L_2. Das propriedades do produto interno seguem as seguintes para a norma:

- $\|x\| \geq 0$, sendo a igualdade válida apenas para $x = 0$;
- $\|ax\| = |a| \, \|x\|$ para todo a escalar;
- $\|x_1 + x_2\| \leq \|x_1\| + \|x_2\|$;
- $|< x_1, x_2 >| \leq \|x_1\| \, \|x_2\| = (< x_1, x_1 >< x_2, x_2 >)^{1/2}$, a desigualdade de Cauchy-Schwarz.

Considere a divisão do produto interno do par (x_1, x_2) pelo produto das suas normas:

$$\frac{< x_1, x_2 >}{\| x_1 \| \| x_2 \|}.$$

Tal medida, invariante às escalas de x_1 e x_2, é uma medida de associação bastante útil. O caso especial vetorial é o conhecido coeficiente de correlação entre x_1 e x_2.

1.2.3 Estatísticas descritivas

Relacionamos algumas das definições básicas das estatísticas descritivas em ADF. Considere $x = (x_1, ..., x_n)$ uma amostra de tamanho n, o produto interno vetorial e 1_n como sendo um vetor de n uns.

- $\bar{x} = n^{-1} < x, 1_n >$ representa a média amostral e $\bar{x} 1_n$ é um vetor com todos elementos iguais $\bar{x}$;
- $s_{x_1 x_2} = n^{-1} < x_1 - \bar{x}_1 1_n, x_2 - \bar{x}_2 1_n >$ é a covariância entre o par (x_1, x_2);
- $s_x^2 = n^{-1} \| x - \bar{x} 1_n \|^2$ é a variância de x.

Outras definições como correlação e média ponderada seguem dessas três últimas.

A extensão para o caso funcional é de certa forma direta quando utilizando a notação de produto interno. Suponha que $x(t)$ varie num intervalo $[0, T]$ e defina

$$< x_1, x_2 > = \int_0^T x_1(t) x_2(t) dt,$$

com x_1, $x_2 \in \mathcal{L}_2$. As estatísticas definidas anteriormente continuam as mesmas apenas substituindo-se n por T e o vetor 1_n pela função $1(t)$ (função constante no intervalo $[0, T]$). Nesse caso, $\bar{x}$ representa o nível médio das funções e a variância é avaliada em torno desse nível médio. A covariância resume a dependência de cada curva ao longo dos valores de t. Essas estatísticas descritivas funcionais são exibidas abaixo:

- $\bar{x} = \dfrac{< x,1 >}{\|1\|^2}$;
- $s_{x_1x_2} = \dfrac{< x_1 - \bar{x}_1 1, x_2 - \bar{x}_2 1 >}{\|1\|^2}$;
- $s_x^2 = \dfrac{\|x - \bar{x}1\|^2}{\|1\|^2}$.

A Figura 1.6 exibe um exemplo de dados funcionais tirados de Ramsay and Silverman (1997). A Figura 1.7 mostra os gráficos da média e do desvio padrão funcionais. A Figura 1.8 mostra os gráficos em contorno e em perspectiva da correlação funcional, definida em função da covariância funcional. Note a região limitada pelas curvas de nível de altura 1 (do canto inferior esquerdo ao canto superior direito), correspondente análoga à diagonal principal de uma matriz de correlação usual. Observando o gráfico em perspectiva, vemos a mesma idéia na direção da parte anterior do gráfico indo para o fundo. Quando nos afastamos perpendicularmente dessa região, observamos quão rapidamente as correlações caem para dois valores de t afastados.

Há uma extensão do produto interno para o caso matricial quando x_1 e x_2 são vetores cujos elementos, por sua vez, são vetores passíveis de produto interno entre si (suponha que x_1 tem n_1 elementos e x_2 tem n_2 elementos). Dessa forma, $<x_1, x_2^T>$ define uma matriz $n_1 \times n_2$ cujas colunas contêm os valores $< x_{1j}, x_{2j} >$ e as propriedades de produto interno permanecem válidas feitas as devidas alterações.

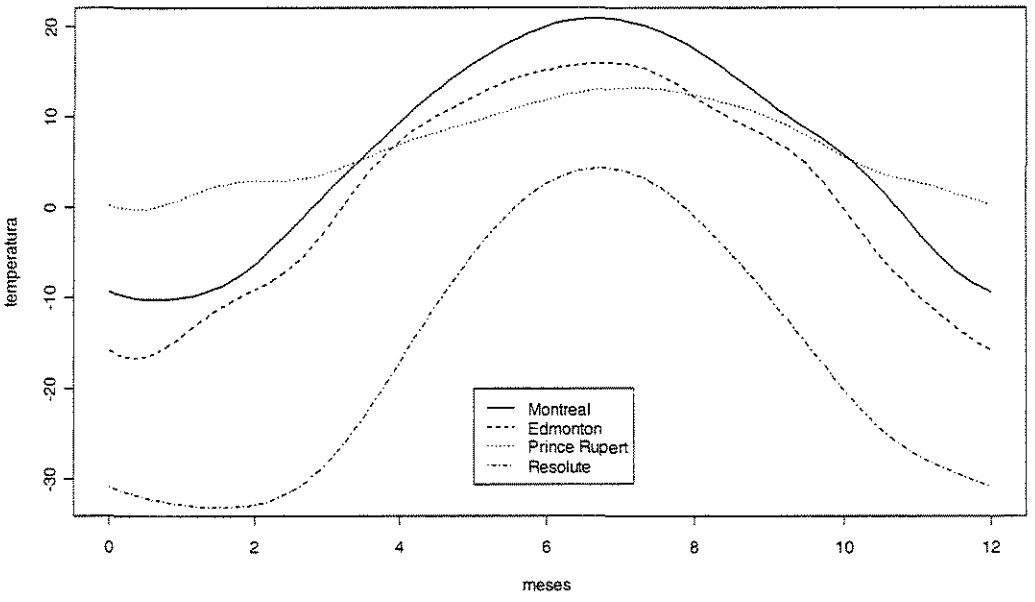


Figura 1.6: Temperatura em °C de $m = 4$ cidades ao longo de 12 meses.

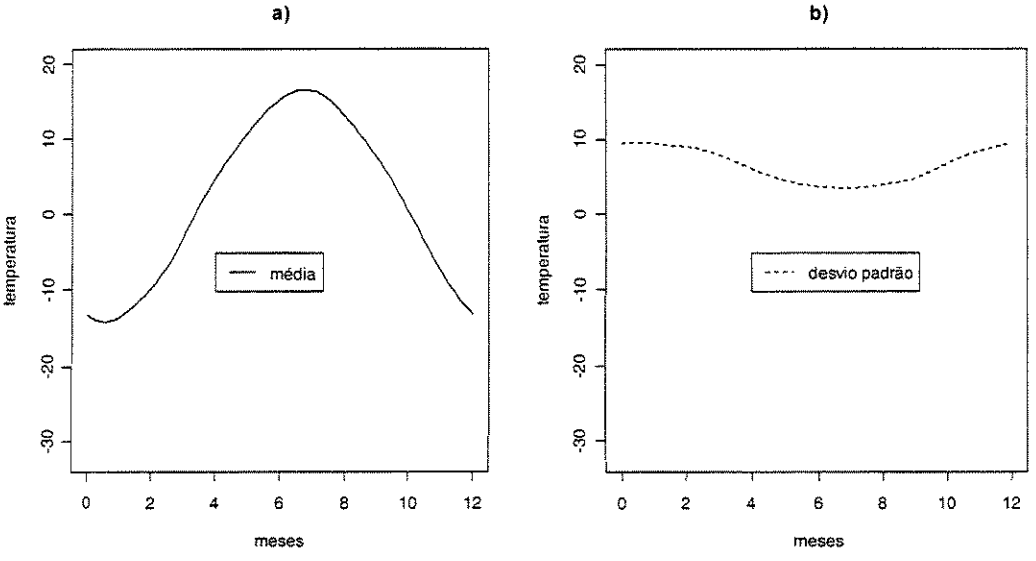


Figura 1.7: Estatísticas descritivas das curvas da Figura 1.6 na mesma escala:
a) Média funcional; b) Desvio padrão funcional.

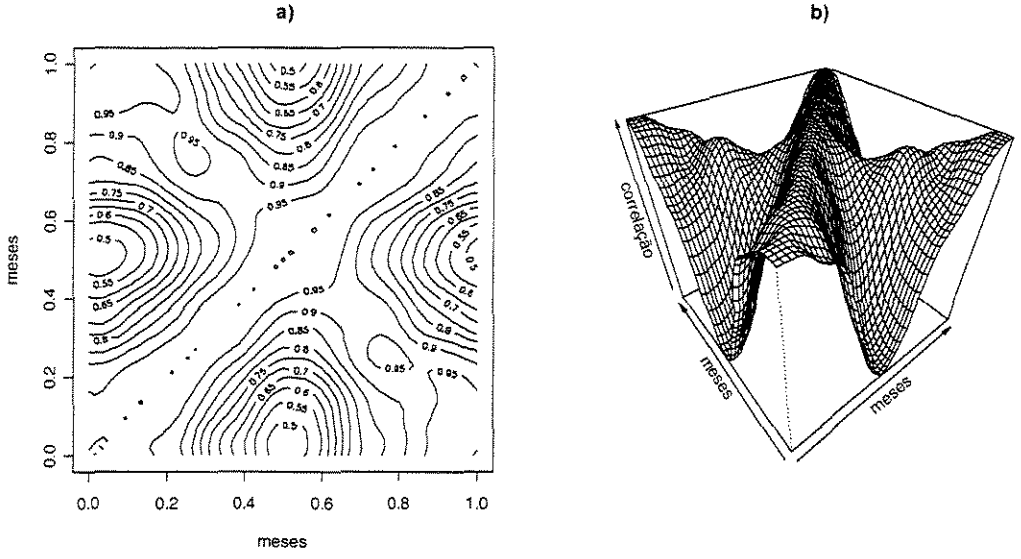


Figura 1.8: Correlação funcional das curvas da Figura 1.6: a) Gráfico em contorno; b) Gráfico em perspectiva.

1.2.4 Funções de base

Como mencionado na Seção 1.2.1, abaixo estão três definições tiradas de Eubank (1984):

- Duas funções $x_1, x_2 \in \mathcal{L}_2[a,b]$ são ditas ortogonais se $\langle x_1, x_2 \rangle = 0$. Para indicar ortogonalidade, será usada a notação $x_1 \perp x_2$.
- Uma seqüência de funções $\{x_i\}$ é dita ser ortonormal se as x_i são ortogonais duas a duas e $\|x_i\| = 1$, $\forall i$.
- Uma seqüência de funções $\{x_i\}$ é dita ser um sistema ortonormal completo (SOC) se $x \perp x_i$, $\forall i \Rightarrow x \equiv 0$.

Um exemplo de SOC são os polinômios de Legendre e há um método famoso para ortonormalizar uma seqüência de funções, conhecido como processo de ortonorma-

lização de Gram-Schmidt (Eubank, 1984). Uma proposição garante que todo SOC $\{x_i\}_{i=1}^{\infty}$ fornece uma base para $\mathcal{L}_2[a,b]$ no sentido de que toda função $f \in \mathcal{L}_2[a,b]$ tenha sua melhor aproximação λ-dimensional dada pela dupla $(\{x_i\}_{i=1}^{\infty}, \beta)$, em que $\beta = (\beta_j)$ com $\beta_j = \langle f, x_j \rangle$, ou seja,

$$\|f - \langle \beta, x \rangle_\lambda\| \leq \|f - \langle \alpha, x \rangle_\lambda\|, \forall \alpha \in \Re^\lambda,$$

em que $\langle \beta, x \rangle_\lambda$ considera apenas as λ primeiras posições de β e x. Além disso, $\|f - \langle \beta, x \rangle_\lambda\|^2 \xrightarrow[\lambda \to \infty]{} 0$.

Isso quer dizer que $\langle \beta, x \rangle_\lambda$ converge para f em norma e que f e $\langle \beta, x \rangle_\infty$ são funções idênticas quando vistas como elementos de $\mathcal{L}_2[a,b]$, embora isso não signifique convergência pontual em todo sentido, devido às propriedades da medida de Lebesgue. Contudo, isso será verdade para a maioria dos casos. Dessa proposição deriva o fato de que podemos representar f (em $\mathcal{L}_2[a,b]$) como uma combinação linear infinita das funções de base $\{x_i\}$. Os coeficientes β compõem a conhecida expansão de uma função em séries de Fourier generalizadas. Esse é um dos tipos de expansão permitidos nos pacotes computacionais citados na Seção 3.5 para representação dos dados como unidades funcionais suaves. No entanto, eles foram mencionados apenas por referência, pois o interesse aqui é na representação por B-*splines*. Antes disso revisaremos o conceito de *spline*.

1.2.5 Uma breve introdução à estimação via *splines*

Dentro dos vários tipos de estimadores que estão disponíveis na literatura hoje em dia, em particular em regressão não-paramétrica, *splines* em especial chamam atenção. As subseções que seguem têm o propósito de exibir uma visão geral dos mesmos auxiliada por exemplos.

1.2.5.1 Regressão polinomial

Na grande diversidade de conjuntos de dados analisados atualmente, nem sempre é possível obter um modelo linear que se ajuste bem aos mesmos. Permitindo uma flexibilidade à forma do modelo de regressão, uma consideração é optar pelo modelo

$$Y = f(x) + \text{erro},$$

para algum tipo de f. A abordagem clássica é considerar f sendo um polinômio de baixo grau, cujos coeficientes devem ser estimados por mínimos quadrados. Essa abordagem é largamente usada na prática e facilmente implementada sob o contexto de regressão múltipla (Green and Silverman, 1994).

No entanto, permitir que f seja um polinômio acaba gerando alguns inconvenientes. Talvez o mais grave deles seja o da influência global, onde uma observação individual pode exercer influência em regiões da curva alheias àquela da observação em questão. Além disso, permitir que os dados nos informem sobre o modelo que explique o processo subjacente é bem mais atrativo do que impô-lo aos dados.

1.2.5.2 Diferenças divididas

Existem várias maneiras de definir diferenças divididas (DD). Utilizaremos uma que é comum a de Boor (1978) e Eubank (1984) para a definição de B-*splines*.

Definição 1.1: A v-ésima diferença dividida de uma função f nos pontos $\xi_k, \ldots, \xi_{k+v}$ é o coeficiente de x^v no polinômio de grau $v+1$ que coincide com f nos pontos $\xi_k, \ldots, \xi_{k+v}$, no sentido da Definição 1.2, e é escrita como

$$[\xi_k, \ldots, \xi_{k+v}]f.$$

Serão listadas apenas as propriedades das diferenças divididas mais relevantes para o presente texto. A primeira propriedade é que $[\xi_k, \ldots, \xi_{k+v}]f$ é uma função simétrica de seus argumentos $\xi_k, \ldots, \xi_{k+v}$, isto é, depende apenas dos valores de $\xi_k, \ldots, \xi_{k+v}$ e não da

ordem em que eles aparecem na seqüência. A segunda propriedade é que se f é um polinômio de grau menor ou igual a v, então $[\xi_k, ..., \xi_{k+v}]f$ é constante como função de $\xi_k, ..., \xi_{k+v}$. Em particular, $[\xi_k, ..., \xi_{k+v}]f = 0$ para todo f polinômio de grau v.

Para entender melhor o significado dos v+1 índices, considere os casos particulares para 1 e 2 índices e a Definição 1.2:

$$[\xi_1]f = f(\xi_1);$$

$$[\xi_1, \xi_2]f = \begin{cases} \dfrac{f(\xi_1) - f(\xi_2)}{\xi_1 - \xi_2}, & \text{se } \xi_1 \neq \xi_2; \\ Df(\xi_1), & \text{se } \xi_1 = \xi_2. \end{cases}$$

Definição 1.2: Seja $\xi = (\xi_j)_{j=1..N}$ uma seqüência de pontos não necessariamente distintos. Diz-se que a função f concorda com a função g em ξ se for verdade que para todo ponto ξ que ocorra q vezes na seqüência $\xi_1,...,\xi_N$, f e g concordam q-uplamente em ξ, isto é,

$$D^{l-1}f(\xi) = D^{l-1}g(\xi), \quad l = 1, ..., q.$$

A definição da v-ésima diferença dividida é tomada nesse sentido quando alguns ou todos os pontos $\xi_k, ..., \xi_{k+v}$ coincidem.

Uma outra propriedade diz que a v-ésima diferença dividida é uma função contínua de seus v+1 argumentos para o caso em que $f \in C^v$. Acreditamos que as duas próximas propriedades sejam as mais importantes para os propósitos computacionais, merecendo ser destacadas:

- Se $f \in C^v$, então existe um ponto τ no menor intervalo que contém $\xi_k, ..., \xi_{k+v}$ de modo que

$$[\xi_k, ..., \xi_{k+v}]f = \frac{D^v f(\tau)}{v!};$$

- É útil notar que $[\xi_k, ..., \xi_{k+v}]f$ pode ser dada por

$$\begin{cases} \dfrac{D^{v} f(\xi_k)}{v!}, & \text{se } \xi_k = \ldots = \xi_{k+v} \text{ e } f \in C^{v}; \\ \dfrac{[\xi_k, \ldots, \xi_{r-1}, \xi_{r+1}, \ldots, \xi_{k+v}]f - [\xi_k, \ldots, \xi_{s-1}, \xi_{s+1}, \ldots, \xi_{k+v}]f}{\xi_s - \xi_r}, & \text{se } \xi_r, \xi_s \in \{\xi_k, \ldots, \xi_{k+v}\}, \ \xi_r \neq \xi_s. \end{cases}$$

Ao produto $(\xi_s - \xi_r)[\xi_k, \ldots, \xi_{k+v}]f$ dá-se o nome de diferenças divididas normalizadas de grau v. A escolha de ξ_r e ξ_s é arbitrária, decorrente da propriedade de simetria de $[\xi_k, \ldots, \xi_{k+v}]f$. Para a definição de B-*splines*, $s=k$ e $r=k+v$. As diferenças divididas podem ser eficientemente computadas seguindo a idéia da Tabela 1.1 (de Boor, 1978).

Tabela 1.1: Cálculo recursivo das diferenças divididas.

ξ	dados	primeira DD	segunda DD	...	(N-2)-ésima DD	(N-1)-ésima DD
ξ_1	$[\xi_1]f = f(\xi_1)$					
		$[\xi_1, \xi_2]f$				
ξ_2	$[\xi_2]f = f(\xi_2)$		$[\xi_1, \xi_2, \xi_3]f$			
		$[\xi_2, \xi_3]f$				
ξ_3	$[\xi_3]f = f(\xi_3)$				$[\xi_1, \ldots, \xi_{N-1}]f$	
						$[\xi_1, \ldots, \xi_N]f$
$\vdots$	$\vdots$				$[\xi_2, \ldots, \xi_N]f$	
ξ_{N-1}	$[\xi_{N-1}]f = f(\xi_{N-1})$		$[\xi_{N-2}, \xi_{N-1}, \xi_N]f$			
		$[\xi_{N-1}, \xi_N]f$				
ξ_N	$[\xi_N]f = f(\xi_N)$					

1.2.5.3 Penalização

Uma pergunta que se pode fazer a essa altura é: por que não procurar por uma função que erre o menos possível ao passar através do conjunto de dados, isto é, que torne mínima uma medida de bondade de ajuste, tal como $\Sigma[Y_j - f(x_j)]^2$? A Figura 1.9 traz duas possíveis respostas. Ambas preocupam-se apenas com a bondade de ajuste. Enquanto a estimativa da Figura 1.9a não goza de diferenciabilidade (requerida em muitas aplicações), a da Figura 1.9b não sofre desse problema, mas insiste em interpolar o erro.

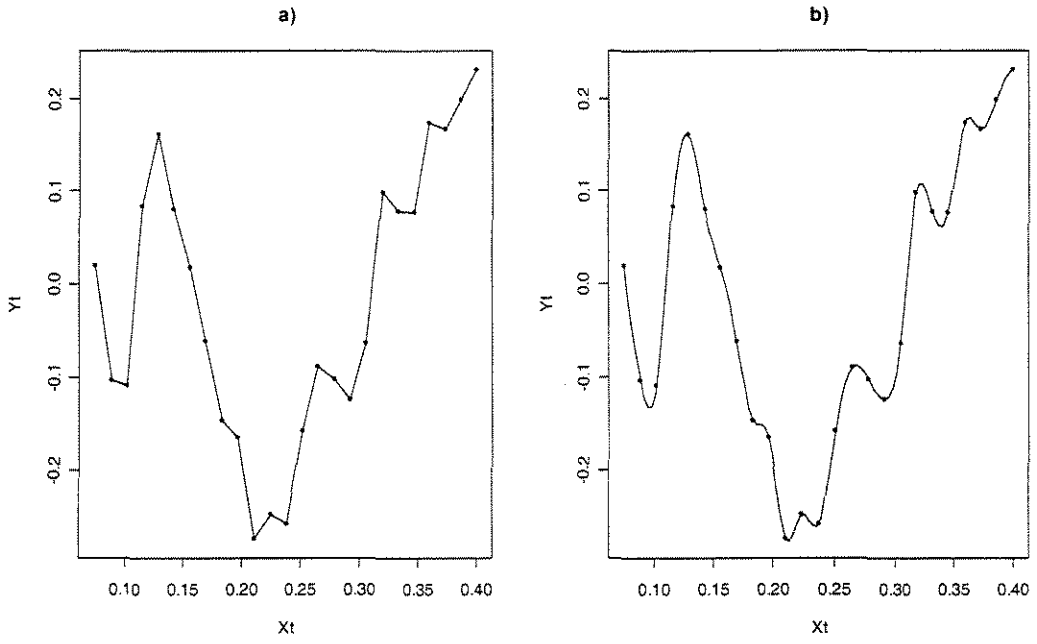


Figura 1.9: a) Interpolação *piecewise*[5] linear; b) Interpolação *piecewise*[5] cúbica.

Mas então que classe de modelos contém funções "apropriadas" para se trabalhar? O que surge agora não é a resposta, mas conduz a ela. A idéia de penalização é uma forma de impedir que a curva seja exageradamente sinuosa, de modo a interpolar os dados. Uma maneira de penalizar é considerar a quantidade de curvatura da função obtida e isso pode ser feito através de uma medida de suavidade como $\int[D^{v}f(x)]^2$, já conhecida e muito usada na literatura de regressão.

De posse dessas duas medidas – bondade de ajuste e suavidade, é possível construir uma classe de estimadores que goze de boas propriedades. Para isso, defina o seguinte critério de mínimos quadrados penalizados:

$$S(f) = \frac{1}{n}\sum_{j=1}^{n}\left[Y_j - f(x_j)\right]^2 + \lambda J_v, \qquad \lambda \geq 0, \tag{1.1}$$

em que

[5] Por partes; função obtida pela concatenação de polinômios.

$$J_v = \int_a^b \left[D^v f(x)\right]^2 dx .$$

Se a curva procurada pertence a $\mathcal{W}_2^v[a,b]$ para um certo $[a,b]$, então a função f que minimiza o critério (1.1) sobre todas as demais funções nessa classe é um *spline*. O parâmetro λ controla a influência que a penalização exerce sobre a estimação. De fato, se o seu valor é grande, estimadores com v-ésimas derivadas grandes são penalizados em favor daqueles com menores derivadas. Dessa forma, quanto maior o seu valor, mais suave será a função obtida. As figuras 1.10 e 1.11 dão exemplos de estimação para alguns valores de λ e para $v = 2$.

Considere primeiro a Figura 1.10. A curva contínua e mais ondulada foi estimada com um valor de λ muito pequeno, não penalizando, dessa forma, a sinuosidade da curva. Ela tenta interpolar os dados e, conseqüentemente, o ruído presente nos mesmos. A outra curva, traço-pontilhada, foi estimada usando-se um valor de λ muito alto, aproximando-se, assim, da curva pontilhada, que é a reta de regressão linear (suavidade máxima), com dois parâmetros: o intercepto e a inclinação.

Os casos extremos para o valor de λ são zero e infinito, no quais, respectivamente, ou os dados são legitimamente interpolados ou a estimativa torna-se aquela de mínimos quadrados usual da regressão linear[6]. Para ilustrar a sensibilidade da estimação ao valor de λ, observe a Figura 1.11. A curva verdadeira não foi exibida por não ser de interesse para essa ilustração. A curva pontilhada foi obtida fixando-se o valor de λ igual a 0.6 enquanto que a curva tracejada foi obtida usando-se o valor de λ que minimiza o critério GCV[7] (Craven and Wabba, 1979), nesse caso, 0.520447. Apesar do fato de que os valores são ligeiramente diferentes (cerca de oito centésimos), as estimativas exibem comportamentos visivelmente díspares em torno de $t = 0.09$ e de $t = 0.14$, mas muito similares em todo o resto do intervalo de estimação. Isso reforça a importância de um parâmetro que controla o peso relativo entre a quantidade de erro acumulada ao se usar uma certa representação para um dado conjunto de dados e a suavidade dessa representação.

[6] Linear para $v = 1$ ou polinomial para o caso mais geral.
[7] Validação Cruzada Generalizada.

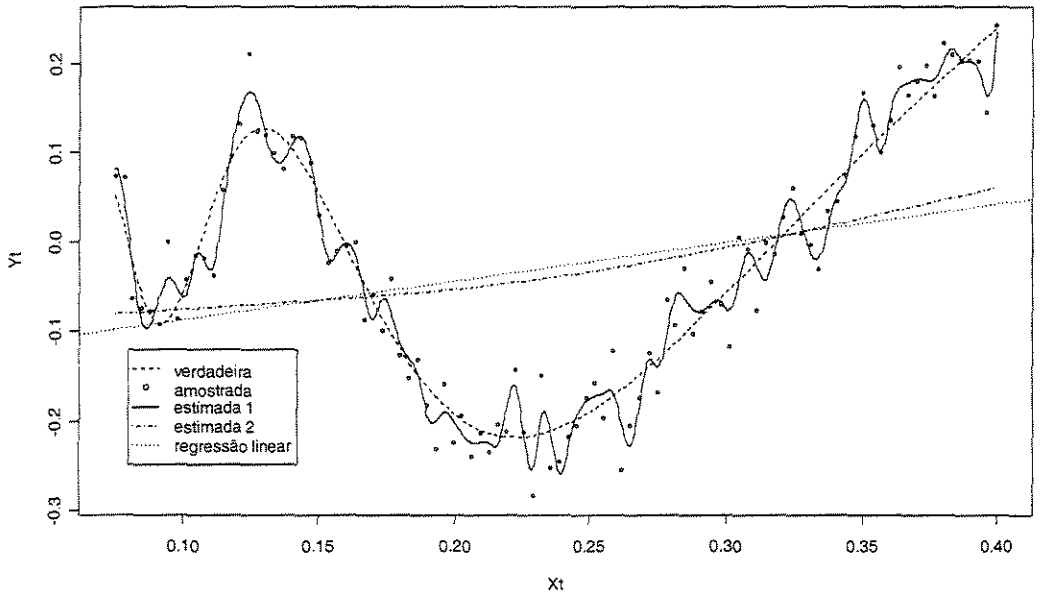


Figura 1.10: Exemplo de estimação para alguns valores de λ.

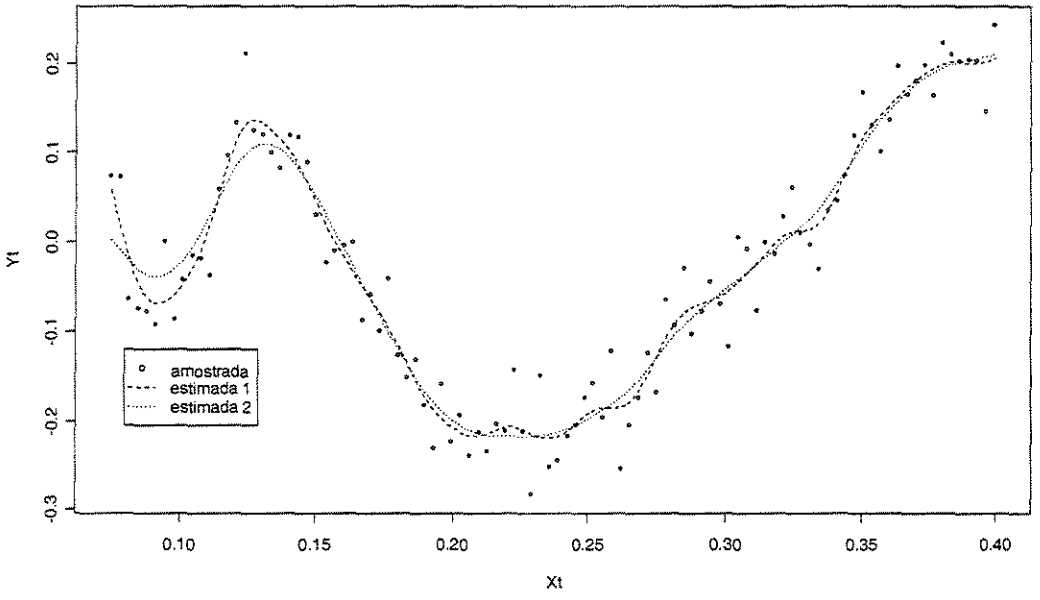


Figura 1.11: Sensibilidade da estimação ao valor de λ.

Em todas classes de estimadores estudadas até os dias de hoje, ainda discute-se como selecionar o parâmetro (ou conjunto de parâmetros) que controla a suavidade da curva. Uma boa referência é o Capítulo 2 de Eubank (1988), que dá uma boa visão de CV[8] (Stone, 1974, 1977), GCV e AIC[9] (Akaike, 1974), entre outros. No entanto, na técnica proposta nesse trabalho, apesar de λ ainda poder ser interpretado como parâmetro de suavização, ele é visto sob um ponto de vista diferente daquele da Equação (1.1). Logo, nos ateremos essencialmente ao tópico dessa seção, qual seja, *splines*.

1.2.5.4 *Splines, splines* cúbicos e suavização por *splines*

Já sabemos como surgem os *splines*. Agora veremos uma definição sua. Suponha que são dados um conjunto de números reais $x_1,...,x_n$ em um intervalo $[a,b]$ com $a \le x_1 < x_2 < ... < x_n \le b$. Uma função f definida em $[a,b]$ é um *spline* de grau $v+1$ se duas condições forem satisfeitas:

a) em cada intervalo de sub-índices consecutivos, f é um polinômio de grau $v+1$;

b) os pedaços polinomiais juntam-se nos pontos x_j de tal maneira que todas as suas v primeiras derivadas são contínuas em cada x_j e conseqüentemente o são em todo o intervalo (a,b) (Green and Silverman, 1994).

Os pontos x_i são denominados *knots*[10]. Para $x_0 = a$ e $x_{n+1} = b$, uma representação do *spline* é

$$f(x) = f_i(x),$$

com

$$f_i(x) = \beta_0 + \sum_{j=1}^{v} \beta_j (x - x_i)^j \ , \quad \text{se } x_i \le x \le x_{i+1}.$$

[8] Validação Cruzada.
[9] Critério de Informação de Akaike.
[10] Nós.

Na literatura constam inúmeras maneiras de se calcular os coeficientes β_j e esse não é objetivo aqui. Duas referências são Wegman and Wright (1983) e Green and Silverman (1994).

Um caso muito comum de *splines* usa $v = 2$, ou seja, penaliza-se a segunda derivada, ou aceleração da curva. A solução do critério em (1.1) compreende, então, os *splines* cúbicos ($2v - 1 = 3$; ver Anselone and Laurent, 1968).

Existem várias razões para optar por $v = 2$. Entre elas, o modelo físico é mais interessante de ser citado. Na verdade, acredita-se que a terminologia *spline* derive desse exemplo (Schoenberg, 1946). Suponha que temos uma peça de madeira fina e flexível restrita a passar por certos pontos de controle – os nós – mas cuja forma seja livre. Essa era a maneira que os desenhistas e engenheiros utilizavam para construção da superfície de navios ou de linhas férreas. Eles alteravam a posição dos nós e dessa forma delineavam o *spline* como queriam, às vezes modificando seu abaulamento em certas regiões, às vezes fazendo-o passar por certos pontos de interesse. O mais interessante é que esse modelo físico de *spline* minimiza[11] a energia de deflexão da tira de madeira sujeita à restrição de passar pelos nós. Como a energia de deflexão é um múltiplo daquela integral em (1.1), a solução do problema é um *spline* cúbico.

Adicionalmente, *splines* gozam de propriedades tais como existência, unicidade e flexibilidade. As referências citadas nessa seção cobrem bem essas vantagens. Além disso, *splines* cúbicos são os responsáveis por uma boa parte das aplicações de estimação não paramétrica que lida com funções suaves. Uma das poucas exceções é quando se deseja considerar derivadas de ordem maior das curvas, por essas trazerem informação adicional para o problema em questão, como é o caso quando se quer considerar a aceleração de uma curva em vez da própria ou mesmo em Análise Diferencial Principal.

Como pôde ser visto na Figura 1.9, interpolar os dados não é agradável quando eles estão sujeitos a erros de medição. Portanto, simplesmente utilizar um *spline* cúbico que interpole os dados não resolve o problema. Contudo, a solução de (1.1) para $v = 2$ é ainda um *spline* cúbico, na verdade, um *spline* cúbico natural. Um *spline* é chamado natural se, além das propriedades a e b anteriores, satisfizer também a seguinte:

[11] Aproximadamente; ver de Boor (1978).

c) f é um polinômio de grau v-1 fora do intervalo $[x_1, x_n]$. Isso é equivalente a definir $2v$ nós adicionais nos extremos a e b (metade em cada extremo) para satisfazer as condições de fronteira naturais.

No entanto, esse processo não pode ser chamado de interpolação. Esse tipo de função é chamado *smoothing spline*[12]. As curvas estimadas das figuras 1.10 e 1.11 são exemplos de suavização por *splines*. Como não é desejo fazer-se uma interpolação, os nós deixam de ser todo o conjunto de dados e passam a ser um conjunto mais restrito de pontos, doravante denominados ξ_j, j=1, ..., N. Como escolher os nós e onde posicioná-los é matéria de estudo até hoje. Nas seções que seguem, a menos que mencionado de forma diferente, *spline* indicará suavização por *spline* natural e quando o grau não for citado, estará implícito que se trata de uma curva cúbica.

1.2.6 A representação por B-*splines*

Há um teorema que garante a existência e unicidade de um *spline* que minimize (1.1) em $\mathcal{W}_2^v[a,b]$. Ele apóia-se em uma base apropriada para o espaço das *splines*. As seções 5.3.1 e 5.3.2 de Eubank (1984) exploram o teorema e algumas conseqüências dele. No entanto, o importante aqui é a representação da solução de (1.1) na forma dessa base apropriada – de fato, um outro teorema[13] garante que toda função polinomial por partes é combinação linear de B-*splines*[14].

Existem várias formas de construir essa base. A escolha da forma de representação tem a ver com o objetivo da aplicação, mas conhecemos que as B-*splines* gozam de propriedades vantajosas para o processo de estimação. Considere a seqüência não decrescente de nós ξ_j, j=1, ..., N. A k-ésima B-*spline* (normalizada) de grau v para a seqüência de nós ξ é dada por

$$B_{k,v,\xi}(x) = (\xi_{k+v} - \xi_k)[\xi_k, ..., \xi_{k+v}](\cdot - x)_+^v, \quad \forall x \in \Re, \tag{1.2}$$

[12] *Spline* suave, *spline* suavizadora ou suavização por *splines*.
[13] Curry and Schoenberg; ver de Boor (1978).
[14] Bases *splines*.

em que $(x)_+ = \max(x, 0)$. A notação $\cdot$ significa que a v-ésima diferença dividida da função $(\xi - x)_+^v$ de variáveis ξ e x considera x fixado e ξ livre para variar, ou seja, é uma função de ξ apenas. É claro que seu valor depende da escolha de x, de modo que varia de acordo com x, o mesmo ocorrendo para $B_{k,v,\xi}$. Abreviadamente, escreveremos B_k quando v e ξ estiverem subentendidos.

Algumas das propriedades que motivam o uso de B-*splines* são listadas abaixo:

i) Suporte compacto – $B_k(x) = 0$ se $x \notin [\xi_k, \xi_{k+v}]$. Isso decorre da segunda propriedade das diferenças divididas, conforme Seção 1.2.5.2, pois se $x \notin [\xi_k, \xi_{k+v}]$, então $f(x) = (\xi - x)_+^v$ é um polinômio de grau menor que ou igual a v. Dessa forma, apenas v+1 B-*splines* contêm um dado intervalo $[\xi_j,\xi_{j+1}]$ em seu suporte. Ou ainda, para a seqüência de nós ξ, apenas as v +1 B-*splines* B_{j-v}, B_{j-v+1},..., B_j são não nulas no intervalo $[\xi_j, \xi_{j+1}]$. A razão de escolher a normalização em (1.2) é que $\sum_j B_j(x) = 1$. Na verdade, basta somar sobre os índices compreendidos entre $r+v$ e s-1, isto é, $\sum_{j=r+v}^{s-1} B_j(x) = 1$.

ii) Não-negatividade – $B_k(x) > 0$ se $\xi_k < x < \xi_{k+v}$.

Essas propriedades dizem que a seqüência (B_k) gera uma partição da unidade (de Boor, 1978).

Considere a Figura 1.12. Um conjunto de 5 B-*splines* cúbicas (v = 3) foi desenhado. Observe que a imagem de todas elas está no intervalo [0,1]. Além disso, por qualquer ponto x escolhido, no máximo 4 (= v + 1) B-*splines* são positivas. Para x = 1, por exemplo, apenas três B-*splines* (B_2, B_3 e B_4) são positivas. Adicionalmente, elas somam 1 (= 0 + 25/36 + 10/36 + 1/36+ 0).

A escolha dos nós controla de certa forma a suavidade da curva. Por exemplo, se um nó não é repetido, v condições de continuidade são satisfeitas naquele nó. Para nós de multiplicidade l, $1 \leq l \leq v+1$, l condições de continuidade deixam de ser satisfeitas. Assim,

$$n.cond(x) = v + 1 - n.nós(x),$$

em que *n.cond*(x) e *n.nós*(x) indicam, respectivamente, o número de condições de continuidade e o número de nós em um ponto x.

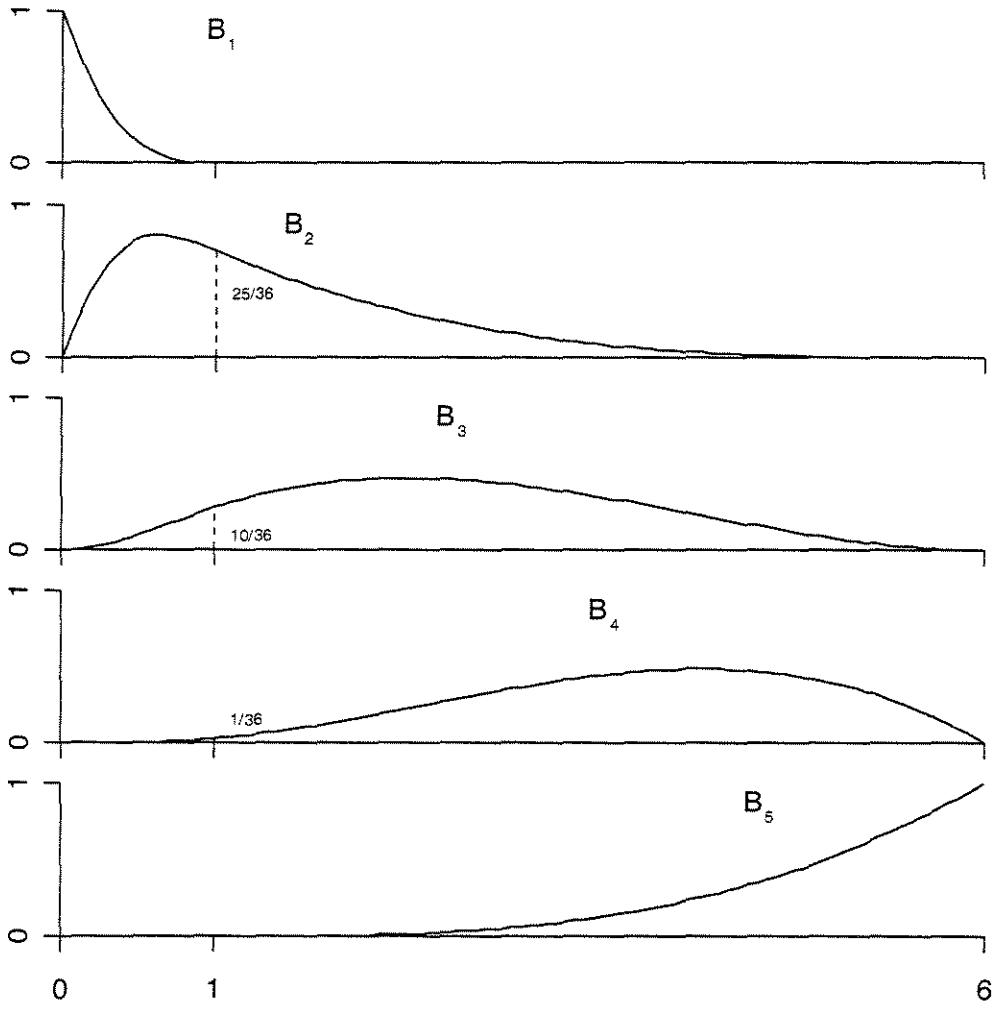


Figura 1.12: Ilustração das propriedades i e ii.

Para se calcular os coeficientes das B-*splines* existem formas mais eficientes e estáveis computacionalmente do que diretamente através das diferenças divididas normalizadas. De fato, uma definição recursiva de B-*splines* que não requer ajustes para

nós de multiplicidade maior que 1 e que não causa perda de significância durante os cálculos é exibida abaixo[15]:

$$B_{k,\nu+1}(x) = \frac{x-\xi_k}{\xi_{k+\nu}-\xi_k} B_{k,\nu}(x) + \frac{\xi_{k+\nu+1}-x}{\xi_{k+\nu+1}-\xi_{k+1}} B_{k+1,\nu}(x). \qquad (1.3)$$

A recursão é iniciada com $B_{k,1}(x)=\mathrm{I}(x\in[\xi_k,\xi_{k+1}))$, em que $\mathrm{I}(P)=1$ se P é uma propriedade verdadeira ou zero se P é falsa. Existem também versões recursivas para derivadas e integrais de B-*splines* que são estáveis e eficientes do ponto de vista computacional. No entanto, não é de interesse primário aqui. Uma referência é de Boor (1978). É importante notar que a matriz $\mathbf{B}_{n\times(\nu+1)}$ para estimação de Y definida para o conjunto de n pontos x_j dados é $(\nu+1)$-diagonal, isto é, $B_{jk} = 0$ se $|k-j|>\nu$. Nesse trabalho, como $\nu = 3$, $\mathbf{B}$ é uma matriz tetra-diagonal. Isso facilita operações como a inversão de matrizes, que podem ser realizadas a um custo computacional baixo. Essas são as informações necessárias para a compreensão das seções que seguem.

1.3 Representando dados funcionais como funções suaves

Já sabemos que a filosofia da análise de dados funcionais considera como unidade de estudo as funções de dados, isto é, cada indivíduo é representado por uma curva, ao invés de por apenas uma observação. As relações da literatura clássica que existem entre as observações individuais pontuais dão lugar às relações entre as curvas ou funções e são objeto de investigação. O termo funcional designa a estrutura dos dados e não a sua forma explícita, apesar de, na prática, os dados serem observados sob forma discreta. Cada observação x consiste de n pares (t_j, y_j) em que y é o registro referente à observação de x no tempo t. Dessa forma, o conjunto de dados consiste de n_i pares (t_{ij}, y_{ij}), $j = 1, ..., n_i$. A explicação por considerar o problema como funcional ao invés de multivariado reside na suposição de que x_i é alguma função bem comportada.

[15] A demonstração da equivalência entre essa forma e (1.2) pode ser encontrada em de Boor (1978).

Existem algumas técnicas na literatura para converter dados brutos em sua forma funcional. Portanto, como os dados podem ser considerados como soma da forma funcional com algum ruído branco, a representação funcional envolve suavização, que é aplicável não apenas às curvas em si como também às suas derivadas.

A suposição mais geral é de que os dados variam num intervalo limitado $\mathcal{T}$ e podem ser representados como funções de t contínuas e suaves nesse intervalo.

Dentre as técnicas de suavização disponíveis, a suavização por kernel possui características computacionais relativamente boas. O método está relacionado ao parâmetro de largura da banda[16] que envolve cada ponto da curva a ser estimado. Uma resposta à instabilidade do método nas regiões extremas dos intervalos é a suavização por *splines*.

1.3.1 Suavização por *splines* e penalização

Uma opção à suavização por kernel é a abordagem de penalização da curvatura[17] da função a ser estimada. Sabe-se que as expansões em funções de base dão bons resultados se estas têm as mesmas características que o processo que gera os dados. Por exemplo, bases de Fourier são as mais indicadas em problemas cujos dados possuem comportamento periódico. No entanto, tal abordagem não tem controle eficaz sobre o grau de suavização e o custo computacional pode ser alto se as bases não gozam da propriedade de ortogonalidade nem da de efeito local.

O método de suavização por *splines* garante que a curva estimada seja um bom ajuste para os dados, por exemplo, em termos da soma de quadrados residual, e não permite que ela seja demasiadamente ondulada, como visto na Seção 1.2.5.3. Os dois extremos dessas exigências são o super ajuste, onde a curva interpola os pontos amostrais e possui alta variância e o caso onde a variância é reduzida ao máximo, inflando dessa forma o erro de ajuste. A idéia de sacrificar um pouco de viés para

[16] *Bandwidth.*
[17] *Roughness.*

reduzir a variância é o que está por trás da suposição de suavidade das curvas a serem estimadas.

A abordagem de funções de base utiliza x como uma combinação linear de um pequeno número K de bases e tenta minimizar o erro residual sujeito a restrições de curvatura. É essa a tal penalização. Uma medida usual de curvatura é J_2 (Equação (1.1)). Curvas cuja variação é alta possuem alto valor de $J_2(x)$ porque suas segundas derivadas são altas no intervalo considerado. Assim, a soma de quadrados dos resíduos penalizada da Equação (1.1) para $v = 2$ pode ser expressa por

$$\mathrm{SQRP}_\lambda(x|\, y) = \Sigma_j[y_j - x(t_j)]^2 + \lambda J_2(x).$$

O objetivo é minimizar $\mathrm{SQRP}_\lambda(x)$ para o espaço de funções x. O parâmetro de suavização λ controla o equilíbrio entre a soma de quadrados residual e a variação da curva.

Já sabemos que a curva resultante é um *spline* cúbico cujos nós são os pontos amostrais t_j.

No entanto, quando as curvas não estão alinhadas, até mesmo as análises mais comuns não podem ser realizadas de maneira coerente. O problema é que o comportamento de cada curva ao longo do tempo (ou de outra escala ao longo da qual as observações variem) poderá ser diferente para indivíduos diferentes e medidas comuns como correlação, por exemplo, já não podem ser interpretadas corretamente. Existem métodos de registro para lidar com vários tipos de alinhamento e, portanto, não será o enfoque considerado aqui.

1.4 Breve revisão das técnicas para ADF

A próxima seção dá uma breve introdução a uma técnica cuja extensão funcional objetiva analisar problemas referentes à análise de dados funcionais – a Análise de Componentes Principais (ACP). Sua característica principal é tomar um pequeno número de modos dominantes de variação e analisar os dados sob esse ponto de vista. Além dessa técnica há também a Análise de Correlação Canônica (ACC) – que tenta explicar como duas ou mais curvas variam conjuntamente ou dependem uma da outra –

e Análise Diferencial Principal (ADP) – que tenta usar a informação presente nas derivadas das curvas para estudar os componentes de variação. Elas podem ser usadas isoladamente ou complementando uma à outra. Maiores detalhes podem ser encontrados em Ramsay and Silverman (1997).

1.4.1 Análise de componentes principais para dados funcionais

Acreditamos que Análise de Componentes Principais para dados funcionais seja a técnica em ADF mais importante a considerar. É possível que se deseje realizar uma análise exploratória dos dados após o registro das curvas para se ver propriedades que caracterizem as funções usadas (por exemplo, pode-se considerar a natureza senoidal das curvas de temperaturas ou quantos tipos de curvas devem ser usadas).

A ACP fornece uma maneira mais informativa de observar a estrutura de covariância entre as funções (tão complexa quanto ou até mais do que a técnica equivalente em Análise Multivariada) do que o exame direto da função de variância-covariância. A extensão da ACP multivariada para o caso funcional começa substituindo o índice discreto j pelo seu equivalente contínuo t e, analogamente, trocando os somatórios pela integral sobre todo t a fim de definir as combinações lineares

$$f_i = \int \beta(t) x_i(t) dt = <\beta, x_i>,$$

em que o vetor de pesos β_i torna-se a função peso $\beta(t)$.

Da mesma forma como na ACP multivariada, o primeiro passo da ACP funcional busca a função ξ_1 que maximiza

$$\frac{1}{n}\sum_i f_{i1}^2 = \frac{1}{n}\sum_i < \xi_1, x_i >$$

sujeita à restrição $\|\xi_1\|^2 = 1$, em que $\|\xi_1\|^2$ representa a norma quadrada $\int[\xi_1(t)]^2 dt$ da função ξ_1. A interpretação para cada função ξ_k é análoga àquela dada para a equivalente multivariada da ACP.

Como em ACP multivariada, cada função peso ξ_k é também restrita à ortogonalidade em relação às demais funções peso em cada passo do procedimento, isto é, $<\xi_k, \xi_l> = 0, \forall\ l < k$. Cada função peso tenta então resumir a maior porção de variação presente nas curvas e ainda ser ortogonal a todas as outras definidas nos passos anteriores. Por suas características gráficas que lembram muito o comportamento físico das ondas, as componentes também são conhecidas como harmônicos.

Uma idéia equivalente de ACP funcional é pensar em obter um conjunto de K funções de base ortonormais ξ_k de modo que a expansão de cada curva em termos dessas funções de base seja o mais aproximada possível. Uma vez que as funções de base são ortogonais, a expansão será da forma

$$\hat{x}_i(t) = \sum_{k=1}^{K} f_{ik}\xi_k(t),$$

com $f_{ik} = <x_i, \xi_k>$. Um possível critério de ajuste para uma dada curva é tentar minimizar a soma de quadrados dos erros, aqui representada por

$$\sum_{i=1}^{n} \| x_i - \hat{x}_i \|^2,$$

em que

$$\| x_i - \hat{x}_i \|^2 = \int [x_i(t) - \hat{x}_i(t)]^2 dt.$$

O problema resume-se então à escolha das bases que minimizam esse critério e tem como resposta as mesmas funções que maximizam as componentes de variância definidas no início dessa seção. Essas funções são denominadas empíricas, pois são determinadas pelos dados que elas devem representar.

Não é tão imediato interpretar as componentes (ou harmônicos) para a maioria dos problemas de ACP funcional e pode-se recorrer a métodos gráficos. É útil, por exemplo, considerar o gráfico da função média estrutural definida na Seção 2.2.1 e as funções obtidas ao adicionar e subtrair um múltiplo coerente do harmônico em questão (gráfico das componentes como perturbação da média). Esse método tenta esclarecer o papel que cada harmônico tem na representação dos dados. Ele pode exibir mudanças

globais e mesmo locais, translações[18] e outros comportamentos. Uma possível escolha para que múltiplo C usar é tomar

$$C^2 = T^{-1} \| \hat{\mu} - \overline{\hat{\mu}} \|^2,$$

em que

$$\overline{\hat{\mu}} = T^{-1} \int \hat{\mu}(t) dt,$$

e traçar $\hat{\mu}$ (a média estrutural) e $\hat{\mu} \pm 0.2C\xi_i$. A constante 0.2 pode ser substituída de maneira subjetiva por uma outra em função do problema, mas deve ser a mesma para todos os harmônicos ξ_i (ver Ramsay and Silverman (1997)).

Um outro método gráfico é o exame dos escores f_{ij} de cada curva em cada harmônico. Ele exibe dois indivíduos relativamente semelhantes mais próximos um do outro do que daqueles que são menos semelhantes. Se o número de harmônicos necessário para resumir uma boa parcela da variação dos dados é grande a análise visual torna-se ineficaz (lembre-se que cada componente equivale a uma dimensão do gráfico e que gráficos tridimensionais já são de análise trabalhosa), mas geralmente, pelo visto em alguns poucos exemplos reais, os dois primeiros harmônicos retêm mais de 70% da variação e são suficientes para a análise. A Figura 1.13 exibe os dois primeiros harmônicos para as curvas da Figura 1.6 adicionados/substraídos de um múltiplo C.

Após toda essa quantidade de informação sobre ADF não se deve deixar de lado duas características presentes em problemas que envolvem dados funcionais: replicação e regularidade, ambas referentes ao uso de informação referente a múltiplos valores de dados para identificar padrões. Enquanto a primeira fornece uma idéia do que ocorre para cada indivíduo em certa região, a segunda ilustra o comportamento ao longo da escala sobre a qual observam-se os dados (em geral t, o tempo).

[18] *Shifts*.

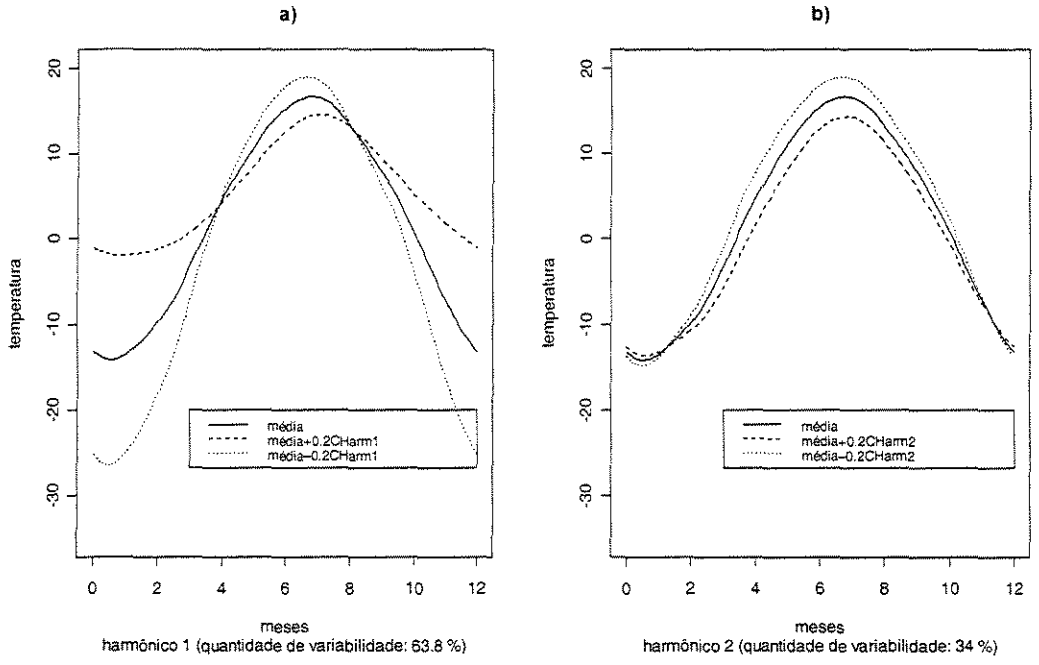


Figura 1.13: Primeiro (a) e segundo (b) harmônicos para as curvas da Figura 1.6.

1.5 Objetivos

O objetivo inicial desse projeto foi a implementação de uma nova metodologia para Análise de Dados Funcionais com Penalização Estocástica. Os dados consistem de um conjunto de curvas supostamente provenientes de um mesmo processo. Dessa forma, as curvas costumam apresentar comportamento similar cujas nuances ocorrem aproximadamente no mesmo lugar ao longo das curvas. Individualmente essas curvas podem ser consideradas como combinação linear de funções apropriadamente escolhidas e que geram uma base para o espaço de funções para as curvas em questão.

Questões como custo computacional e plausibilidade das suposições do modelo em geral foram de interesse primário. As simulações iniciais foram feitas através do software R (http://cran.r-project.org) e posteriormente, por motivos de comparação e velocidade, foi realizada sua implementação em Ox (http://www.nuff.ox.ac.uk/users/Doornik). Através delas, utilizou-se uma variante do algoritmo SEM para estimação dos parâmetros do modelo.

Em todas as simulações esteve implícito que o argumento sobre o qual são feitas as medidas, t, é unidimensional e que as funções são registradas e alinhadas em algum sentido, de modo que propriedades importantes de cada curva ocorram aproximadamente sobre os mesmos valores de t.

O Capítulo 2 dá uma introdução à ADF via penalização estocástica, mostrando as idéias preliminares, o modelo proposto e algumas formas de resumir as curvas ao longo das iterações. O Capítulo 3 exibe os resultados da análise da eficiência do algoritmo proposto através de simulações. Ele considera também as medidas resumo das curvas e um exemplo de problema onde a técnica de registro é necessária antes de aplicação da metodologia proposta. O Capítulo 4 finaliza a dissertação com as conclusões e considerações finais, fornecendo algumas sugestões para trabalhos futuros. No Apêndice A constam as derivadas da função de log-verossimilhança da Equação (2.7) – e sua versão modificada através da transformação da Equação (2.21) – necessárias para os métodos numéricos de maximização utilizados nos algoritmos propostos. O Apêndice B exibe alguns casos particulares em que as medidas resumo do Capítulo 2 são consideradas equivalentes. No Apêndice C encontram-se as listagens das implementações elaboradas em linguagem corrente dos programas R, Ox, Matlab e Maple citadas ao longo da dissertação.

2 *ADF via Penalização Estocástica*

2.1 Introdução

As principais técnicas em ADF são Análise de Componentes Principais (a mais usada), Análise de Correlação Canônica e Análise Diferencial Principal. A filosofia desse método, no entanto, é diferente das técnicas atualmente existentes. Essa seção define o modelo proposto e percorre as idéias e problemas que surgiram ao longo da dissertação.

2.1.1 O modelo proposto

Considere uma coleção de m curvas, cada uma representando um processo medido para certo indivíduo. Cada curva da amostra constitui-se dos pontos x_{ij}, para $j = 1, ..., n_i$. Seja Y_{ij} o resultado medido de acordo com o i-ésimo indivíduo no ponto x_{ij}. Assuma que

$$Y_{ij} = Y(x_{ij}) = \sum_{k=1}^{K} Z_{ki}\beta_{ki}\mathbf{B}_k(x_{ij}) + \varepsilon_{ij}, \tag{2.1}$$

com $\varepsilon_{ij} \sim \mathcal{N}(0, \sigma^2)$ e $\mathbf{B}_k$ denotam as funções B-*splines* cúbicas,

$$\mathrm{P}(Z_{1i} = z_{1i}, ..., Z_{Ki} = z_{Ki}) = \prod_{k=1}^{K} \theta_{ki}^{z_{ki}} (1-\theta_{ki})^{1-z_{ki}}, \tag{2.2}$$

$Z_{ki} \in \{0, 1\}$, $\mathrm{P}(Z_{ki} = 1) = \theta_{ki}$, para todo $k = 1, ..., K$ e $i = 1, ..., m$. O valor de K é então o número de funções de base utilizadas para representar os dados.

Defina $Y_i = (Y_{i1}, ..., Y_{in_i})^T$, $Z_i = (Z_{1i}, ..., Z_{Ki})^T$, $\beta_i = (\beta_{1i}, ..., \beta_{Ki})^T$, $Z_k = (Z_{k1}, ..., Z_{km})$ e $\beta_k = (\beta_{k1}, ..., \beta_{km})$, ou seja, quando quisermos nos referir aos elementos de Z_{ki} ou de β_{ki} mantendo k fixo e variando o indivíduo usaremos Z_k e β_k e quando estivermos nos referindo ao indivíduo e variando a base usaremos Z_i e β_i.

Conseqüentemente, a densidade condicional de $(Y_i|Z_i)$ é dada por

$$f(y_i \mid z_i) = \varphi\left(\frac{y_i - \sum_{k=1}^{K} z_{ki}\beta_{ki}\mathbf{B}_k(x_i)}{\sigma}\right), \tag{2.3}$$

em que $\varphi(\cdot)$ é a densidade de uma normal padrão. Se tomarmos $P(Z_i = z_i) = p(z_i)$, temos que

$$\begin{aligned} f(y_i, z_i) &= f(y_i \mid z_i)p(z_i) \\ &\propto \sigma^{-\frac{n_i}{2}} \exp\left\{-\frac{1}{2\sigma^2}\left\|y_i - \sum_{k=1}^{K} z_{ki}\beta_{ki}\mathbf{B}_k(x_i)\right\|^2\right. \\ &\quad \left. + \sum_{k=1}^{K}\left[z_{ki}\log(\theta_{ki}) + (1 - z_{ki})\log(1-\theta_{ki})\right]\right\}. \end{aligned} \tag{2.4}$$

Assim, a densidade conjunta de (Y,Z) pode ser escrita como

$$\begin{aligned} l(y, z \mid \sigma^2, \beta, \theta) &= \log f(y, z \mid \sigma^2, \beta, \theta) = \sum_{i=1}^{m} \log f(y_i, z_i) \\ &\propto \sum_{i=1}^{m}\left\{-n_i \log\sigma^2 - \frac{1}{2\sigma^2}\left\|y_i - \sum_{k=1}^{K} z_{ki}\beta_{ki}\mathbf{B}_k(x_i)\right\|^2\right. \\ &\quad \left. + \sum_{k=1}^{K}\left[z_{ki}\log\left(\frac{\theta_{ki}}{1-\theta_{ki}}\right) + \log(1-\theta_{ki})\right]\right\}. \end{aligned} \tag{2.5}$$

Maximizar a verossimilhança completa $l(\sigma^2, \beta, \theta) = l(y, z|\sigma^2, \beta, \theta)$ é equivalente a resolver um problema de mínimos quadrados penalizados associado à Equação (2.5) (Green and Silverman, 1994). As variáveis aleatórias Z determinam quais bases entrarão no modelo e conseqüentemente a dimensão do espaço aproximativo para cada curva i.

Contudo, este tipo de parametrização faz com que o problema de mínimos quadrados penalizados tenha solução limite com $z_{ki} = 1$ e $\theta_{ki} \longrightarrow 1$ quando $K \longrightarrow \infty$. Dessa forma, foi utilizada a transformação

$$\theta_{ki} = 1 - e^{-\lambda|\beta_{ki}|}, \tag{2.6}$$

em que λ será considerado por enquanto um parâmetro pré-fixado, pois estimar λ que maximize (2.5) ainda força a entrada de todas as bases no modelo[19]. Portanto, a Equação (2.5) pode ser reescrita na forma

$$\begin{aligned} l\left(y, z \mid \sigma^2, \beta, \theta\right) &= \sum_{i=1}^{m} \log f(y_i, z_i) \\ &\propto \sum_{i=1}^{m} \left\{ -n_i \log\sigma^2 - \frac{1}{2\sigma^2} \left\| y_i - \sum_{k=1}^{K} z_{ki} \beta_{ki} \mathbf{B}_k(x_i) \right\|^2 \right. \\ &\quad \left. + \sum_{k=1}^{K} \left[z_{ki} \log\left(\frac{1 - e^{-\lambda|\beta_{ki}|}}{e^{-\lambda|\beta_{ki}|}} \right) \right] - \lambda \sum_{k=1}^{K} |\beta_{ki}| \right\}. \end{aligned} \tag{2.7}$$

No entanto, as variáveis Z_{ki} são não observáveis e para fazer uso do algoritmo EM[20] precisamos da distribuição condicional de $(Z|Y)$, ou seja,

$$f_\psi(z, y) = \frac{f_\psi(y \mid z)}{f(y)} \propto f_\psi(y \mid z), \tag{2.8}$$

com $\psi = (\sigma^2, \beta, \theta)$. Como não temos a condicional completa, pois é extremamente custoso computacionalmente obter a constante (nos parâmetros) $f(y)$, podemos fazer uso do algoritmo SEM[21] simulando Z_{ki} da distribuição condicional (2.8) via algoritmo Metropolis-Hastings. No entanto, esse procedimento ainda é extremamente custoso porque precisamos executar uma simulação Metropolis-Hastings para cada iteração do algoritmo. Dessa forma, passamos a usar uma variante do algoritmo SEM, descrita na Seção 2.3. Sem perda de generalidade, para as próximas seções, suponha $n_i = n$, $i = 1, ..., m$.

[19] Essa transformação é útil porque vincula o peso que uma base tem no modelo e a probabilidade de ela ser selecionada através da variável Z.
[20] Esperança-Maximização.
[21] EM estocástico.

2.2 Idéias preliminares

2.2.1 Estimação da média estrutural

No início do projeto foram utilizadas duas formas de representação dos dados para estimação da média estrutural. A primeira, que chamamos de método de pós-suavização, é mais simples, lida com os dados brutos (observações discretas), obtém a média estrutural e só após isso fornece uma estimativa suave da mesma. A segunda forma, denominada método de pré-suavização, fornece uma representação suave[22] de cada curva como dado inicial e a partir daí obtém a média estrutural do conjunto. Suavizar ou não a média estrutural resultante é praticamente indiferente para a maioria dos casos onde a pré-suavização é usada, de acordo com os resultados alcançados, mas por ser de custo computacional insignificante nessa etapa do processo, proceder-se-á dessa maneira.

Para avaliar qual procedimento é mais eficaz em termos de soma de quadrados de erros foram gerados dois tipos de curvas. O primeiro, de comportamento periódico, possui oscilações mais brandas enquanto que o segundo possui uma única oscilação forte. Embora não tenha surtido muito efeito suavizar a estimativa da média estrutural para a pré-suavização, os resultados são exibidos para as três idéias.

A Figura 2.1 exibe o gráfico de uma curva simulada e uma amostra realizada da mesma. Como sabemos, a interpolação dos pontos fornece uma representação grosseira do processo (curva) em consideração. Na Figura 2.2 percebe-se a diferença entre o dado bruto (não suavizado) e o dado na sua forma funcional. Há $m = 3$ curvas amostradas do processo verdadeiro e suavizadas, cada uma acompanhando o padrão da curva verdadeira.

[22] Dado na forma funcional.

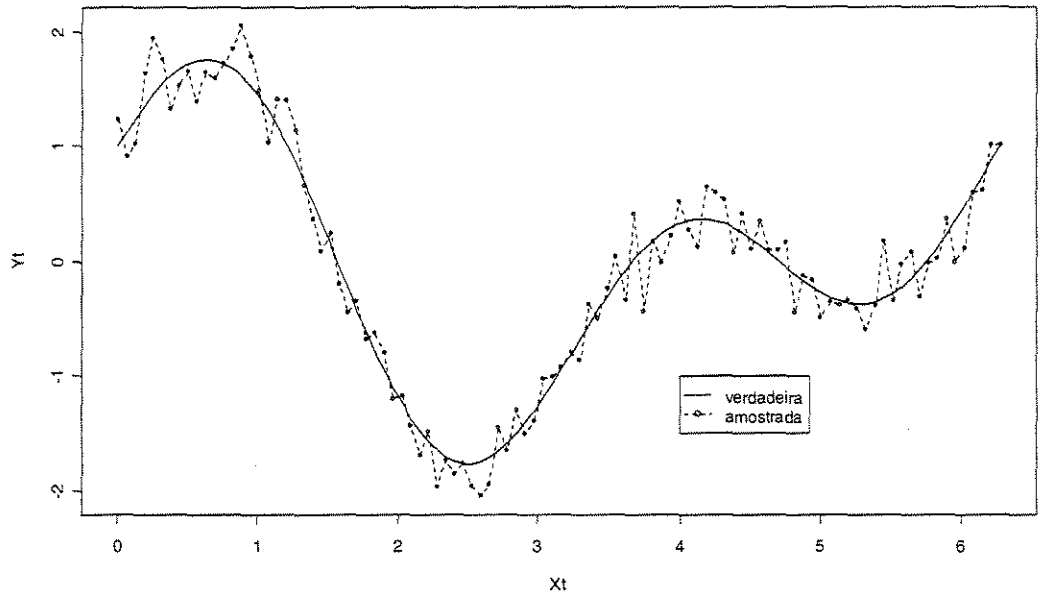


Figura 2.1: Amostra de uma curva simulada adicionada de ruído de baixa variância e sua interpolação (sem suavização).

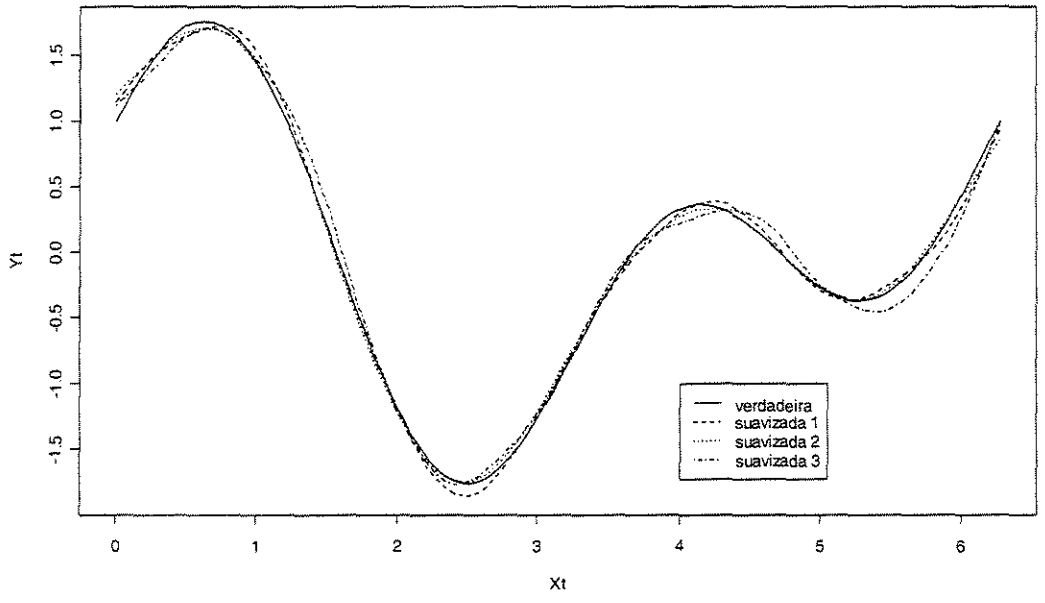


Figura 2.2: Amostra suavizada de três curvas simuladas adicionadas de ruído de baixa variância.

A Figura 2.3 exibe os dois métodos utilizados para obtenção da média estrutural. Observe que ambos se comportam de maneira praticamente idêntica para essa amostra e a diferença entre o método de pré-suavização com ou sem a suavização adicional da média estrutural é bem menor. Na verdade, os três EQM combinados[23] para essa amostra são 2.2496×10^{-5}, 2.2501×10^{-5} e 9.5160×10^{-12} – diferença entre a pós e pré-suavização, pós e pré-suavização com suavização adicional e pós-suavização com e sem suavização adicional, nessa ordem. Esses valores ilustram dois fatos. Primeiro, a diferença em termos de EQM entre os métodos é negligenciável. Segundo, suavizar ou não a média estrutural para o método de pré-suavização fornece um EQM combinado baixíssimo da ordem de 10^{-11}. As curvas obtidas para os três casos aparecem sobrepostas na Figura 2.3.

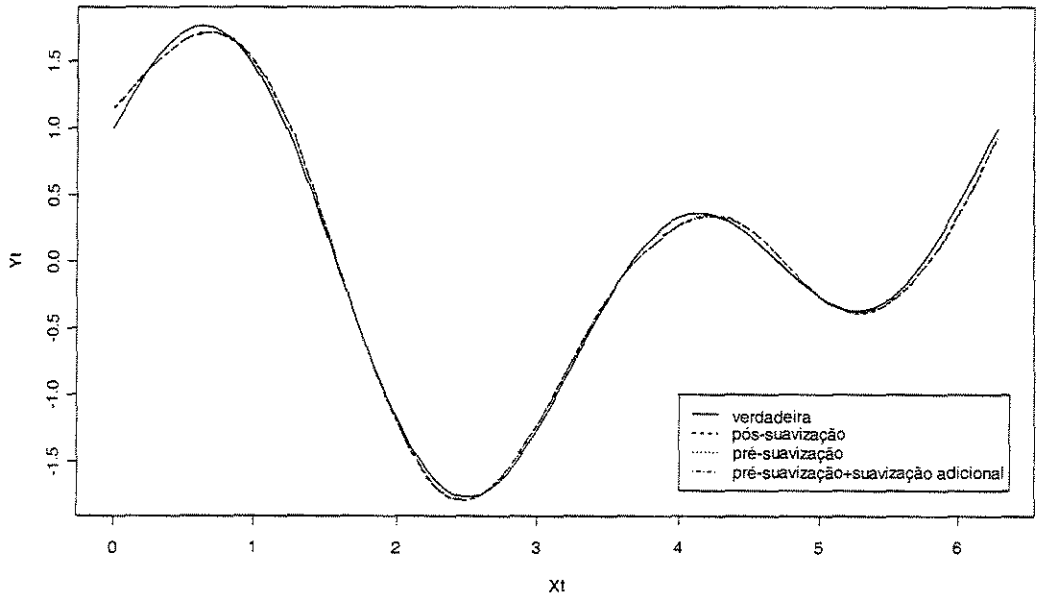


Figura 2.3: Média estrutural das (m = 3) curvas utilizando os 2 métodos.

Foram geradas 1000 amostras de cada curva adicionada de ruído branco, visando estudar o comportamento relativo das três idéias. A Figura 2.4 exibe o boxplot dos três EQM combinados ao longo das simulações. Foi utilizada escala logarítmica por dois motivos: há muitos valores acima do segundo limite interquartílico e a terceira medida

[23] Considerando cada estimativa obtida pelos métodos como referência para cálculo do erro (incluindo a suavização adicional), isto é, o EQM para uma curva tendo como referência uma segunda.

(pós-suavização com e sem suavização adicional) tem escala muito menor do que têm as demais. O gráfico mostra que suavizar ou não a média estrutural quando aplicando a pré-suavização não altera significativamente as estimativas. Se aumentarmos as variâncias do ruído o resultado é praticamente o mesmo.

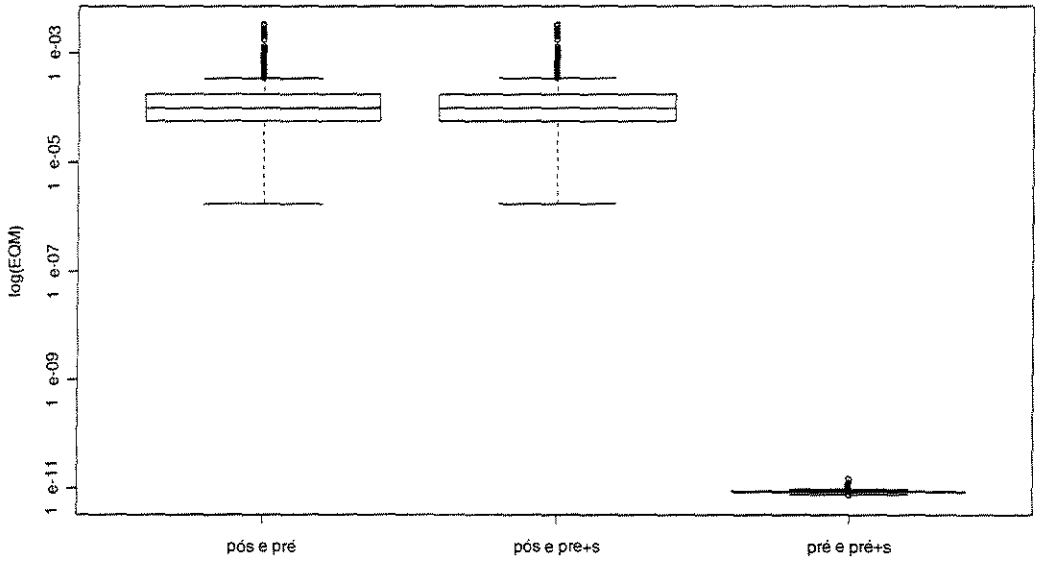


Figura 2.4: Boxplot dos EQM combinados para 1000 amostras de $m = 3$ curvas.

Em caráter ilustrativo, as duas próximas tabelas consideram duas medidas para cada tipo de curva considerada. A medida E1 refere-se ao número relativo de vezes que a pós-suavização foi melhor que a pré-suavização em termos de EQM. O valor de E2 mede o número relativo de vezes que a pré-suavização sem suavização adicional foi melhor que a mesma com suavização adicional.

A Tabela 2.1 é referente à curva

$$Y_t = \cos(X_t) + \text{sen}(2X_t) + \text{erro}. \tag{2.9}$$

A pós-suavização obteve melhor desempenho que a pré-suavização. Além disso, se o número de indivíduos aumenta (aumentando a informação disponível), essa superioridade é ainda mais evidente. A medida E2 informa que é desnecessário suavizar a média estrutural na pré-suavização.

Tabela 2.1: Freqüência relativa dos erros para a curva da Equação (2.9).

Número de replicações (m)	Medidas (%)	
	E1	E2
3	0.726	0.757
5	0.801	0.831
10	0.907	0.903

A Tabela 2.2 é referente à curva

$$Y_t = \alpha X_t + (1-\alpha)e^{-\frac{1}{2}(X_t-\bar{X})^2} + \text{erro}, \tag{2.10}$$

para $\alpha = 0.1$. As conclusões a que se chega são as mesmas daquelas da tabela anterior, ou seja, a pós-suavização é superior em termos de EQM e isso melhora com o aumento do número de curvas. Considerando o tempo computacional, essa idéia é reforçada, pois para 3 curvas, por exemplo, ela leva cerca de 40% do tempo para fornecer a estimativa. Note que se o número de curvas aumenta, isso tende a melhorar, pois deixamos de fazer $m-1$ suavizações para obter a média estrutural[24].

Tabela 2.2: Freqüência relativa dos erros para a curva da Equação (2.10).

Número de replicações (m)	Medidas (%)	
	E1	E2
3	0.731	0.215
5	0.800	0.227
10	0.857	0.235

[24] Para 10 curvas, por exemplo, esse valor é de cerca de 15%.

Por desempenhar-se indiferentemente melhor para os casos estudados, a pós-suavização será utilizada neste trabalho quando for necessário obter a média estrutural de um conjunto de curvas.

2.2.2 Estimação via funções de base: como resumir a curva

Um problema sempre presente em estimação não paramétrica é a escolha do *bandwidth*, ou equivalentemente, do número K de bases, se for o caso, que representarão os dados. Não há consenso geral sobre qual valor tomar e o método usado aqui consiste, como enunciado, em simular o vetor θ através do qual Z definirá quais bases entrarão no modelo. A idéia é realizar esse procedimento um número de vezes suficientemente grande, estimando em cada passo do procedimento os parâmetros β referentes às bases selecionadas por Z e guardar alguma medida resumo das curvas estimadas para ser utilizada na estimativa final. A primeira idéia que surge para a medida resumo é utilizar a estimativa de cada curva $\hat{Y}_i$. No entanto, propomos algumas variações nessa seção. O procedimento é descrito no Algoritmo 2.1.

Algoritmo 2.1: Estimação da média estrutural.

i) Fixe $K_{máx}$, o número máximo de bases que serão utilizadas para representação de cada curva;

ii) Para cada curva i, faça $\theta_{ki} = ½$, $k = 1, ..., m$, a probabilidade de que a base k entre na representação da curva, e simule Z_{ki} com probabilidade θ_{ki}. Se $Z_{ki} = 1$ a base entrará na representação da curva;

iii) Estime $\hat{\beta}_{ki}$ via mínimos quadrados penalizados – referente à expansão em funções de base da curva i considerando as bases selecionadas por Z_{ki};

iv) Volte ao passo (ii) até que um número suficiente de iterações tenha sido efetuado.

Observe que para cada vetor $\hat{\beta}$ obtido na iteração, podemos obter a estimativa da curva para aquela iteração como

$$\hat{Y}_i^{(l)} = \mathbf{B}\hat{\beta}_{Z_i^{(l)}}, \tag{2.11}$$

em que $\mathbf{B}_{K\times n}$ é a matriz de B-*splines* avaliada sobre o *grid* $x_1, ..., x_n$ e

$$\hat{\beta}_{Z_i^{(l)}} = \text{diag}(1_K)_{.Z_i^{(l)}}\hat{\beta}_i \tag{2.12}$$

são as estimativas dos coeficientes das bases selecionadas para a curva i na l-ésima iteração expandidos para o $\Re^K$ [25].

Essa é uma maneira de resumir a estimativa para a curva i ao longo das iterações, guardando a estimativa da curva por inteiro. A estimativa final da curva i pode então ser calculada como

$$\hat{Y}_i = \frac{1}{L}\sum_{l=1}^{L}\hat{Y}_i^{(l)}. \tag{2.13}$$

Considere a seguir algumas formas propostas para resumir a curva através dos coeficientes das bases selecionadas. Seja

$$\tilde{\beta}_{ki} = \frac{1}{L}\sum_{l=1}^{L}\left[Z_{ki}^{(l)}\hat{\beta}_{ki}^{(l)}\right], \tag{2.14}$$

uma média aritmética considerando como zero os coeficientes das bases não selecionadas[26]. Por exemplo, para $m = 1$ curva, $L = 3$ iterações e $K = 5$ bases, suponha que as estimativas dos parâmetros referentes às bases selecionadas para uma dada curva sejam $\hat{\beta}_1^{(1)}, \hat{\beta}_3^{(1)}, \hat{\beta}_4^{(1)}$ para a primeira iteração, $\hat{\beta}_2^{(2)}, \hat{\beta}_3^{(2)}, \hat{\beta}_5^{(2)}$ para a segunda iteração e $\hat{\beta}_3^{(3)}, \hat{\beta}_4^{(3)}, \hat{\beta}_5^{(3)}$ para a terceira iteração. Então,

$$\tilde{\beta}_1 = \frac{\hat{\beta}_1^{(1)}}{3}; \tilde{\beta}_2 = \frac{\hat{\beta}_2^{(2)}}{3}; \tilde{\beta}_3 = \frac{\hat{\beta}_3^{(1)} + \hat{\beta}_3^{(2)} + \hat{\beta}_3^{(3)}}{3}; \tilde{\beta}_4 = \frac{\hat{\beta}_4^{(1)} + \hat{\beta}_4^{(2)}}{3}; \tilde{\beta}_5 = \frac{\hat{\beta}_5^{(2)} + \hat{\beta}_5^{(3)}}{3}.$$

As curvas utilizadas como entrada no procedimento são estimadas via pós-suavização como descrito anteriormente, isto é, geram-se as amostras das curvas para cada iteração e a média estrutural é obtida. Como podemos resumir as estimativas $\hat{Y}_i^{(l)}$ ao longo das iterações para cálculo da estimativa $\hat{Y}_i$ de cada curva da mesma maneira

[25] Preenchendo com zeros as posições das bases que não foram selecionadas.

[26] $\hat{\beta}_{ki}^{(l)}$ é o coeficiente estimado para a base k que entra na representação da curva i na l-ésima iteração.

que podemos resumir cada curva estimada $\hat{Y}_i$ para cálculo da estimativa final $\hat{Y}$, consideraremos aqui apenas estimativas das m curvas para a curva final, pois a analogia para as iterações é direta. Portanto, (2.14) é o caso não ponderado de

$$\tilde{\beta}_k = \frac{\sum_{i=1}^{m} \left[Z_{ki} \hat{\beta}_{ki} \right]}{n(Z_k)}, \tag{2.15}$$

com

$$n(Z_k) = \begin{cases} m, & \text{caso não ponderado;} \quad (2.15a) \\ \max\{1, \sum_{i=1}^{m} Z_{ki}\}, & \text{caso ponderado.} \quad (2.15b) \end{cases}$$

O denominador do caso ponderado é não nulo para evitar divisão por zero[27]. Ou seja, a ponderação leva em conta o número de vezes que cada base k entrou na representação da curva. Dessa forma, as estimativas referentes ao exemplo anterior para iterações seriam calculadas por (2.15b) como

$$\tilde{\beta}_1 = \frac{\hat{\beta}_1^{(1)}}{1}; \tilde{\beta}_2 = \frac{\hat{\beta}_2^{(2)}}{1}; \tilde{\beta}_3 = \frac{\hat{\beta}_3^{(1)} + \hat{\beta}_3^{(2)} + \hat{\beta}_3^{(3)}}{3}; \tilde{\beta}_4 = \frac{\hat{\beta}_4^{(1)} + \hat{\beta}_4^{(2)}}{2}; \tilde{\beta}_5 = \frac{\hat{\beta}_5^{(2)} + \hat{\beta}_5^{(3)}}{2}.$$

Há uma forma semelhante de ponderação, descrita por

$$\tilde{\beta}_k = \frac{\sum_{i=1}^{m} \left[Z_{ki} \hat{\beta}_{ki} \sum_{r=1}^{K} Z_{ri} \right]}{n'(Z_k)}, \tag{2.16}$$

com

$$n'(Z_k) = \begin{cases} \sum_{i=1}^{m} \sum_{r=1}^{K} Z_{ri}, & \text{caso não ponderado;} \quad (2.16a) \\ \max\left\{1, \sum_{i=1}^{m} \left[Z_{ki} \sum_{r=1}^{K} Z_{ri} \right]\right\}, & \text{caso ponderado.} \quad (2.16b) \end{cases}$$

[27] A base pode não ser selecionada na representação de nenhuma curva.

Essa forma de ponderação considera dois valores: o número de bases necessárias para expandir cada curva e o número de vezes que a base foi selecionada para expandir uma dada curva[28]. Uma outra forma de resumir as curvas também foi considerada e é dada por

$$\tilde{\beta}_k^{\cdot\cdot} = \frac{\sum_{i=1}^{m} \left[\frac{Z_{ki}\hat{\beta}_{ki}}{\sum_{r=1}^{K} Z_{ri}} \right]}{n^{\cdot\cdot}(Z_k)}, \tag{2.17}$$

com

$$n^{\cdot\cdot}(Z_k) = \begin{cases} \sum_{i=1}^{m} \frac{1}{\sum_{r=1}^{K} Z_{ri}}, & \text{caso não ponderado;} \quad (2.17a) \\ \begin{cases} 1, & \text{se } \sum_{i=1}^{m} Z_{ki} = 0 \\ \sum_{i=1}^{m} \frac{Z_{ki}}{\sum_{r=1}^{K} Z_{ri}}, & \text{se } \sum_{i=1}^{m} Z_{ki} > 0 \end{cases}, & \text{caso ponderado.} \quad (2.17b) \end{cases}$$

Ela é análoga à forma anterior, mas considera como peso o inverso do número de bases selecionadas para expandir cada curva[29], ou seja, quanto mais bases há na representação de uma dada curva, menor é o peso dado a cada base. No Apêndice B encontram-se ilustrações para casos particulares em que as medidas são equivalentes.

Todas essas seis formas de resumir as estimativas $\hat{Y}_i$ das curvas para posteriormente calcular a estimativa final $\hat{Y}$ mostraram-se boas em termos de EQM, como poderá ser visto na seção de simulações. Dessa forma, podemos optar por resumir cada curva através dos coeficientes das bases e calcular

[28] Novamente, o denominador considera o máximo porque a base pode não ser selecionada na representação de nenhuma curva.

[29] A bifurcação para o caso ponderado é usada porque a base pode não ser selecionada por nenhuma curva. Não foi usado o máximo desta vez porque a soma pode ser menor que 1 e o máximo desconfiguraria a idéia da ponderação.

$$\hat{Y} = \mathbf{B}\dot{\beta}, \tag{2.18}$$

em que $\dot{\beta}$ é obtido por (2.15), (2.16) ou (2.17), ou através da sua estimativa final derivada de (2.11) e (2.13):

$$\hat{Y} = \frac{1}{m}\sum_{i=1}^{m} \hat{Y}_i, \tag{2.19}$$

em que

$$\hat{Y}_i = \mathbf{B}\dot{\beta}_{Z_i} \tag{2.20}$$

e Z_i é um vetor k-dimensional preenchido com uns nas posições equivalentes às bases selecionadas para a representação da curva i.

No entanto, para $\dot{\beta} = \tilde{\beta}$ no caso não ponderado, é útil notar que (2.18) e (2.19) são equivalentes (Proposição B.1 do Apêndice B). Isso significa que basta considerar os coeficientes das bases para resumo das curvas estimadas. Há duas vantagens computacionais em optar por usar o resumo pelos coeficientes. A primeira é que economizamos espaço de memória, pois $K < n$ (possivelmente $K << n$). A outra é que o cálculo usando $\hat{Y}_i$ pode ser instável, já que efetua m produtos internos $\mathbf{B}\hat{\beta}_{Z_i}$ antes da estimativa final, ao passo que $\hat{Y} = \mathbf{B}\tilde{\beta}$ necessita de apenas um.

Além disso, é mais flexível idealizar medidas resumo a partir dos coeficientes do que de $\hat{Y}_i$. Tente por exemplo ponderar $\hat{Y}_i$ de alguma outra forma.

2.3 Algoritmos EM, SEM e uma variante

Freqüentemente encontramos problemas na Estatística onde há dados faltantes ou incompletos. O algoritmo EM, formalizado por Dempster et al. (1977) é uma técnica popular para lidar com tais problemas. Problemas onde há variáveis latentes podem ser formulados de modo a serem resolvidos aplicando-se a idéia do algoritmo EM.

Além da facilidade de programação do algoritmo, ele fornece estimativas de máxima verossimilhança. No entanto, pode haver convergência para máximos locais ou

mesmo para pontos de sela da função de log-verossimilhança, além de sua possível sensibilidade aos valores iniciais. Ele pode ainda envolver computação de integrais múltiplas de alta ordem.

O algoritmo SEM (Celeux and Diebolt, 1985) fornece uma alternativa para o algoritmo EM. No passo S, os dados faltantes são imputados com valores plausíveis, dado o conjunto de observações e uma estimativa dos parâmetros. No passo M, os parâmetros de máxima verossimilhança são computados, baseados nos dados completados. Ao contrário do EM determinístico, a saída do algoritmo SEM é uma amostra de uma distribuição estacionária cuja média é próxima à estimativa de máxima verossimilhança e cuja variância reflete a perda de informação devido aos dados faltantes.

Em geral, o algoritmo SEM converge mais rápido para seu regime estacionário do que algumas cadeias longas utilizadas em aplicações do amostrador de Gibbs. Em relação ao EM, o algoritmo SEM substitui o passo E pela simulação.

2.3.1 Uma variante do algoritmo SEM

Podemos estimar a curva que melhor representa o processo, objetivo de nosso trabalho, através de uma variante do algoritmo SEM. O interesse primário é verificar se essa variante é apropriada para essa classe de problemas, isto é, se sua performance é adequada. Para tanto, implementamos uma versão da mesma, descrita pelo Algoritmo 2.2.

O passo 2 do algoritmo consiste tão somente do chute inicial para o vetor $\hat{\theta}$. Para o passo 3, a restrição em (i) é motivada por identificabilidade do problema (deve haver pelo menos 3 bases para estimação de cada curva). Os passos (ii)-(iv) constituem a etapa EM do algoritmo. Como a solução de máxima verossimilhança é obtida através de métodos numéricos (observe que $\hat{\beta}$ é um vetor para cada curva), temos que fornecer um chute inicial para $\hat{\beta}$ (neste caso, a estimativa via mínimos quadrados). Pela própria estrutura da equação de verossimilhança, $\hat{\lambda}$ tende a ser tão grande quanto se é permitido. Por isso, ele será tratado aqui como parâmetro fixado pelo usuário. A atualização de $\hat{\theta}$ (passo (v)) é realizada exclusivamente nas bases selecionadas por Z. As demais

permanecem inalteradas. O critério de parada que está sendo utilizado atualmente é uma medida de proximidade entre a estimativa e a verdadeira curva que gerou o processo – para estudo teórico, pois a verdadeira curva não é conhecida.

Algoritmo 2.2: Estimação da média estrutural através da variante do algoritmo SEM.

1. Fixe $K_{máx}$;
2. Para cada curva i, faça $\hat{\theta}_{ki}^{(0)} = ½$, $k = 1, ..., K_{máx}$;
3. Para cada iteração l:
   i. gere $Z_{ki}^{(l)} \sim \text{Bernoulli}\left(\hat{\theta}_{ki}^{(l)}\right)$ até $\sum_{k=1}^{K_{máx}} Z_{ki}^{(l)} \geq 3$;
   ii. Estime o vetor $\hat{\beta}_i^{(l)}$ e $\hat{\sigma}^2$ via mínimos quadrados;
   iii. Estime $\hat{\lambda}$;
   iv. Obtenha o vetor $\hat{\beta}_i^{(l)}$ e $\hat{\sigma}^2$ via máxima verossimilhança;
   v. Atualize $\hat{\theta}_{ki}$ através da Equação 2.6: $\hat{\theta}_{ki}^{(l+1)} = 1 - e^{-\hat{\lambda}^{(l)}|\hat{\beta}_k^{(l)}|}$;
   vi. Guarde $\hat{\beta}_i^{(l)}$;
   vii. Se o critério de parada for satisfeito, pare. Senão, volte ao passo i.
4. Obtenha a média estrutural através de (2.18).

2.3.2 Um exemplo

A Figura 2.5 mostra a curva estimada utilizando todas as bases possíveis. Observe que pelo fato de que esse número é alto, a estimativa tende a interpolar o erro e, assim, perde sua natureza suave.

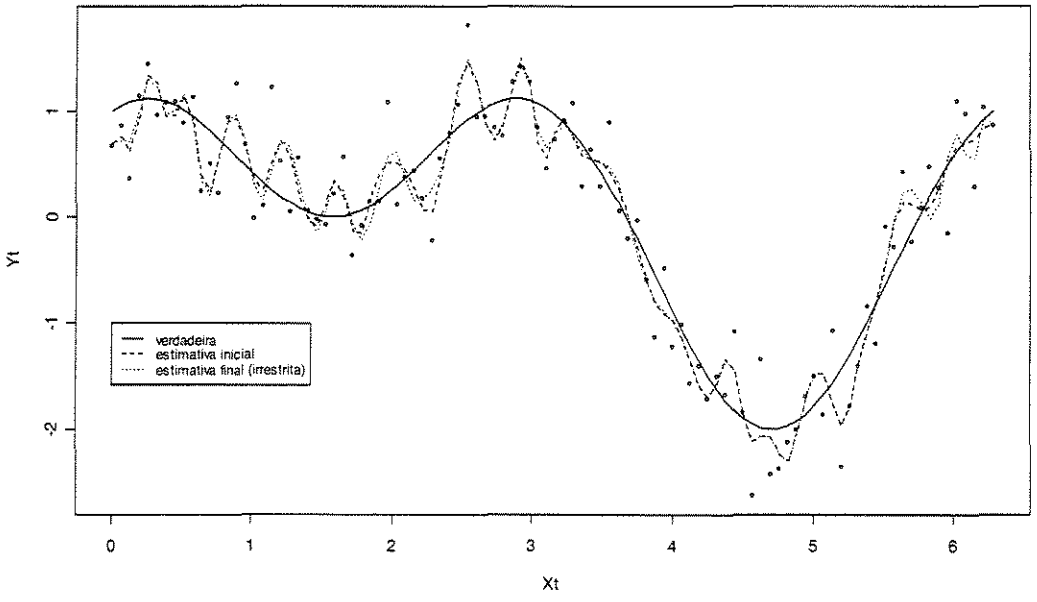


Figura 2.5: Estimação de uma curva utilizando o número máximo de bases (passo 0).

Esse é o papel da variável Z – selecionar quais bases devem entrar na representação de cada curva. Se não impusermos restrições a $\hat{\lambda}$, a estimativa final tenderá a interpolar o ruído – não sendo muito diferente da estimativa inicial, pois todo o vetor $\hat{\theta}$ tenderá a 1 – pelo menos para as bases selecionadas.

Para exemplificar isso, fixamos o valor de $K_{máx}$ como 50 e efetuamos um total de 100 simulações para uma única curva. A Figura 2.6 exibe o comportamento do vetor $\hat{\theta}$ durante as 70 primeiras iterações para cada uma das 50 bases (possivelmente) selecionadas. Note que cada boxplot concentra-se nos altos valores de $\hat{\theta}$, forçando, em probabilidade, a entrada de todas as bases no modelo (Equação 2.6), aumentando, assim, a probabilidade de que cada uma das bases seja selecionada. Os *outliers* não foram exibidos por não serem relevantes para essa ilustração.

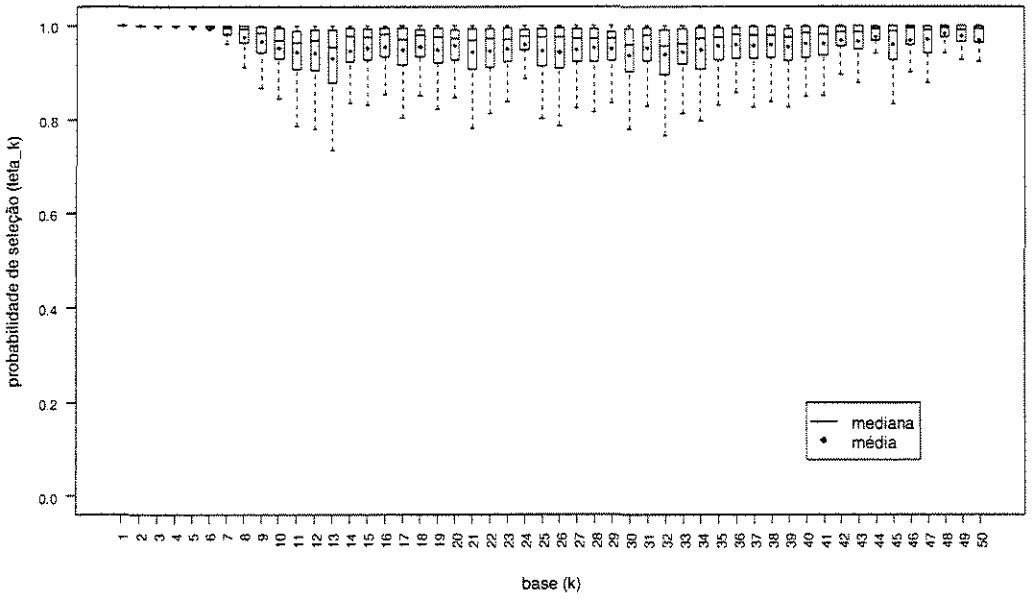


Figura 2.6: 100 Valores de $\hat{\theta}_k$, $k = 1, ..., 50$, para as 70 primeiras iterações na estimação de uma curva.

A Figura 2.7 resume a informação contida ao longo das 70 primeiras iterações do exemplo anterior para a dimensão do vetor $\hat{\beta}$ (número de bases selecionadas por Z). Ela confirma a idéia de que se os valores de $\hat{\theta}_k$ conjuntamente aproximam-se de 1 (Figura 2.6), então a dimensão de $\hat{\beta}$ (= K) tenderá a $K_{máx} = 50$. Ainda que consideremos que a variação desse valor seja alta, seu mínimo é sempre maior ou igual a 40 a partir da terceira iteração.

Como esse problema deriva do fato de que $\hat{\lambda}$ tende a ser tão grande quanto possível, seria interessante encontrar uma forma racional de limitar seu valor. Para tanto, estudamos o comportamento do número de bases selecionadas ao final das iterações em função de um limitante M. Um exemplo é exibido na Figura 2.8.

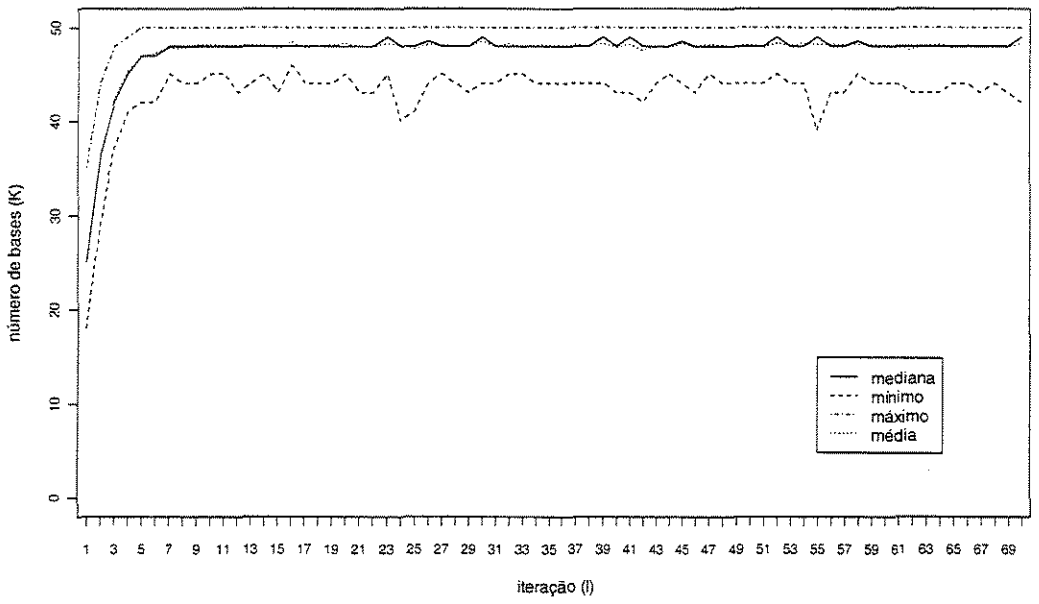


Figura 2.7: Número de bases ($K^{(l)}$) selecionadas ao longo das 70 primeiras iterações na estimação de uma curva para os 100 casos simulados.

Aparentemente M é uma função exponencial do número ideal de bases. No gráfico acima encontram-se cinco vetores de estatísticas $S(K)$. Os três centrais são a média, a mediana e a moda do número de bases selecionadas ao longo das iterações. A curva mais à esquerda representa o mínimo enquanto a mais à direita o número máximo de bases selecionadas. Por possuírem aparência mais comportada entre as demais, optamos pelo uso da média e da mediana para estudar o relacionamento entre M e K.

Montamos então um modelo de regressão (Figura 2.9) para as estatísticas selecionadas conforme abaixo:

$$M = \alpha_0 e^{-\alpha_1 S(K)},$$

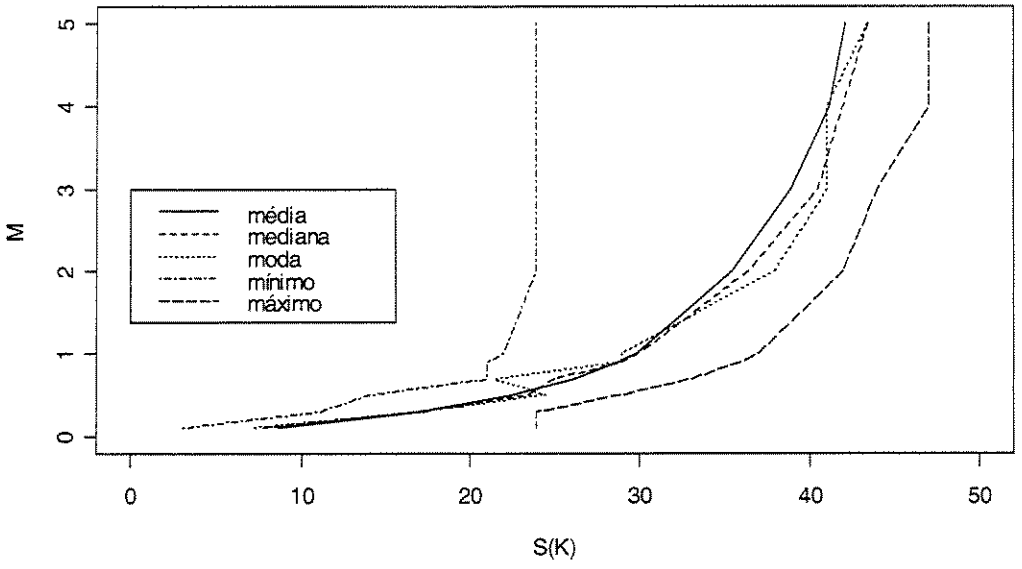


Figura 2.8: Relacionamento empírico entre M e K.

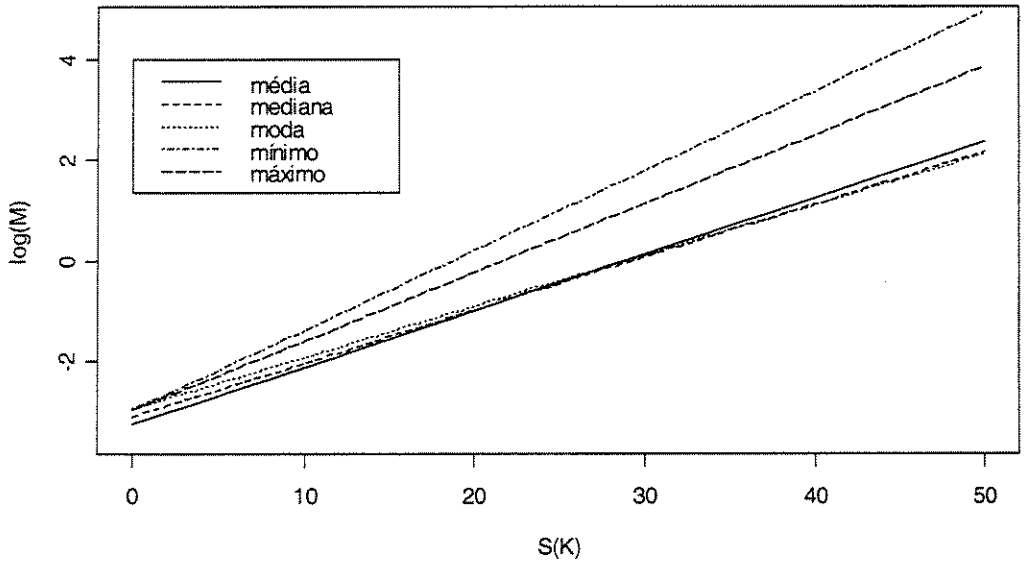


Figura 2.9: Resultado da regressão para as curvas da Figura 2.8.

em que $S(K)$ é ou a média ou a mediana do número de bases selecionadas ao longo das iterações. Para os dois tipos de curva simulados (tipo A e tipo B), os valores das estimativas de regressão são dados na Tabela 2.3. A Figura 2.9 reforça o uso da média e da mediana como representantes do comportamento empírico de M em função de K (estas duas, além da moda, encontram-se mais próximas uma da outra e seu comportamento é mais regular para as curvas estudadas até agora – não usamos a moda porque ela pode não ser única).

Tabela 2.3: Estimativas dos parâmetros da regressão do modelo log-linear para dois exemplos.

$S(K)$	Curva tipo A		Curva tipo B	
	$\hat{\alpha}_0$	$\hat{\alpha}_1$	$\hat{\alpha}_0$	$\hat{\alpha}_1$
Média	-3.2477	0.1122	-5.1200	0.1290
Mediana	-3.0871	0.1053	-4.8254	0.1171

No entanto, para todas as simulações λ tende a ser tão grande quanto se é permitido. Portanto, decidimos limitar seu valor pois nossa análise do comportamento do limitante M com relação a λ não permitiu encontrar um critério automático de escolha para o valor de M. O procedimento atual é normalizar as bases, modificando a Equação (2.6) para

$$\theta_{ki} = 1 - e^{-\lambda \frac{|\beta_{ki}|}{\sum_r |\beta_{ri}|}}. \tag{2.21}$$

Tal transformação (que mantém o vínculo entre β e θ como na Equação (2.6)) modifica apenas as derivadas da Equação (2.7), necessárias para o processo de maximização (ver Apêndice A), mas torna mais claro o significado do valor de λ. Isso porque o maior valor possível da fração do expoente é 1[30], pois para k e i fixos,

$$\lim_{\beta_{ki} \to \infty} \frac{|\beta_{ki}|}{\sum_r |\beta_{ri}|} = 1. \tag{2.22}$$

[30] Quando há uma base cujo coeficiente estimado (em módulo) é grande quando comparado com a soma dos módulos de todos os coeficientes.

Por exemplo, para λ=1, θ_k será no máximo $1 - e^{-1} \cong 0.63$. Portanto, é intuitivamente mais fácil escolher um valor de λ que impeça valores extremos de θ.

Além disso, precisamos analisar a dimensão do espaço onde as bases serão estimadas. Quanto maior for essa dimensão, associada ao valor de K, maior a chance de termos muitas bases no modelo, pois a probabilidade de seleção de cada base é sempre positiva. Uma sugestão (Dias and Gamerman, 2002) é que pelo menos

$$K = 3b + 2, \tag{2.23}$$

onde b é o número de *bumps* para cada curva. O valor que estamos utilizando é $4b$+3. Esse valor será sempre maior ou igual a 3[31] e maior que a sugestão da literatura, isso porque uma vez fixado o valor de $K_{máx}$, o número de bases ao longo das iterações será no máximo $K_{máx}$.

Os exemplos ilustrados referem-se a dois modelos de curva, uma razoavelmente complexa, com algumas oscilações e uma outra mais simples, mas que possui propositalmente um ponto de inflexão não muito evidente para ver se as estimações corroboram a existência dessa inflexão ou ignoram-na pela natureza da variabilidade do ruído.

Para a curva do tipo 1, o conjunto de bases gerado para a estimação encontra-se na Figura 2.10c. Devido à quantidade de *bumps*, esse número é grande para permitir que as bases captem todas as oscilações. Para a curva do tipo 2, tal conjunto (Figura 2.10d) é menor pelo mesmo motivo. Observe que essas são as bases que serão ou não selecionadas por Z através de θ. Se Z não seleciona apropriadamente tais bases, por exemplo, retirando bases de locais onde elas são necessárias, a estimativa final não acompanhará totalmente a oscilação da curva, como é visto na Figura 2.11.

[31] Atendendo à restrição (3.i) do Algoritmo 2.2.

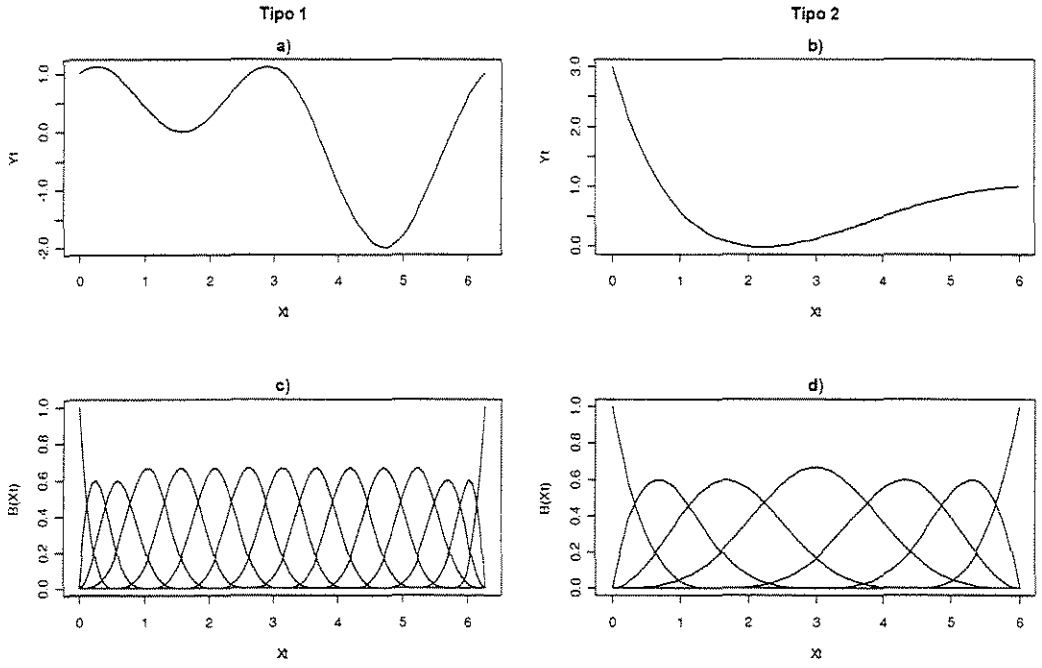


Figura 2.10: Superior: a) e b) Curvas estudadas, com *b* sendo escolhido como 3 e 1, respectivamente; Inferior: c) e d) Bases geradas para a representação de cada curva.

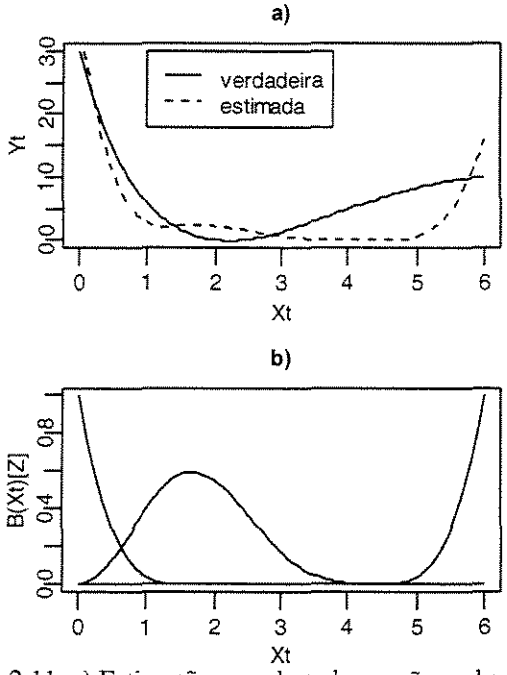


Figura 2.11: a) Estimação quando as bases são mal selecionadas;
b) As 3 bases selecionadas por *Z*.

3 *Simulações*

3.1 Suposições

Antes de continuar é preciso fazer algumas considerações. Para realizar as simulações esteve implícito que os dados gerados representam um problema após alguma limpeza como, por exemplo, estudo da presença de *outliers*. Além disso, o argumento sobre o qual são feitas as medidas é unidimensional e será denotado por t. Sem perda de generalidade as observações são igualmente espaçadas e tomadas sobre os mesmos valores (*grid*) para todos os indivíduos. Será assumido também que há replicações de cada função indexadas por i, $i = 1, ..., m$, e que o *grid*, idêntico para todos os indivíduos, tem n pontos. Adicionalmente, as funções estão registradas e alinhadas em algum sentido, de modo que propriedades importantes de cada curva ocorrem aproximadamente sobre os mesmos valores de t.

3.2 Implementações

Entre a fase inicial e mediana da dissertação, as implementações eram realizadas exclusivamente no programa R (http://cran.r-project.org). Como as simulações estavam demandando muito tempo, resolvemos implementar uma versão em Ox (http://www.nuff.ox.ac/uk/users/Doornik). Ao contrário do R, o Ox não possui uma versão implementada para cálculo da matriz **B** de B-*splines*. Portanto, foi necessário implementar uma versão eficiente da Equação (1.3). Todas as comparações de velocidade foram feitas em um laptop Intel(R) Celeron (TM) CPU 1100 MHz, 1.10 Ghz, com 112Mb de RAM, sob plataforma Microsoft Windows XP 2000 Professional, versão 2002.

com 112Mb de RAM, sob plataforma Microsoft Windows XP 2000 Professional, versão 2002.

Estudamos duas implementações disponíveis na Internet (Fortran Source Codes e a versão em C de Douglas M. Bates and William (Bill) N. Venables para o R. Preferimos a versão em C porque além de ser a mesma usada pelo R (por questões de comparação), ela computa B-*splines* de qualquer grau, enquanto que a versão em Fortran fornece apenas B-*splines* cúbicas. Nossa versão é uma tradução para Ox da função splineDesign() do pacote splines do R. Para avaliar a estabilidade e velocidade de cálculo das versões em Ox e R, implementamos também, experimentalmente, uma versão no Matlab (http://www.mathworks.com) – baseada no código presente em Eilers and Marx (1996) – e uma tradução dela para a linguagem do Maple (http://www.maplesoft.com). As principais implementações utilizadas, em suas versões na data de entrega da versão final dessa dissertação, encontram-se no Apêndice C.

Apesar de a versão em Ox ter consumido cerca de 75% do tempo necessário para realizar a mesma tarefa pela versão em R – cálculo de 10000 matrizes de B-*splines* cúbicas com 15 bases num *grid* variável de 100 pontos –, essa não é a maior vantagem de velocidade computacional da versão em Ox, pois o Algoritmo 1.2 necessita que a matriz **B** seja calculada uma única vez.

Além disso, constatou-se que a superioridade de precisão da versão em Ox é pequena o suficiente para poder ser considerada desprezível[32]. As versões em Ox e em Matlab foram mais similares nesse aspecto do que a versão em R. A comparação numérica foi feita com a versão em Maple, aumentando a precisão para 100 dígitos. Utilizando variáveis simbólicas no Maple para alguns exemplos, obteve-se a matriz **B** como função dos nós e do *grid* x onde ela era avaliada e comparou-se com valores numéricos das versões em Ox, R e Matlab.

Nessa seção, os resultados das simulações foram todos obtidos através da implementação em Ox do Algoritmo 1.2. No entanto, por motivos de uniformidade, continuamos a exibir os gráficos no R. Quando for de interesse ilustrar a vantagem de ter utilizado uma implementação em Ox (em geral a implementação do Algoritmo 1.2 em

[32] A maior diferença numérica em módulo encontrada nas posições da matriz **B** foi da ordem de 10^{-15}.

Ox foi de 3 a 7 vezes mais rápida do que a equivalente em R), serão exibidos o seu tempo de simulação nas duas versões.

3.3 Estudo das formas de resumo da curva estimada

Em primeiro âmbito analisaremos o comportamento do EQM ao se estimar a curva final por cada uma das equações (2.15), (2.16) e (2.17).

A Figura 3.1 exibe as estimativas obtidas através desses estimadores. A simulação consiste da geração de $m = 3$ curvas com ruído de variância pequena. As estimativas obtidas através de (2.15), (2.16) e (2.17) são praticamente idênticas, ilustrando a idéia de que os estimadores são similares. Além disso, elas inferem bem a verdadeira curva. A convergência foi atingida em 21, 141 e 159 iterações, respectivamente, para cada curva.

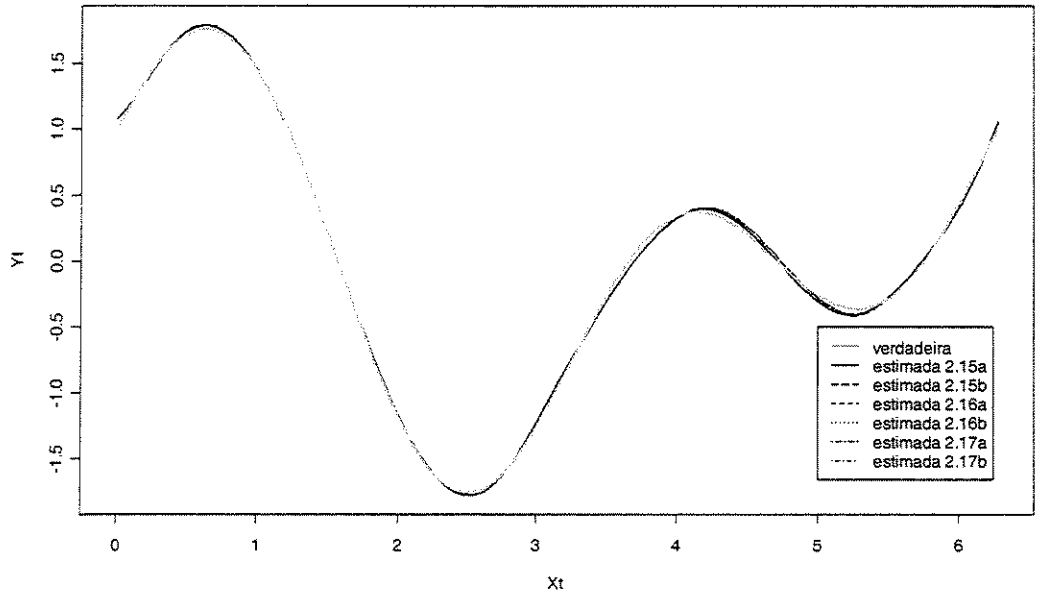


Figura 3.1: Curvas simuladas adicionadas de ruído de baixa variância.

A Figura 3.2 exibe a estimação para $m = 3$ curvas com ruído de variância maior. Observe como a estimação foi prejudicada. Isso é explicado porque o processo de

estimação para uma das curvas não atingiu a convergência em menos de 1000 iterações. Isso pode acontecer quando uma ou mais curvas possuem variância alta o suficiente para omitir o padrão comum. Aumentar o número máximo de iterações pode não resolver o problema. Há curvas cujo processo de estimação não converge mesmo para 10000 iterações. Considerar como bons os estimadores que sobressaíram-se em relação aos demais para casos como esse não é conclusivo, pois sua aparente robustez aparece em um problema[33] fácil de ser resolvido.

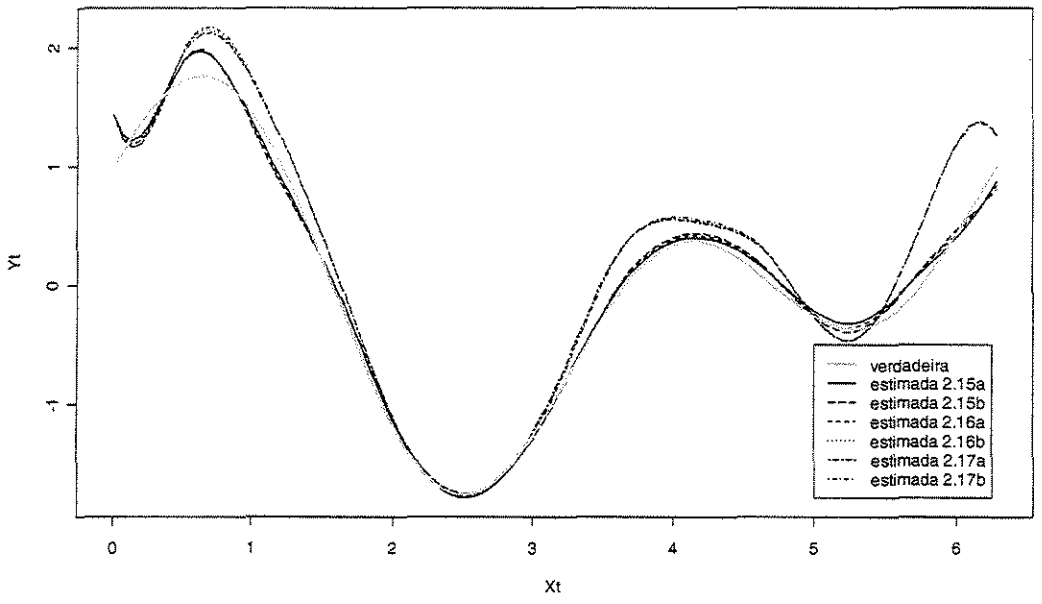


Figura 3.2: Curvas simuladas adicionadas de ruído de alta variância – convergência não atingida.

Em termos de simulação, ou até mesmo na prática, não dá para definir um critério de parada isento de críticas. Quão baixo deve estar o EQM para considerarmos boa a estimação? Isso depende da escala das curvas e da variância do ruído. Pode ainda depender da semente utilizada para iniciar o gerador de números aleatórios.

Foi pensando nisso que criamos um critério de parada flexível. Seja δ o valor de tolerância para o critério de parada, isto é, o processo de estimação continua até que o EQM seja menor ou igual a δ. Para cada curva a ser estimada, se o limite máximo do número de iterações (1000) for atingido, e o critério de parada não for atingido, faça

[33] Falta de convergência.

$\delta \leftarrow c\delta$ e reinicie a estimação para essa curva, onde c é uma constante maior que um[34]. Valores muito pequenos de c podem tornar o processo de estimação da curva lento, em especial nos casos de variância muito alta. Em contrapartida, valores muito grandes de c flexibilizam muito o critério de parada, permitindo que estimativas com alto valor de EQM sejam aceitas como representantes da curva em questão. Para as simulações nesse trabalho, o valor de c = 1.3 forneceu bons resultados. A Figura 3.3 exibe a estimação para as mesmas curvas simuladas da Figura 3.2.

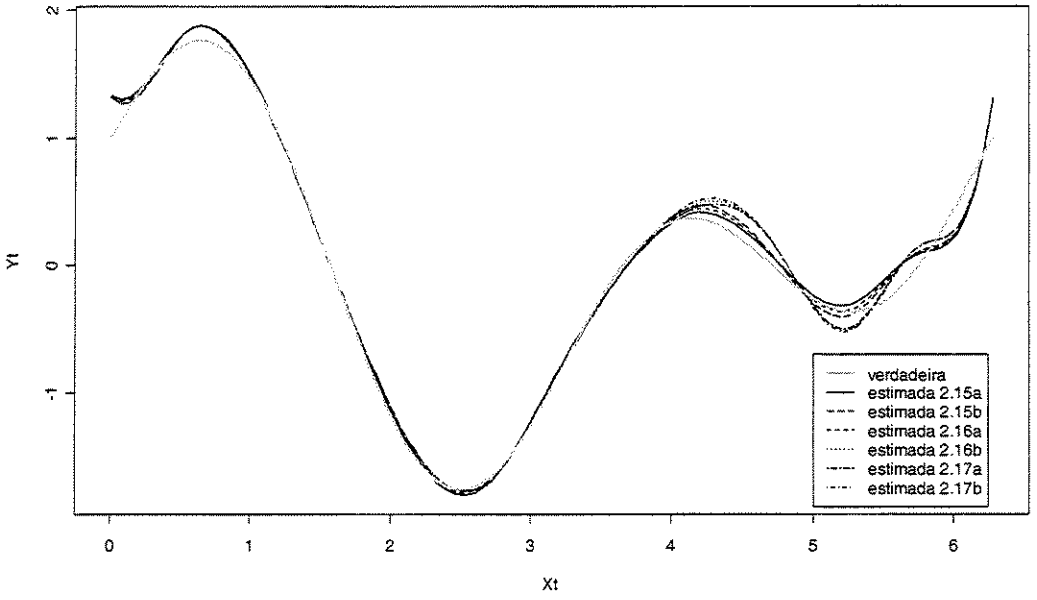


Figura 3.3: Curvas simuladas adicionadas de ruído de alta variância – critério de parada flexível.

Dessa vez o resultado é melhor. A convergência para a curva que causava problema foi atingida logo na oitava iteração do segundo laço, ou seja, foram necessárias apenas 1008 iterações. As demais curvas atingiram convergência com o mesmo número de iterações que no caso anterior.

A análise de apenas um caso pode estar sujeita à escolha da semente para o gerador de números aleatórios utilizado. Portanto, estudamos o EQM para 100 casos de simulação, variando-se apenas a semente do gerador. A Figura 3.4 exibe o resultado para 100 simulações com m = 3 curvas adicionadas de ruído de baixa variância. Em todos os

[34] O valor de c é especificado pelo usuário.

casos a convergência foi atingida. A escala logarítmica foi usada para melhorar a visualização do gráfico. O tempo de simulação no Ox foi cerca de 38 minutos, enquanto que no R foi de aproximadamente 3 horas e 25 minutos. Pela figura, concluímos que os estimadores (2.15a), (2.16a) e (2.17a) produziram menores EQM que os demais.

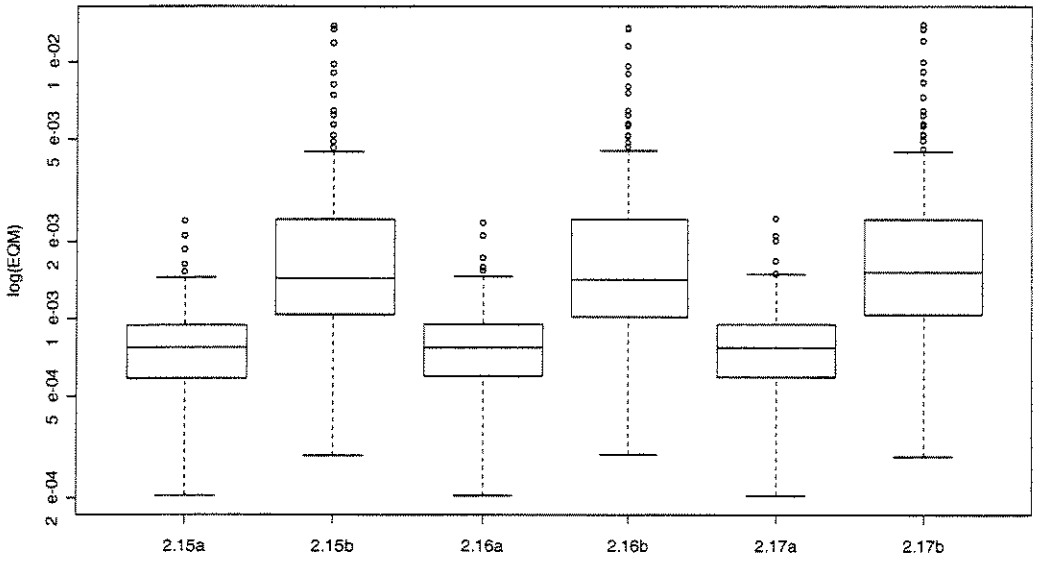


Figura 3.4: EQM de 100 simulações. Curvas com ruído de baixa variância – critério de parada flexível.

A Figura 3.5 mostra a simulação de 100 casos para $m = 3$ curvas com ruído de alta variância. As conclusões são as mesmas. A única mudança é que os EQM foram mais dispersos, provavelmente devido à alta variância do ruído. Em nenhum caso foram necessários mais que quatro laços de 1000 iterações para atingir a convergência, ilustrando a vantagem de utilizar o critério de parada flexível.

Para ilustrar a vantagem em se utilizar um critério flexível, observe a Figura 3.6, onde o critério de parada foi fixado. Além do EQM ser em média mais alto para cada estimador considerado, ele apresentou uma dispersão maior, talvez porque as estimativas que não atingiram convergência sejam bem diferentes das que atingiram.

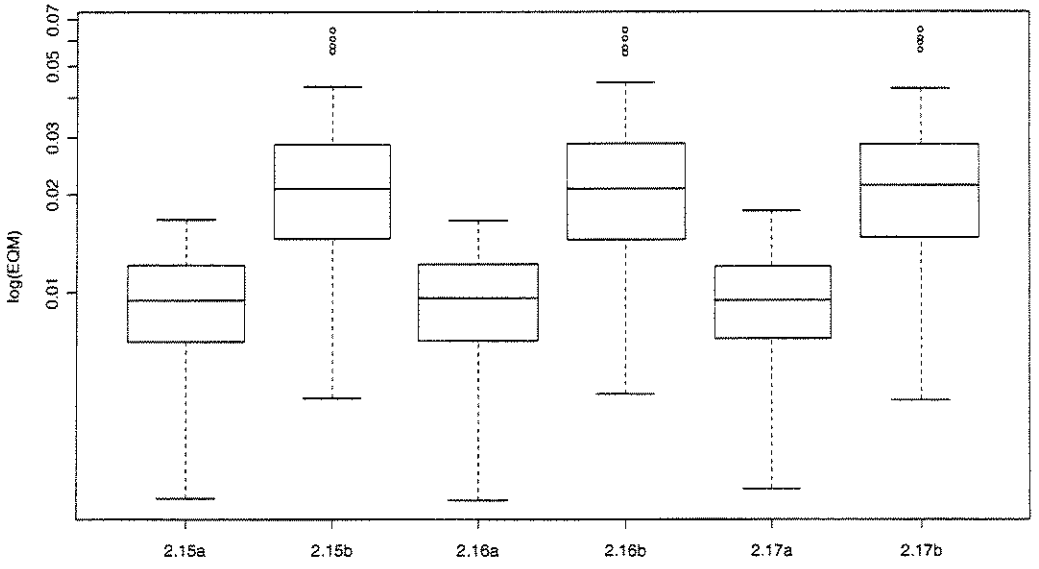


Figura 3.5: EQM de 100 simulações. Curvas com ruído de alta variância – critério de parada flexível.

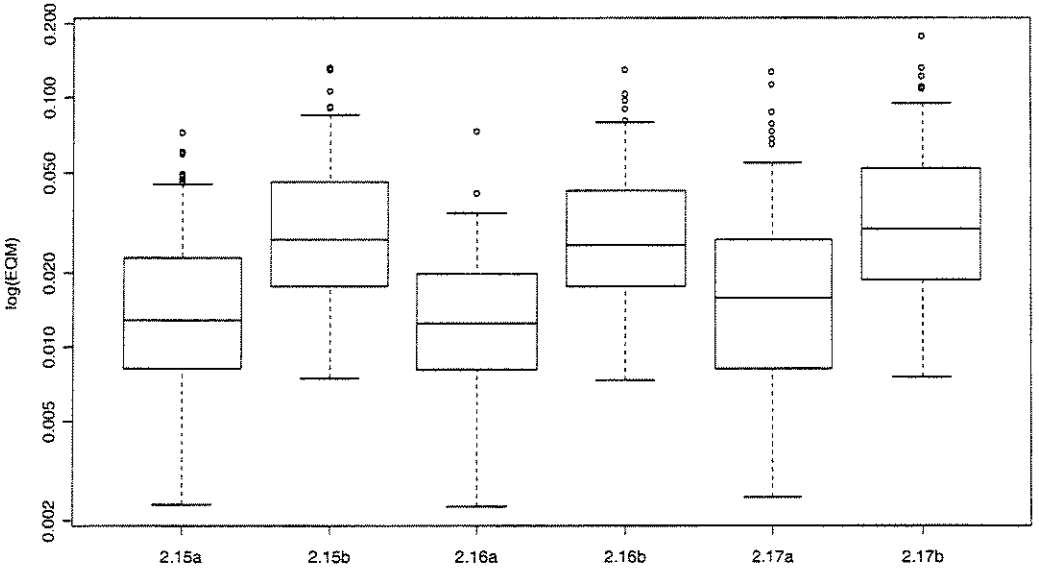


Figura 3.6: EQM de 100 simulações. Curvas com ruído de alta variância – critério de parada fixo.

Em todos esses casos, as versões não ponderadas dos estimadores foram as que obtiveram menor EQM médio. As diferenças entre os EQM médios para essas versões foram sutis o suficiente para não serem consideradas significantes – cerca de 10^{-4}. A Figura 3.7 tem papel meramente ilustrativo. Ela considera os EQM médios de 100 simulações para $m = 7$ curvas com ruído de baixa variância. Note que o fato de termos mais curvas e, portanto, mais informação na amostra, diminui os EQM médios, além de diferenciar ainda mais a eficiência entre as versões não ponderadas e ponderadas dos estimadores que resumem as curvas ao longo das iterações.

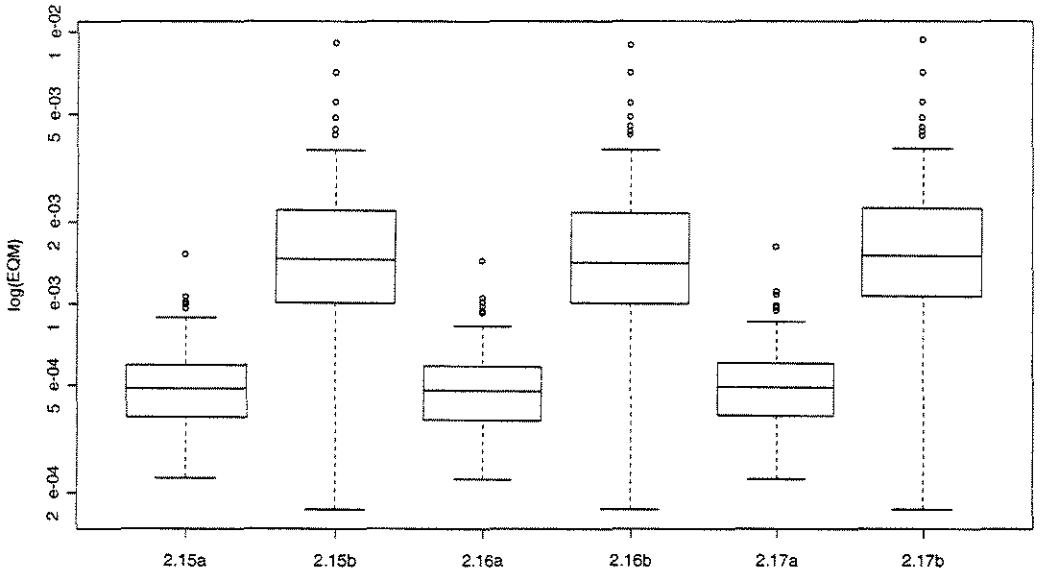


Figura 3.7: EQM para 100 simulações de $m = 7$ curvas com ruído de baixa variância – critério de parada flexível.

Dessa forma, concluímos nessa seção que as versões não ponderadas das equações (2.15), (2.16) e (2.17) são mais apropriadas para resumir as curvas ao longo das iterações. Adicionalmente, o critério de parada flexível mostrou-se útil quando comparando os EQM das estimativas produzidas por ele com o daquelas produzidas com o critério fixo. De fato, quando diminuímos o número máximo de iterações por laço de 1000 para um valor bem menor (por exemplo, 400), o critério de parada flexível mostrou ainda bons resultados. Além disso, com essa alteração o tempo até que se atinja convergência é bastante reduzido.

Contudo, não podemos diminuir demais esse valor. Lembre-se que estamos procurando um subconjunto de bases dentre os $2^{K_{max}}$ possíveis (na verdade, $2^{K_{max}} - \Sigma_{k=0}^{2}\binom{K_{max}}{k}$ – desconsideramos os casos com nenhuma, uma e duas bases, devido à restrição (3.i) do Algoritmo 2.2) e se esse valor é pequeno podemos restringir demais o espaço onde procuramos. Baseados nesses resultados, nas próximas seções utilizaremos as implementações das versões (2.15a), (2.16a) e (2.17a) com o critério de parada flexível.

3.4 Análise do EQM considerando o número de curvas, variância do ruído e forma da função simulada

Nessa seção analisaremos o comportamento do Algoritmo 2.2 quando aumentamos o número de curvas na amostra e alteramos a variância do ruído para duas formas de curva (equações (2.9) e (2.10)).

As figuras 3.8, 3.9 e 3.10 consideram a estimação para a curva da Equação (2.9) adicionada de ruído de baixa, moderada e alta variância, respectivamente, quando aumentamos o número de curvas na amostra.

É visível que quanto maior a variância do ruído, menor é a qualidade da estimação resultante. Por outro lado, quando aumentamos o número de curvas na amostra, a qualidade da estimação melhora. Compare os gráficos superior (a) e inferior (c) das figuras 3.9 e 3.10 para ver que o aumento do número de curvas na amostra balanceia a falta de informação produzida pela alta variância do ruído. A Tabela 3.1 resume essas idéias considerando a curva da Equação (2.9).

Ao que parece, todas as medidas resumo forneceram boas estimativas. Aparentemente, a medida (2.16a) aproxima melhor a verdadeira curva nos casos onde há alta variância ou muitas curvas na amostra. Em apenas um caso a medida (2.15a) sobressai-se. Nos restantes, (2.17a) foi a medida mais vantajosa. Ao observar a Tabela 3.2, que considera a curva da Equação (2.10), chegamos praticamente à mesma conclusão. Cada medida saiu-se melhor aproximadamente nos mesmos casos tanto para a curva (2.9) quanto para a curva (2.10). Contudo, note que há valores de EQM idênticos

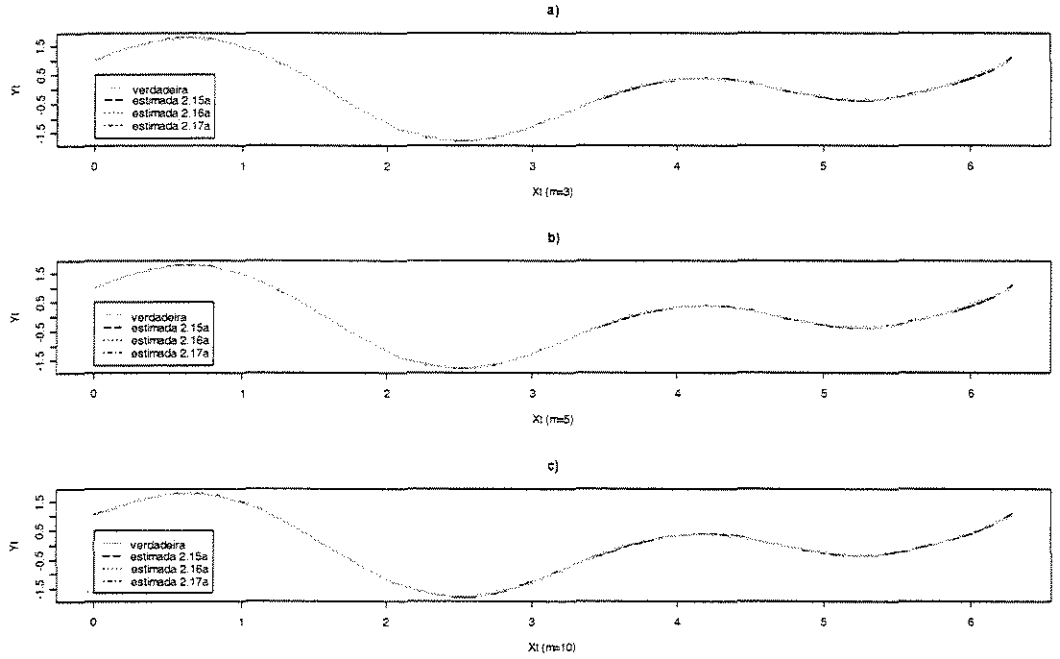


Figura 3.8: Curvas com ruído de baixa variância.

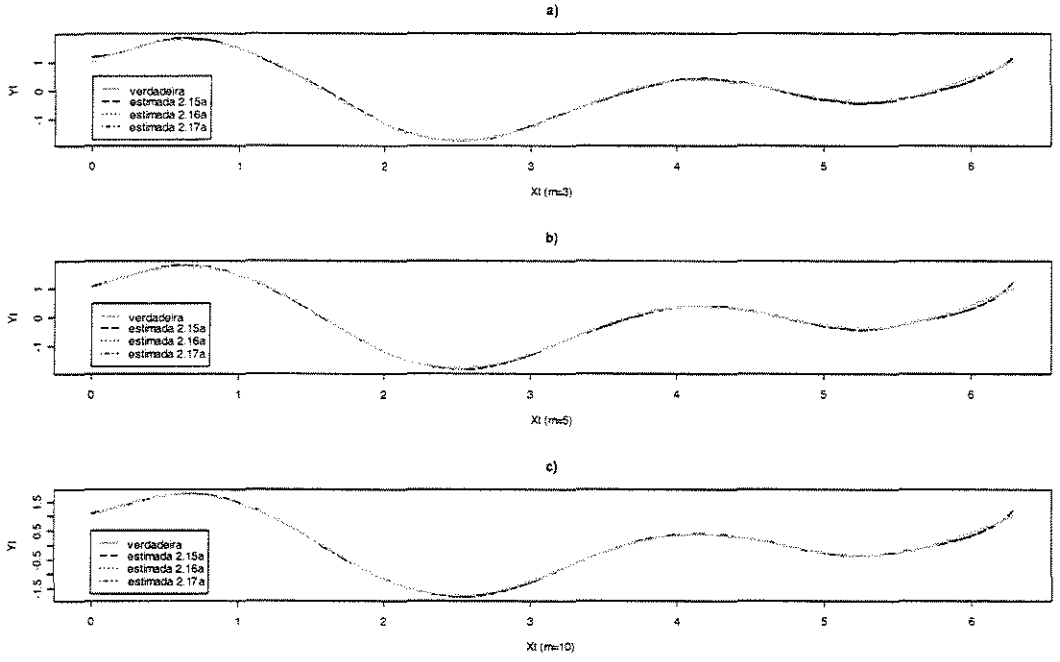


Figura 3.9: Curvas com ruído de moderada variância.

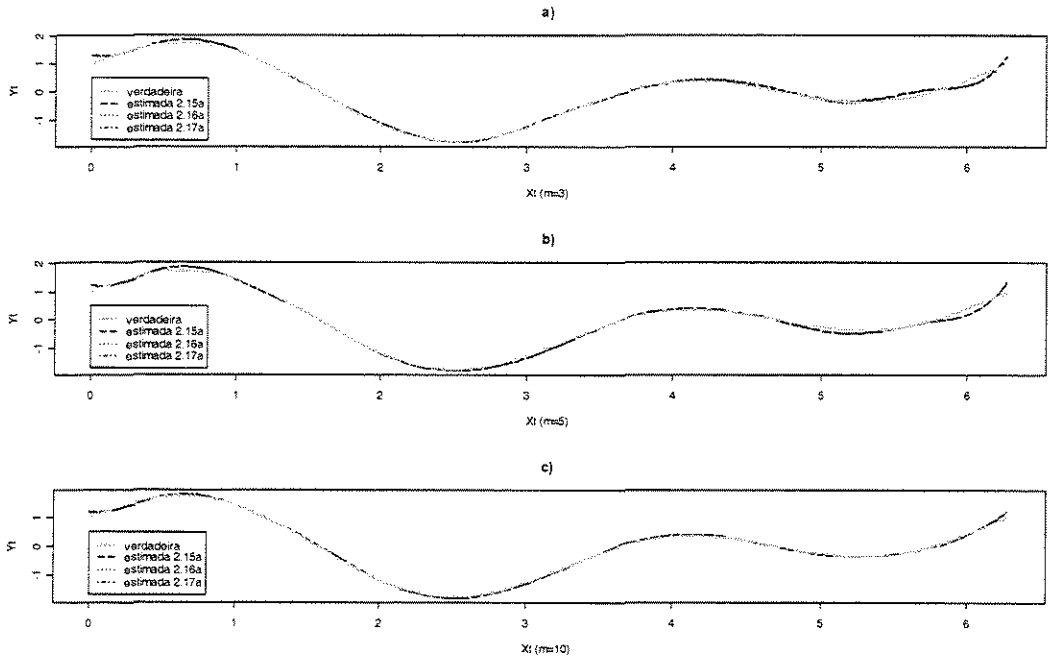


Figura 3.10: Curvas com ruído de alta variância.

para várias células da Tabela 3.2. Desconsiderando a aleatoriedade do ruído, isso ocorre porque a complexidade simples das curvas e o baixo número de curvas diminuem o número de soluções diferentes para o problema[35] (no Apêndice B, Seção B.2, constam algumas demonstrações de casos em que as medidas resumo são equivalentes).

As tabelas 3.1 e 3.2 exibem o EQM para apenas um caso de estimação. No entanto, devido à aleatoriedade das curvas simuladas, é possível que algumas estimativas sejam dependentes da semente utilizada para inicializar o gerador de números aleatórios. Pensando nisso, simulamos 100 exemplos para cada caso constante nas tabelas modificando-se apenas as sementes e, ao final, calculando o EQM médio. As figuras 3.11 e 3.12 ilustram o comportamento do EQM ao longo dos 100 exemplos para cada caso considerado. Em todos eles, o EQM médio é mais baixo para a medida (2.16a) – sua variância também é a mais baixa –, seguido do EQM médio da medida (2.15a). Ou seja, aquela forma de ponderação que considera o número de bases selecionadas na representação de cada curva é a que produziu menor EQM. A média estrutural usual, que é a medida (2.15a), produziu o segundo menor EQM em todos os casos analisados.

[35] Diminuindo a dimensão do espaço onde procuramos: $K_{máx} = 4b + 3$; veja a Equação (2.23).

Tabela 3.1: EQM para a curva da Equação (2.9).

variância do ruído	m			medida resumo
	3	5	10	
baixa	0.00043794	0.00039638	0.00038885	2.15a
	0.00042958	0.00038262	0.00038777	2.16a
	0.00044904	0.00041288	0.00039103	2.17a
moderada	0.00247203	0.0015368	0.00047427	2.15a
	0.00243231	0.0015690	0.00047547	2.16a
	0.00251961	0.0015318	0.00048777	2.17a
alta	0.01022376	0.0052438	0.0028920	2.15a
	0.01036693	0.0051877	0.0027893	2.16a
	0.01032797	0.0053522	0.0030323	2.17a

Tabela 3.2: EQM para a curva da Equação (2.10).

variância do ruído	m			medida resumo
	3	5	10	
baixa	0.00041087	0.00026237	0.00010961	2.15a
	0.00041087	0.00026237	0.00010239	2.16a
	0.00041087	0.00026237	0.00011708	2.17a
moderada	0.00112870	0.00066720	0.00037311	2.15a
	0.00124190	0.00070159	0.00035894	2.16a
	0.00102730	0.00065098	0.00039457	2.17a
alta	0.00430000	0.0014879	0.00130922	2.15a
	0.00430000	0.0016498	0.00135498	2.16a
	0.00430000	0.0013900	0.00128243	2.17a

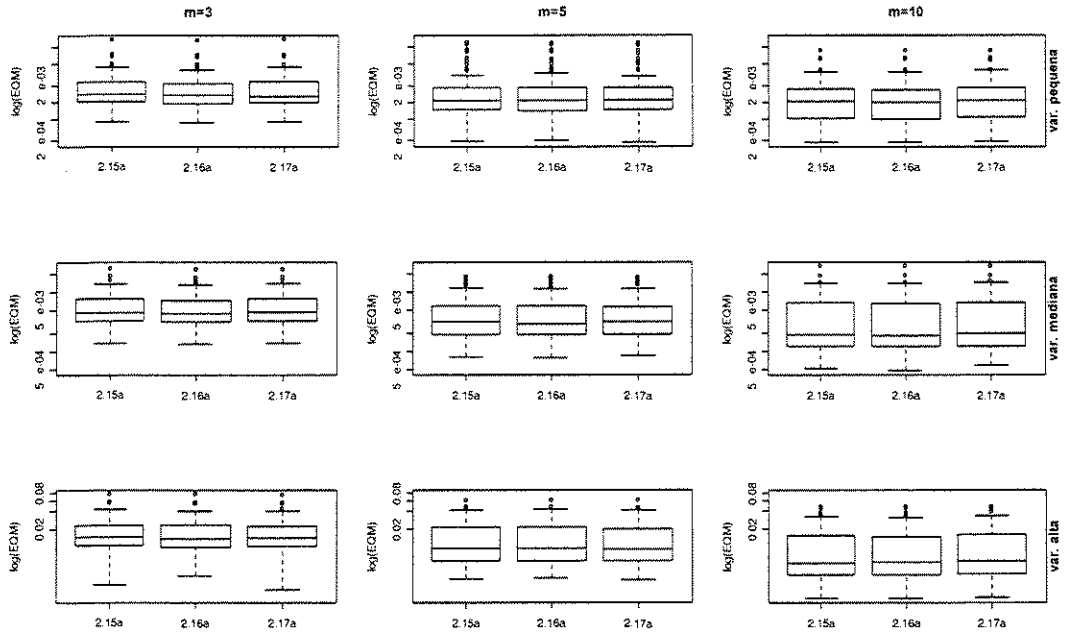


Figura 3.11: EQM de 100 exemplos para a curva da Equação (2.9).

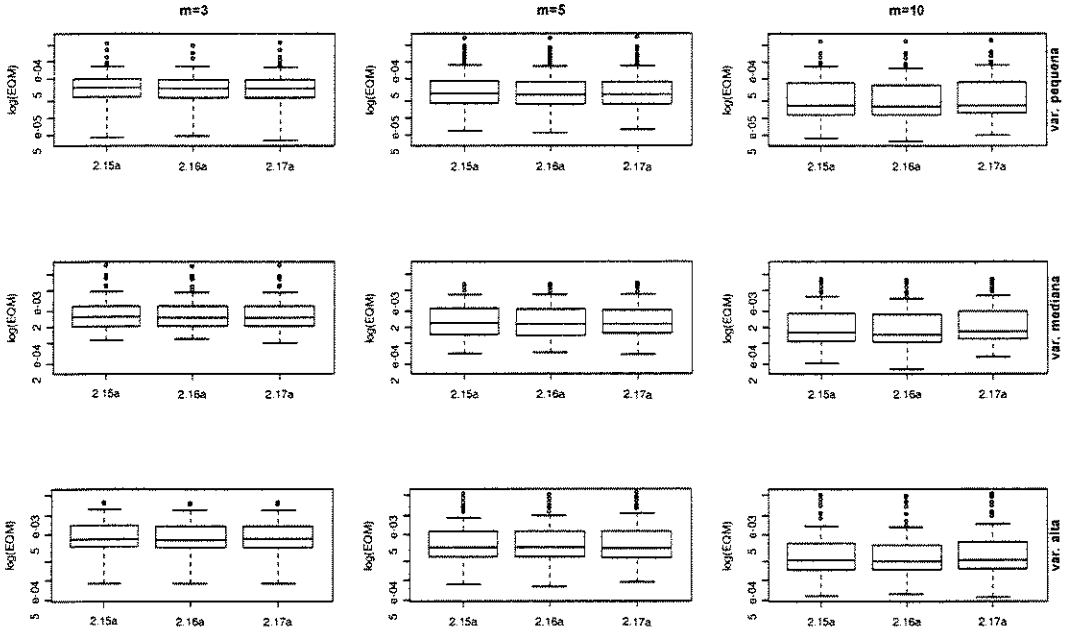


Figura 3.12: EQM de 100 exemplos para a curva da Equação (2.10).

Para uma mesma variância os gráficos foram traçados na mesma escala, para efeito de comparação. Eles demonstram que ao aumentarmos a variância do ruído, prejudicamos a estimação em termos de EQM e que quando o número de curvas na amostra aumenta, a estimação melhora em termos de EQM. É possível notar também que quando o número de curvas na amostra aumenta, diminui o EQM, mas sua dispersão aumenta e que para a curva da Equação (2.10), de forma mais simples, o EQM em geral é menor. Em todos os casos, a eficiência das medidas resumo é praticamente a mesma, pois a vantagem que a medida (2.16a) fornece em termos de EQM é pouca em relação às outras medidas.

3.5 Casos com curvas que necessitam de registro

Em todos os casos estudados até agora as curvas não necessitaram de registro para ser possível aplicar o Algoritmo 2.2, isto é, as curvas estão alinhadas de tal forma que nuances específicas de cada curva ocorrem aproximadamente sobre a mesma região ou intervalo de t. As figuras 1.3 e 1.4 dão exemplos de casos onde essa suposição não é válida.

Técnicas de registro podem ser encontradas na literatura como em, por exemplo, Ramsay and Li (1998) e Ramsay (2001a). Ramsay disponibiliza implementações dessas técnicas em pacotes para o R e o Matlab. A idéia é utilizar as implementações dessas técnicas em alguns exemplos simulados e depois aplicar o Algoritmo 2.2. O pacote em R para realizar o registro de curvas dispõe de duas técnicas, chamadas de registro por *landmark* e registro contínuo.

A técnica *landmark* considera que as curvas possuem certas propriedades visíveis ao usuário, como mínimos e máximos. Suponha dessa forma que existem Q tais propriedades[36]. O registro da curva $x_i(t)$ é então uma questão de computar uma função estritamente monótona $h_i(t)$, a função *warping*, que endireita a curva, de modo que

$$x_i[h(t_{iq})] = x_0(t_{0q}),\ q = 0, \ldots, Q+1. \tag{3.1}$$

[36] Adicionalmente, o início e o fim das curvas são incluídos como propriedades, de modo $Q+2$ propriedades são utilizadas no cômputo da função *warping*.

Por essa técnica, as curvas são alinhadas através da transformação h do tempo de modo que aquelas propriedades, chamadas marcas, ocorram nos mesmos valores dos tempos transformados. A versão em R para essa técnica é acessada pela função landmarkreg(). As marcações podem ser definidas estabelecendo-se o número de marcas por curva e clicando no gráfico com o mouse naqueles pontos que representam as saliências de cada curva, como mínimos e máximos, através da função identify().

Note que a marcação está sujeita a erros do usuário, ausência de certas saliências para algumas curvas ou mesmo ambigüidade na demarcação das mesmas. Quando o número de curvas e/ou saliências[37] é alto, ela é muito tediosa e demorada. Além disso, desenvolver algoritmos automáticos para identificação das marcas pode ter resultados enganosos (Ramsay, 2001a).

A técnica de registro contínuo idealiza resolver esse problema. Ela tenta maximizar uma medida de similaridade entre as formas das curvas, utilizando-as por completo, ao invés de apenas a informação presente em suas saliências. Uma medida de similaridade usada é

$$F(h_i) = \int_a^b \{x_i[h_i(t)] - x_0(t)\}^2 dt\,, \tag{3.2}$$

em que $[a, b]$ é o intervalo onde os dados em sua forma funcional variam. Essa medida funciona bem se a forma de h não é complexa em demasia. Contudo, ela pode falhar em casos onde as formas de x_i e x_0 diferem apenas em amplitude. O registro contínuo é obtido no R através da função registerfd(). Ela necessita de pelo menos três parâmetros. Os dois primeiros são os objetos x_i e x_0 em sua forma funcional, em que x_0 é a função referência para que as outras funções sejam registradas[38]. O terceiro indica se as curvas são periódicas ou não.

Para transformar os dados em sua forma funcional, o pacote do R disponibiliza duas funções, create.fourier.basis() e create.bspline.basis(), que transformam os dados em forma funcional através de séries de Fourier ou de B-*splines*. Infelizmente, as funções não são muito flexíveis. Por exemplo, se a opção para curvas periódicas for acionada, a forma funcional será através de séries de Fourier. Além disso, ela depende

[37] Propriedades.
[38] A idéia é alinhar as curvas a x_0.

da escolha do número de bases e a proposta aqui é que o procedimento para essa escolha seja adaptativo.

Desconsiderando essas dificuldades, suponha que escolhemos apropriadamente o número de bases e estamos querendo registrar uma coleção de curvas para então aplicar o Algoritmo 2.2. Para isso, simulamos um conjunto de $m = 3$ curvas, adicionando um ruído de baixa variância e transladando cada uma por uma constante aleatória u_{1i}, $i = 1, ..., m$ (caso mais simples de desalinhamento, mostrado na Figura 1.3). A constante aleatória para cada curva foi obtida de uma distribuição uniforme $\mathcal{U}(0,2)$. Esses valores não têm muita importância pelo fato de que as funções são periódicas. A Figura 3.13 mostra as curvas geradas e sua forma funcional expressa em termos de séries de Fourier. O número de bases usado foi seis. Aparentemente, qualquer número maior que 6 fornece bons resultados. No entanto, a quantidade de computação e espaço de memória requeridos para transformar os dados em sua forma funcional são altamente dependentes desses valores. Por exemplo, 20 bases exigem cerca de 15 vezes mais tempo de computação e 70 bases requerem que a quantidade padrão de memória disponível do R seja aumentada.

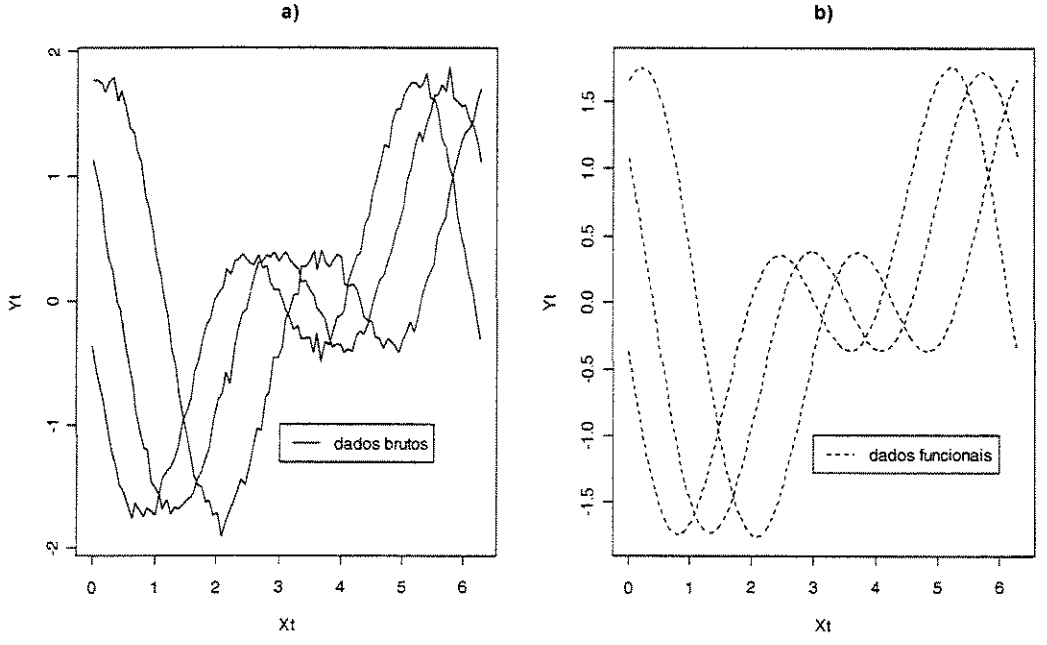


Figura 3.13: Caso simples de desalinhamento: a) Curvas simuladas e b) Curvas sob forma funcional obtida através da função create.fourier.basis().

A Figura 3.14 exibe o resultado do registro contínuo para as curvas da Figura 3.13. Note que a média estrutural difere em forma da curva verdadeira. Isso é porque o registro de duas das curvas não foi eficiente. A função referência utilizada, apenas para efeitos teóricos, foi a verdadeira curva.

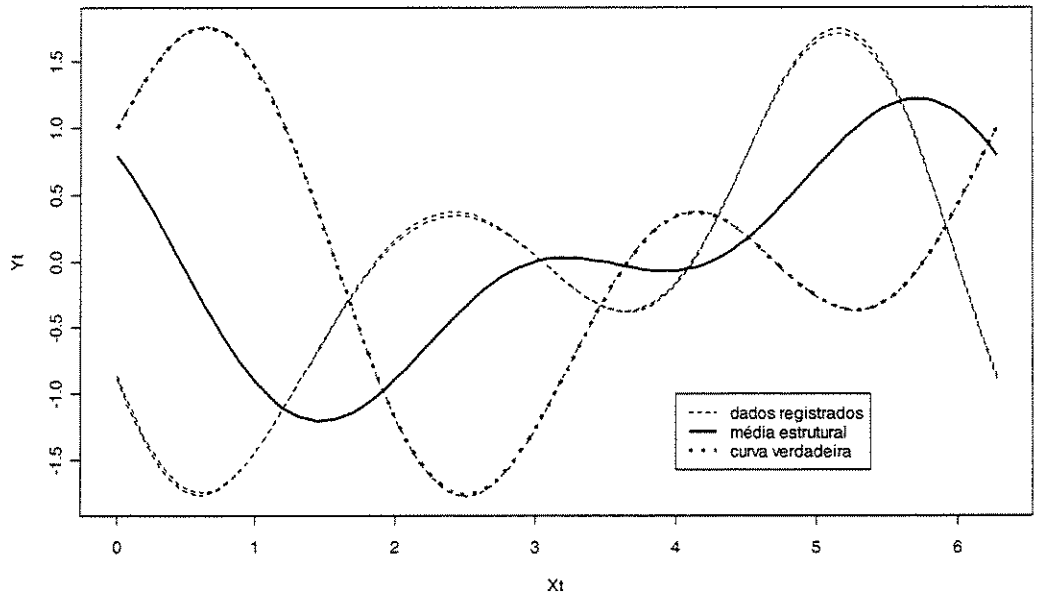


Figura 3.14: Caso simples de desalinhamento – Curvas registradas através da função registerfd().

Consideraremos dessa vez um caso menos simples de desalinhamento de curvas. As curvas diferem não apenas em fase como também em amplitude, de modo que as nuances[39] semelhantes de cada curva ocorrem em intensidades diferentes. Para simular isso, fixamos um eixo horizontal como referência e definimos como saliências as porções da curva verdadeira limitadas entre dois zeros (cruzamentos entre o eixo horizontal e a curva). A Figura 3.15 mostra o eixo escolhido $Y_t = 0$, a curva verdadeira e o que chamamos de saliências. A partir disso, para cada saliência, gere u_{2iq}, $i = 1, ..., m$; $q = 1, ..., Q$ (o número de saliências); de uma uniforme $\mathcal{U}(0.5, 1.5)$. Dessa forma, as saliências podem sofrer uma variação de amplitude que faz sua intensidade variar entre 50% e 150% do seu valor. A Figura 3.14 exibe um conjunto de três curvas geradas utilizando essa idéia adicionadas de um ruído de baixa variância. A Figura 3.17 exibe o

[39] Ou propriedades.

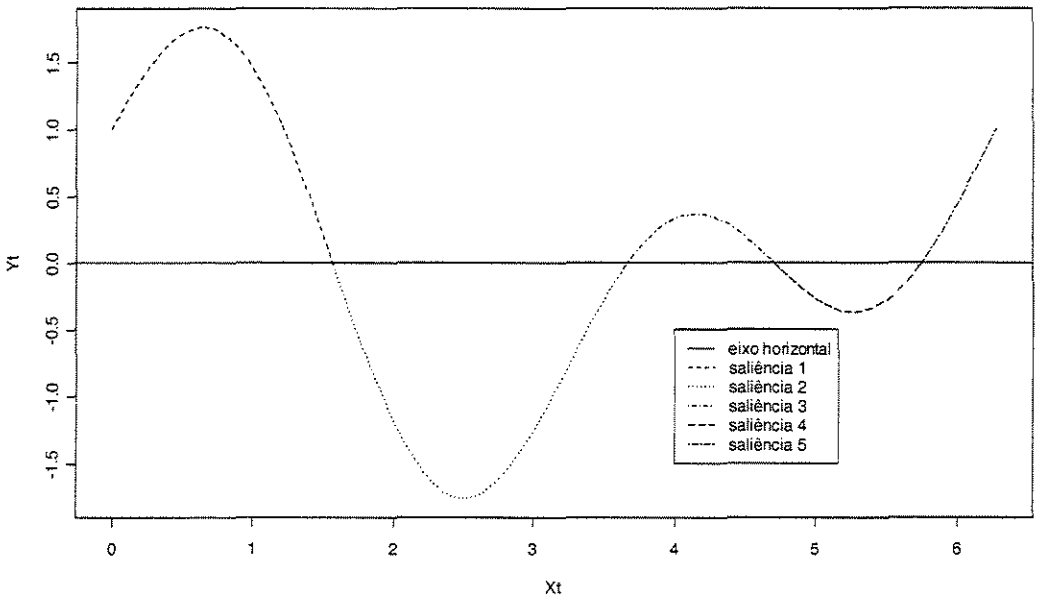


Figura 3.15: Conceito de saliência – variação de fase e de amplitude.

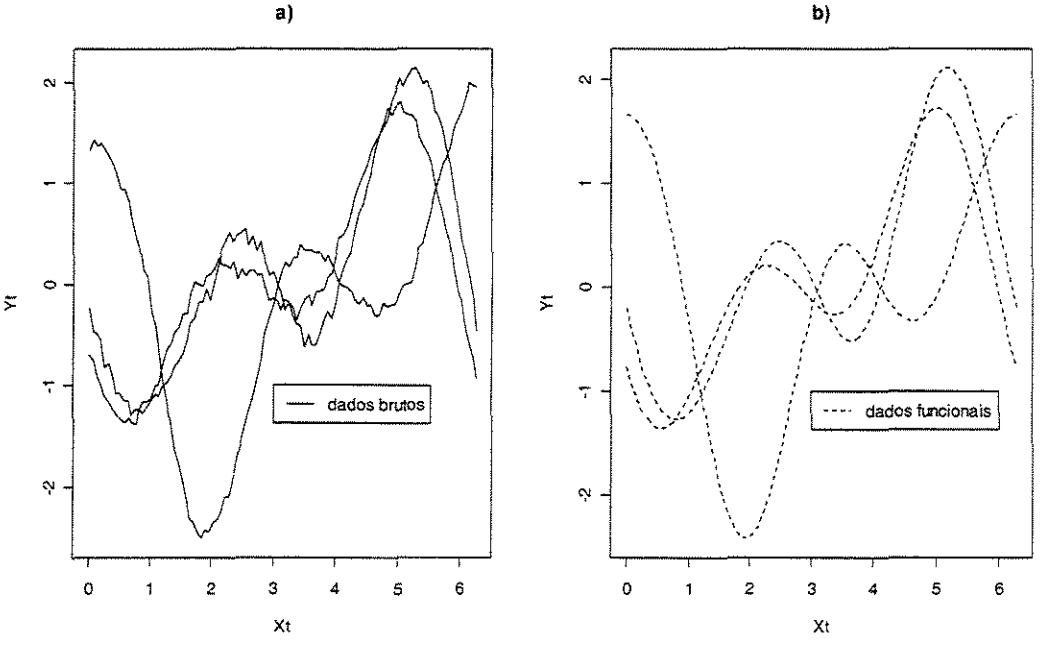


Figura 3.16: Caso complexo de desalinhamento: a) Curvas simuladas e b) Curvas sob forma funcional obtida através da função create.fourier.basis().

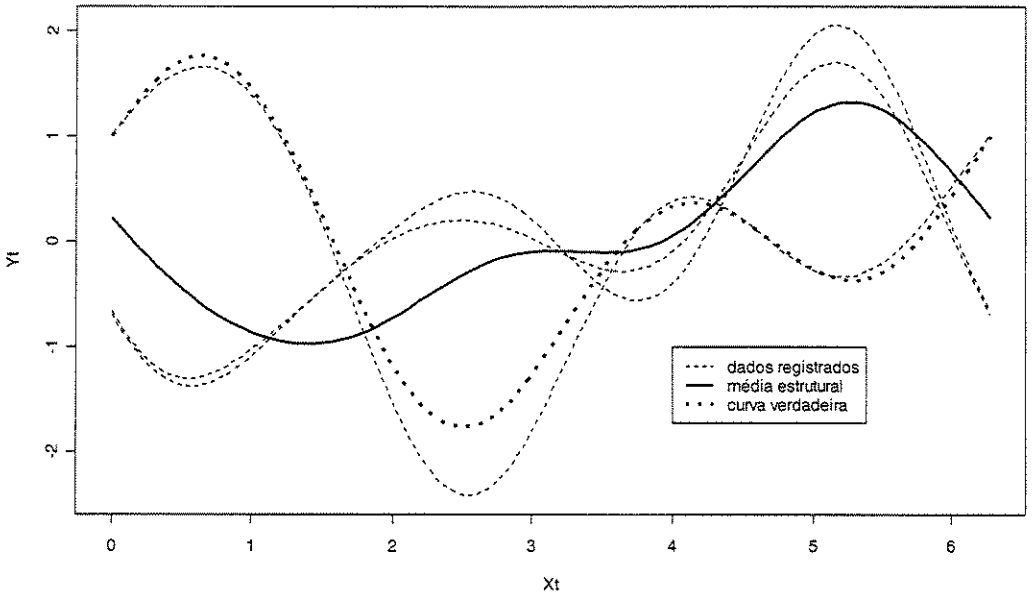


Figura 3.17: Caso complexo de desalinhamento – Curvas registradas através da função registerfd().

resultado obtido quando tentamos alinhar as curvas através da função registerfd(). Como o processo de registro falha novamente, a média estrutural difere demasiadamente da função verdadeira que gerou o processo.

Foi pensando nisso que desenvolvemos, experimentalmente, um procedimento simples que tenta registrar as curvas simuladas. Ele é baseado na Equação (3.2). No entanto, o que tentamos minimizar é a quantidade de diferença acumulada entre cada curva e uma curva tida como referência. Como as curvas possuem escalas possivelmente diferentes, nós normalizamos cada uma delas através de

$$x_i^{*} = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)}. \tag{3.3}$$

A partir disso, podemos utilizar uma idéia presente em Ramsay (2001a) que sugere substituir cada curva por sua v-ésima derivada, de modo que nosso objetivo é minimizar

$$\tilde{\delta}_i = \arg\min_{\delta_i} \int_a^b \{D^v[x_i^*(t+\delta_i) - x_0^*(t)]\}^2 dt \qquad (3.4)$$

para cada curva i, $i = 1, ..., m$. Como a derivada é distributiva com relação à adição, isso é equivalente a minimizar a diferença em área entre as derivadas de ordem v das curvas normalizadas. Para curvas periódicas com período T, é suficiente procurar por δ_i em $[0,T]$. Para nossos estudos, utilizamos $v = 0$, de modo que consideramos apenas as curvas normalizadas. Dessa forma, para efeitos de simulação, analisaremos a seguir o Algoritmo 3.1. É bom lembrar que o sucesso na obtenção de $\tilde{\delta}_i$ depende do *grid* onde realizamos a procura. Quanto mais fino esse, menor a chance de errar ao se procurar $\tilde{\delta}_i$, mas mais demorada será a busca. O *grid* que usamos consiste dos próprios pontos amostrais.

Algoritmo 3.1: Registro experimental.

0.i) Use o Algoritmo 2.2 para obter a representação funcional de cada curva. Considere para critério de parada $\hat{\sigma}^2$, a estimativa da variância do ruído;

0.ii) Normalize cada curva obtida através da Equação (3.3);

0.iii) Obtenha $\tilde{\delta}_i$, $i = 1, ..., m$, para as curvas normalizadas x_i^*;

0.iv) Translade as curvas não normalizadas fazendo $x_i(t) \leftarrow x_i(t + \tilde{\delta}_i)$.

Aplicamos o passo (0.iv) aos dados brutos pelos motivos exibidos na Seção 2.2.1 (pós-suavização). Após isso, é só aplicar o Algoritmo 2.2 de maneira usual. A Figura 3.18 exibe um conjunto de $m = 7$ curvas simuladas com variação de fase e de amplitude adicionadas de ruído de baixa variância. A Figura 3.19 mostra que o Algoritmo 3.1 foi útil para obtenção da média estrutural. Todas as três estimativas obtidas pelas equações (2.15a), (2.16a) e (2.17a) foram similares e produziram baixos valores de EQM.

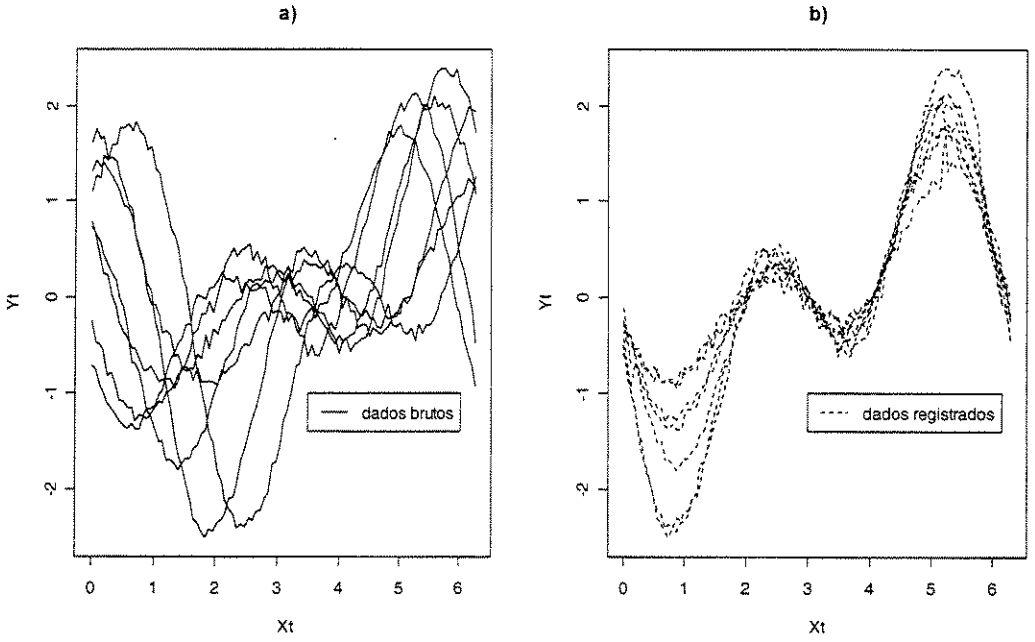


Figura 3.18: Caso complexo de desalinhamento – a) Curvas simuladas; b) Curvas registradas através do Algoritmo 3.1.

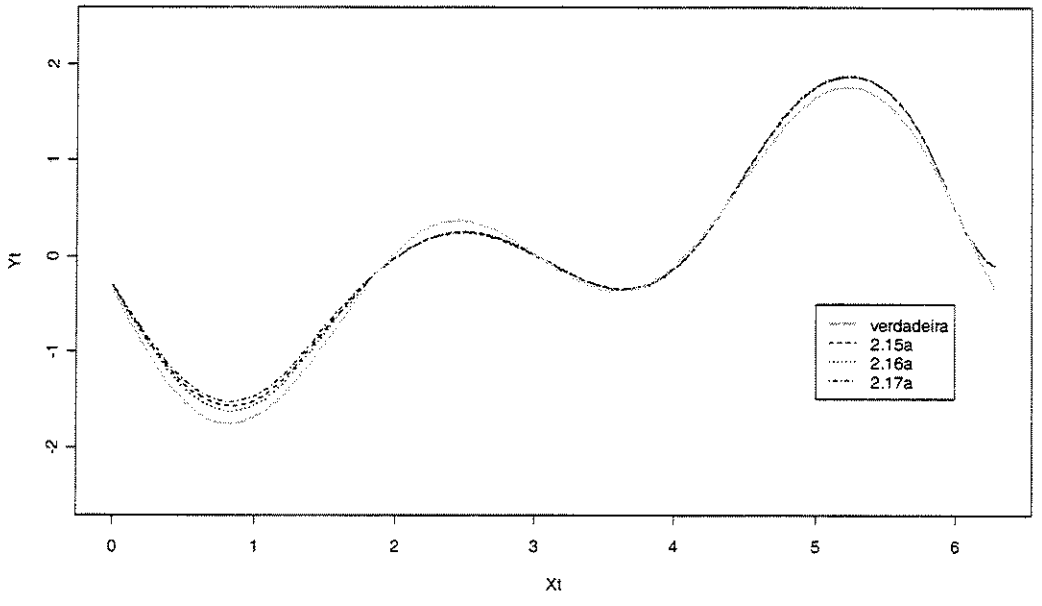


Figura 3.19: Caso complexo de desalinhamento – Aplicação do Algoritmo 2.1 às curvas da Figura 3.16b.

Considere dessa vez as curvas da Figura 3.20. Aumentamos em cerca de 25 vezes a variância do ruído e simulamos $m = 7$ curvas, modificando apenas a semente do gerador de número aleatórios[40].

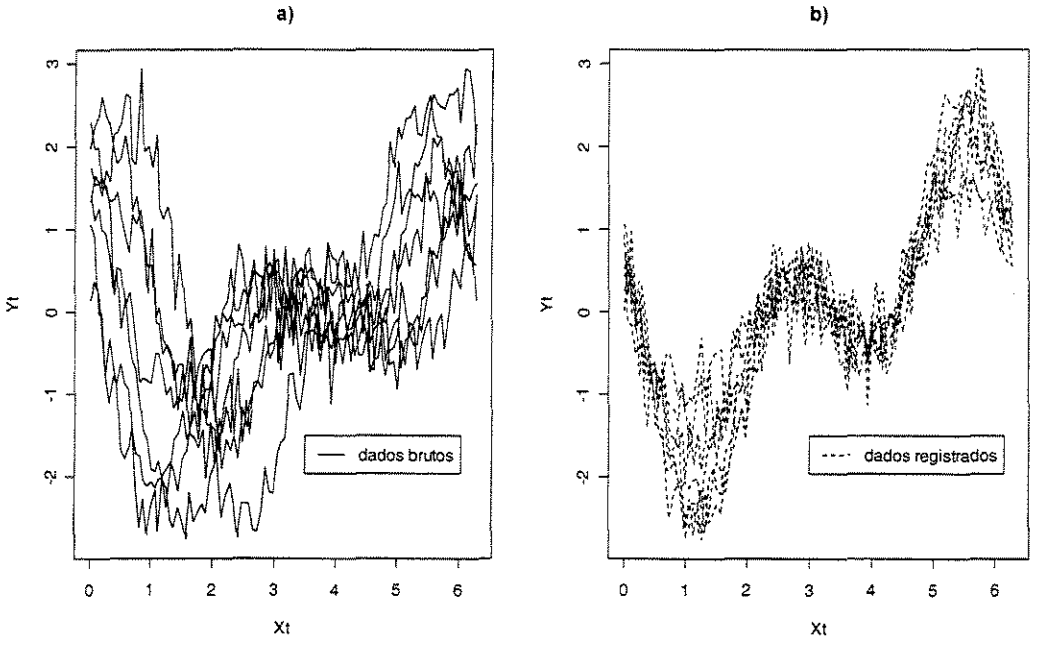


Figura 3.20: Caso complexo de desalinhamento – a) Curvas simuladas adicionadas de ruído de alta variância; b) Curvas registradas através do Algoritmo 3.1.

Ao que parece o Algoritmo 3.1 funcionou bem, apesar da dificuldade causada pelo emaranhado observado na Figura 3.20a. O resultado da aplicação do Algoritmo 2.2 está exibido na Figura 3.21. As estimativas acompanham de certa forma bem o padrão verdadeiro. É bom salientar que essa curva verdadeira não apenas sofreu simples transformações do tipo translação, como também teve cada uma de suas saliências alteradas por um fator de escala antes da adição do ruído. Ou seja, cada uma das sete curvas tem comportamento individual, de certa forma lembrando o padrão verdadeiro. Essa é a idéia de replicação presente em Análise de Dados Funcionais e necessária para aplicação da metodologia que desenvolvemos. Seja qual for das estimativas que escolhemos para utilizar no resumo das curvas, todas elas inferem ligeiramente bem o padrão por trás dos dados que queríamos descobrir.

[40] Do contrário teríamos as mesmas curvas anteriores, modificada apenas a variância do ruído.

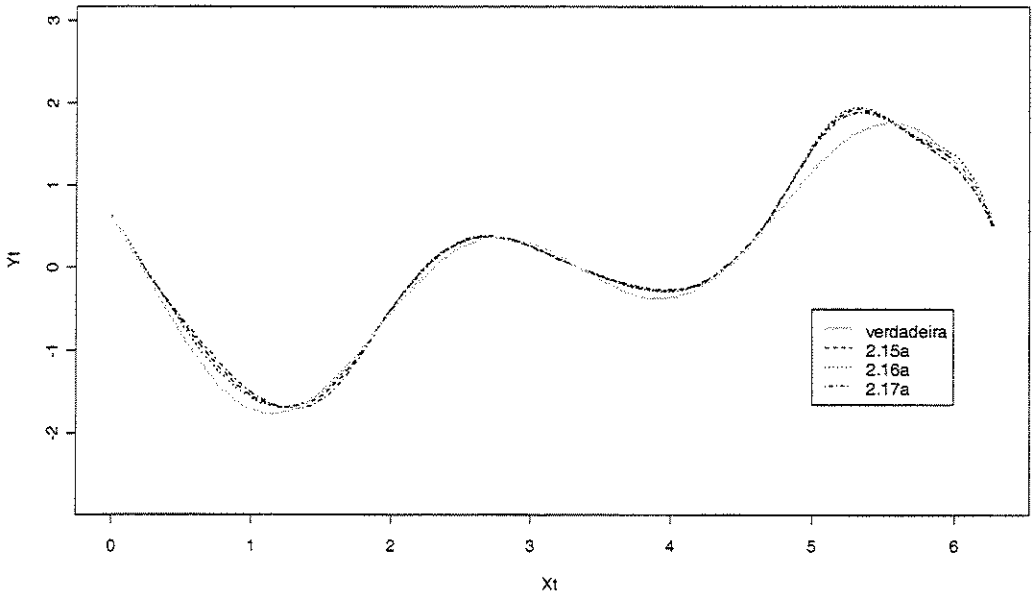


Figura 3.21: Caso complexo de desalinhamento – Aplicação do Algoritmo 2.1 às curvas da Figura 3.18b.

4 *Conclusões*

4.1 Considerações finais

Os resultados obtidos são baseados em simulações e abrangem um pequeno conjunto de possibilidades. No entanto, as análises gráficas de cada caso mencionado, bem como dos valores dos EQM obtidos, confirmaram a adaptatividade e o sucesso do método. Adicionalmente, quando aumentamos ou número de curvas ou, em contrapartida, diminuímos a variância do ruído, ou ainda, as curvas são em um certo sentido bem comportadas, a estimativa final do padrão que gerou os dados torna-se mais próxima da verdadeira curva, levando a crer na consistência do método.

Para atingir o critério de parada a implementação no *software* R não demandou tempo em exagero (quando o número de iterações para atingir a convergência foi pequeno). Contudo, a implementação no Ox de uma rotina que lide com estimação de funções de base B-*splines* foi extremamente vantajosa, chegando em alguns casos a necessitar de apenas 17% do tempo necessário para as mesmas simulações no R. Optamos por realizar todas as simulações numa mesma máquina para permitir comparações de velocidade.

Em resumo, o método mostrou-se adaptativo, de maneira que conseguiu capturar as nuances importantes de cada curva, resumindo-as na estimativa final que, nas simulações, demonstrou ser uma excelente aproximação da curva verdadeira. Adicionalmente, os algoritmos produzidos perfizeram um custo computacional relativamente baixo quando utilizados nas simulações do modelo.

Gostaríamos de agradecer ao Dr. Douglas M. Bates por sua pronta ajuda quando solicitamos o código fonte da versão da função splineDesign() usada pelo R e ao Dr. Jim O. Ramsay por sua pronta resposta nas comunicações referentes aos pacotes computacionais que lidam com a técnica de registro.

4.2 Problemas na estimação de λ

Nas implementações do Algoritmo 2.2, nada foi dito a respeito de como estimar λ. Já sabemos que estimar λ por máxima verossimilhança força (em probabilidade) a entrada de todas as bases no modelo. Dessa forma, se impuséssemos diretamente um limitante M para λ, a estimativa de máxima verossimilhança $\hat{\lambda}_{MV}$ seria igual a M, ou seja, estaríamos dizendo quantas bases queríamos no modelo e o procedimento deixaria de ser adaptativo, além de seu sucesso passar a estar sujeito à escolha do usuário.

Uma alternativa seria utilizar um critério como o GCV para seleção de λ a cada iteração. Essa tarefa exige mais tempo de computação se não dispomos de um atalho computacional. Tal atalho existe para a computação de λ na forma da Equação (1.1). No entanto, não é o nosso caso. Uma versão do critério de CV que varre os valores de λ num pequeno *grid* escolhido foi rapidamente estudada e indicou que quanto maior o valor de λ menor o critério de CV, de modo que o CV mínimo é atingido quando todas as bases são selecionadas, não modificando em nada o nosso problema.

4.3 Sugestões para trabalhos futuros

Pretende-se futuramente estudar a validade do método para outros tipos de curva e talvez sob violação de algumas das suposições do mesmo. Cogita-se ainda estudar a validade de intervalos de confiança para a estimativa final. Abaixo fornecemos a lista de algumas idéias e sugestões que surgiram ao longo da dissertação e que ainda não foram analisadas:

- Incorporar as técnicas de registro contínuo existentes na literatura ao Algoritmo 2.2 e testar sua eficácia, pois o Algoritmo 3.1 é experimental e só deve ser considerado para efeitos de simulação de dados periódicos;

- Acrescentar a restrição de que os nós extremos da matriz **B** de B-*splines* possuam mesma ordenada para os casos onde o usuário informa que os dados são periódicos. Provavelmente isso melhoraria a performance de técnicas baseadas na idéia do Algoritmo 3.1, pois a abcissa de corte para translação não atende às restrições de continuidade;

- Avaliar se a técnica LASSO[41] (Tibshirani, 1996) torna a estimação de λ automática através de uma restrição do tipo $\|\beta_i\|_1 \leq t$ e utilizando a Equação (2.6) no lugar da transformação (2.21);

- Investigar o comportamento do EQM frente à seleção de λ pelo critério GACV[42] (Xiang and Wahba, 1996) cujo tempo de computação em geral é bem menor do que sua versão exata, o GCV;

- Analisar as probabilidades de cobertura de intervalos de confiança empíricos obtidos através de bootstrap (Efron and Tibishirani, 1993) em função do número de curvas na amostra e da variância do ruído;

- Estudar estruturas de covariância entre as bases selecionadas para cada curva e covariância cruzada entre as bases selecionadas por curvas diferentes;

- Investigar critérios de parada alternativos, como por exemplo utilizar como referência a média estrutural das curvas registradas;

- Avaliar o tempo computacional e eficiência ao se estimar as curvas conjuntamente, de modo que a cada iteração da estimação de cada curva, a média estrutural funcional seja atualizada e utilizada como critério de parada (baseando-se no método *procrustes*[43]);

---

[41] *Least Absolute Shrinkage and Selection Operator.*
[42] *Generalized Approximate Cross validation.*
[43] Como referenciado em Ramsay and Silverman, 1997.

- Investigar se o número e talvez a posição das bases escolhidas para representar cada curva durante o registro são boas estimativas para utilizar no Algoritmo 2.2;

- Desenvolver a teoria que justifique a seleção empírica do vetor β.

Apêndice A

Derivadas das funções de log-verossimilhança

Exibimos aqui as derivadas da Equação (2.7) – com e sem a transformação (2.21) – que podem ser usadas pelos métodos numéricos de maximização.

A.1 Derivadas para a Equação (2.7)

Considerando apenas os termos que dependem de λ e de β, (2.7) pode ser escrita como

$$\sum_{i=1}^{m}\left\{-\frac{1}{2\sigma^2}\left\|y_i-\sum_{k=1}^{K}z_{ki}\beta_{ki}\mathbf{B}_k(x_i)\right\|^2+\sum_{k=1}^{K}\left[z_{ki}\log\frac{1-e^{-\lambda|\beta_{ki}|}}{e^{-\lambda|\beta_{ki}|}}\right]-\lambda\sum_{k=1}^{K}|\,\beta_{ki}\,|\right\}=$$

$$\sum_{i=1}^{m}\left\{-\frac{1}{2\sigma^2}\left\|y_i-\sum_{k=1}^{K}z_{ki}\beta_{ki}\mathbf{B}_k(x_i)\right\|^2+\sum_{k=1}^{K}\left[z_{ki}\log\left(1-e^{-\lambda|\beta_{ki}|}\right)\right]-\lambda\sum_{k=1}^{K}[\,|\,\beta_{ki}\,|\,(1-z_{ki})]\right\}. \quad \text{(A.1)}$$

Sejam

$$A=-\frac{1}{2\sigma^2}\sum_{i=1}^{m}\left\|y_i-\sum_{k=1}^{K}z_{ki}\beta_{ki}\mathbf{B}_k(x_i)\right\|^2, \quad \text{(A.2a)}$$

$$B = \sum_{i=1}^{m}\sum_{k=1}^{K}\left[z_{ki}\log\left(1-e^{-\lambda|\beta_{ki}|}\right)\right] \tag{A.2b}$$

e

$$C = \lambda\sum_{i=1}^{m}\sum_{k=1}^{K}[|\beta_{ki}|(1-z_{ki})], \tag{A.2c}$$

de modo que (A.1) = $A + B - C$. Isto simplificará o modo como exibiremos as derivadas.

A.1.1 Derivadas parciais em relação a λ

É imediato que

$$\frac{\partial A}{\partial \lambda} = 0; \tag{A.3a}$$

$$\frac{\partial B}{\partial \lambda} = \sum_{i=1}^{m}\sum_{k=1}^{K}\left[\frac{z_{ki}|\beta_{ki}|}{1-e^{-\lambda|\beta_{ki}|}}e^{-\lambda|\beta_{ki}|}\right]; \tag{A.3b}$$

$$\frac{\partial C}{\partial \lambda} = \sum_{i=1}^{m}\sum_{k=1}^{K}[|\beta_{ki}|(1-z_{ki})]. \tag{A.3c}$$

Desse modo,

$$\frac{\partial(2.7)}{\partial\lambda} = \frac{\partial(\text{A.1})}{\partial\lambda} = \sum_{i=1}^{m}\sum_{k=1}^{K}\left[\frac{z_{ki}|\beta_{ki}|}{1-e^{-\lambda|\beta_{ki}|}}e^{-\lambda|\beta_{ki}|}\right] - \sum_{i=1}^{m}\sum_{k=1}^{K}[|\beta_{ki}|(1-z_{ki})]. \tag{A.4}$$

A.1.2 Derivadas parciais em relação a β

Note que (A.2a) pode ser escrita (Seção 1.2.2) como

$$-\frac{1}{2\sigma^2}\sum_{i=1}^{m}\sum_{j=1}^{n}\left[y_{ij}-\sum_{k=1}^{K}z_{ki}\beta_{ki}\mathbf{B}_k(x_{ij})\right]^2. \tag{A.5}$$

Assim,

$$\begin{aligned}\frac{\partial A}{\partial \beta_{ri}}=\frac{\partial(\text{A.4})}{\partial \beta_{ri}}&=-\frac{1}{2\sigma^2}\left\{\sum_{j=1}^{n}\left[-2\left(y_{ij}-\sum_{k=1}^{K}z_{ki}\beta_{ki}\mathbf{B}_k(x_{ij})\right)z_{ri}\mathbf{B}_r(x_{ij})\right]\right\}\\&=\frac{1}{\sigma^2}\left[y_i-\sum_{k=1}^{K}z_{ki}\beta_{ki}\mathbf{B}_k(x_i)\right]^T z_{ri}\mathbf{B}_r(x_i)\\&=\frac{z_{ri}}{\sigma^2}\left[y_i-\sum_{k=1}^{K}z_{ki}\beta_{ki}\mathbf{B}_k(x_i)\right]^T\mathbf{B}_r(x_i)\\&\overset{(*)}{=}\frac{1}{\sigma^2}\left[y_i-\sum_{k=1}^{K}z_{ki}\beta_{ki}\mathbf{B}_k(x_i)\right]^T\mathbf{B}_r(x_i);\end{aligned} \tag{A.6a}$$

$$\frac{\partial B}{\partial \beta_{ri}}=\frac{\lambda z_{ri}\text{sign}(\beta_{ri})}{1-e^{-\lambda|\beta_{ri}|}}e^{-\lambda|\beta_{ri}|}\overset{(*)}{=}\frac{\lambda\text{sign}(\beta_{ri})}{1-e^{-\lambda|\beta_{ri}|}}e^{-\lambda|\beta_{ri}|}; \tag{A.6b}$$

$$\frac{\partial C}{\partial \beta_{ri}}=\lambda(1-z_{ri})\text{sign}(\beta_{ri})\overset{(*)}{=}0. \tag{A.6c}$$

As passagens assinaladas com $^{(*)}$ são válidas porque só faz sentido calcular as derivadas para as bases que são selecionadas para o modelo e, nesse caso, $z_{ri} = 1$.

Dessa forma,

$$\frac{\partial(2.7)}{\partial \beta_{ri}}=\frac{1}{\sigma^2}\left[y_i-\sum_{k=1}^{K}z_{ki}\beta_{ki}\mathbf{B}_k(x_i)\right]^T\mathbf{B}_r(x_i)+\frac{\lambda\text{sign}(\beta_{ri})}{1-e^{-\lambda|\beta_{ri}|}}e^{-\lambda|\beta_{ri}|}. \tag{A.7}$$

A.2 Derivadas para a Equação (2.7) modificada pela transformação (2.21)

Usando a mesma idéia presente em (A.1), a Equação (2.7) pode ser escrita, aplicando a transformação (2.21), como

$$\sum_{i=1}^{m}\left\{-\frac{1}{2\sigma^2}\left\|y_i-\sum_{k=1}^{K}z_{ki}\beta_{ki}\mathbf{B}_k(x_i)\right\|^2+\sum_{k=1}^{K}\left[z_{ki}\log\frac{1-e^{-\lambda\frac{|\beta_{ki}|}{\sum_{r=1}^{K}|\beta_{ri}|}}}{e^{-\lambda\frac{|\beta_{ki}|}{\sum_{r=1}^{K}|\beta_{ri}|}}}\right]\right.$$

$$\left.-\lambda\sum_{k=1}^{K}\frac{|\beta_{ki}|}{\sum_{r=1}^{K}|\beta_{ri}|}\right\}=$$

$$\sum_{i=1}^{m}\left\{-\frac{1}{2\sigma^2}\left\|y_i-\sum_{k=1}^{K}z_{ki}\beta_{ki}\mathbf{B}_k(x_i)\right\|^2+\sum_{k=1}^{K}\left[z_{ki}\log\left(1-e^{-\lambda\frac{|\beta_{ki}|}{\sum_{r=1}^{K}|\beta_{ri}|}}\right)\right]\right.$$

$$\left.-\lambda\left[1-\sum_{k=1}^{K}\left(z_{ki}\frac{|\beta_{ki}|}{\sum_{r=1}^{K}|\beta_{ri}|}\right)\right]\right\}. \qquad \text{(A.8)}$$

Sejam

$$A=-\frac{1}{2\sigma^2}\sum_{i=1}^{m}\left\|y_i-\sum_{k=1}^{K}z_{ki}\beta_{ki}\mathbf{B}_k(x_i)\right\|^2, \qquad \text{(A.9a)}$$

$$B=\sum_{i=1}^{m}\sum_{k=1}^{K}\left[z_{ki}\log\left(1-e^{-\lambda\frac{|\beta_{ki}|}{\sum_{r=1}^{K}|\beta_{ri}|}}\right)\right] \qquad \text{(A.9b)}$$

e

$$C = \lambda \sum_{i=1}^{m} \left[1 - \sum_{k=1}^{K} \left(z_{ki} \frac{|\beta_{ki}|}{\sum_{r=1}^{K} |\beta_{ri}|} \right) \right], \tag{A.9c}$$

de modo que (A.8) = $A + B - C$.

A.2.1 Derivadas parciais em relação a λ

$$\frac{\partial A}{\partial \lambda} = 0; \tag{A.10a}$$

$$\frac{\partial B}{\partial \lambda} = \sum_{i=1}^{m} \sum_{k=1}^{K} \left[\frac{z_{ki}}{1 - e^{-\lambda \frac{|\beta_{ki}|}{\sum_{r=1}^{K} |\beta_{ri}|}}} e^{-\lambda \frac{|\beta_{ki}|}{\sum_{r=1}^{K} |\beta_{ri}|}} \frac{|\beta_{ki}|}{\sum_{r=1}^{K} |\beta_{ri}|} \right]$$

$$= \sum_{i=1}^{m} \left[\frac{1}{\sum_{r=1}^{K} |\beta_{ri}|} \sum_{k=1}^{K} \left(\frac{z_{ki} |\beta_{ki}|}{1 - e^{-\lambda \frac{|\beta_{ki}|}{\sum_{r=1}^{K} |\beta_{ri}|}}} e^{-\lambda \frac{|\beta_{ki}|}{\sum_{r=1}^{K} |\beta_{ri}|}} \right) \right]; \tag{A.10b}$$

$$\frac{\partial C}{\partial \lambda} = \sum_{i=1}^{m} \left[1 - \frac{1}{\sum_{r=1}^{K} |\beta_{ri}|} \sum_{k=1}^{K} \left(z_{ki} |\beta_{ki}| \right) \right]; \tag{A.10c}$$

de maneira que

$$\frac{\partial(\text{A.8})}{\partial\lambda} = \sum_{i=1}^{m}\left[\frac{1}{\sum_{r=1}^{K}|\beta_{ri}|}\sum_{k=1}^{K}\left(\frac{z_{ki}\,|\beta_{ki}|}{1-e^{-\lambda\frac{|\beta_{ki}|}{\sum_{r=1}^{K}|\beta_{ri}|}}}e^{-\lambda\frac{|\beta_{ki}|}{\sum_{r=1}^{K}|\beta_{ri}|}}\right)\right]$$

$$-\sum_{i=1}^{m}\left[1-\frac{1}{\sum_{r=1}^{K}|\beta_{ri}|}\sum_{k=1}^{K}\left(z_{ki}\,|\beta_{ki}|\right)\right]. \qquad \text{(A.11)}$$

A.2.2 Derivadas parciais em relação a β

Como (A.9a) = (A.2a),

$$\frac{\partial A}{\partial\beta_{ri}} \overset{(**)}{=} \frac{1}{\sigma^2}\left[y_i - \sum_{k=1}^{K} z_{ki}\beta_{ki}\mathbf{B}_k(x_i)\right]^T \mathbf{B}_r(x_i). \qquad \text{(A.12)}$$

Considere B_i o valor de B referente à curva i. A derivada pode então ser obtida por

$$\frac{\partial B_i}{\partial \beta_{si}} = \begin{cases} \dfrac{z_{si}\lambda}{1-e^{-\lambda\frac{|\beta_{si}|}{\sum_{r=1}^{K}|\beta_{ri}|}}} e^{-\lambda\frac{|\beta_{si}|}{\sum_{r=1}^{K}|\beta_{ri}|}} \left[\dfrac{\operatorname{sign}(\beta_{si})\sum_{r=1}^{K}|\beta_{ri}| - |\beta_{si}|\operatorname{sign}(\beta_{si})}{\left(\sum_{r=1}^{K}|\beta_{ri}|\right)^2} \right] = \\ \dfrac{z_{si}\lambda}{\left(\sum_{r=1}^{K}|\beta_{ri}|\right)^2\left(1-e^{-\lambda\frac{|\beta_{si}|}{\sum_{r=1}^{K}|\beta_{ri}|}}\right)} e^{-\lambda\frac{|\beta_{si}|}{\sum_{r=1}^{K}|\beta_{ri}|}} \operatorname{sign}(\beta_{si})\sum_{\substack{r=1 \\ r\neq s}}^{K}|\beta_{ri}|, \quad \text{se } s = k; \\ \dfrac{z_{ki}\lambda}{1-e^{-\lambda\frac{|\beta_{ki}|}{\sum_{r=1}^{K}|\beta_{ri}|}}} e^{-\lambda\frac{|\beta_{ki}|}{\sum_{r=1}^{K}|\beta_{ri}|}} \left[\dfrac{-|\beta_{ki}|\operatorname{sign}(\beta_{si})}{\left(\sum_{r=1}^{K}|\beta_{ri}|\right)^2} \right] = \\ \dfrac{-z_{ki}\lambda|\beta_{ki}|\operatorname{sign}(\beta_{si})}{\left(\sum_{r=1}^{K}|\beta_{ri}|\right)^2\left(1-e^{-\lambda\frac{|\beta_{ki}|}{\sum_{r=1}^{K}|\beta_{ri}|}}\right)} e^{-\lambda\frac{|\beta_{ki}|}{\sum_{r=1}^{K}|\beta_{ri}|}}, \quad \text{se } s \neq k. \end{cases} \tag{A.13}$$

Portanto, somando para todas as m curvas, obtemos

$$\frac{\partial B}{\partial \beta_{si}} \overset{(**)}{=} \frac{\operatorname{sign}(\beta_{si})}{\left(\sum_{r=1}^{K}|\beta_{ri}|\right)^2} \left\{ \frac{\lambda\sum_{\substack{r=1 \\ r\neq s}}^{K}|\beta_{ri}|}{1-e^{-\lambda\frac{|\beta_{si}|}{\sum_{r=1}^{K}|\beta_{ri}|}}} e^{-\lambda\frac{|\beta_{si}|}{\sum_{r=1}^{K}|\beta_{ri}|}} - \lambda\sum_{\substack{k=1 \\ k\neq s}}^{K} \left[\frac{z_{ki}|\beta_{ki}|}{1-e^{-\lambda\frac{|\beta_{ki}|}{\sum_{r=1}^{K}|\beta_{ri}|}}} e^{-\lambda\frac{|\beta_{ki}|}{\sum_{r=1}^{K}|\beta_{ri}|}} \right] \right\}. \tag{A.14}$$

De forma análoga, considere C_i o valor de C referente à curva i. A derivada, por conseguinte, é dada por

$$\frac{\partial C_i}{\partial \beta_{si}} = \begin{cases} z_{si}\lambda\left[\dfrac{\text{sign}(\beta_{si})\sum_{r=1}^{K}|\beta_{ri}| - |\beta_{si}|\,\text{sign}(\beta_{si})}{\left(\sum_{r=1}^{K}|\beta_{ri}|\right)^2}\right], & \text{se } s = k; \\ -\dfrac{z_{ki}\lambda|\beta_{ki}|\,\text{sign}(\beta_{si})}{\left(\sum_{r=1}^{K}|\beta_{ri}|\right)^2}, & \text{se } s \neq k. \end{cases} \tag{A.15}$$

Somando para as m curvas, obtemos

$$\begin{aligned} \frac{\partial C}{\partial \beta_{si}} &= \frac{\lambda\,\text{sign}(\beta_{si})}{\left(\sum_{r=1}^{K}|\beta_{ri}|\right)^2}\left[z_{si}\sum_{\substack{r=1\\ r\neq s}}^{K}|\beta_{ri}| - \sum_{\substack{r=1\\ r\neq s}}^{K}(z_{ri}\,|\beta_{ri}|)\right] \\ &\overset{(**)}{=} \frac{\lambda\,\text{sign}(\beta_{si})}{\left(\sum_{r=1}^{K}|\beta_{ri}|\right)^2}\left\{\sum_{\substack{r=1\\ r\neq s}}^{K}[|\beta_{ri}|(1-z_{ri})]\right\}. \end{aligned} \tag{A.16}$$

A idéia das passagens marcadas com $^{(**)}$ é a mesma daquela para as passagens marcadas com $^{(*)}$, como visto antes, pois para as bases selecionadas para o modelo, $z_{si} = 1$.

Conseqüentemente,

$$\frac{\partial(\text{A.8})}{\partial\beta_{si}} = \frac{1}{\sigma^2}\left[y_i - \sum_{k=1}^{K} z_{ki}\beta_{ki}\mathbf{B}_k(x_i)\right]^T \mathbf{B}_r(x_i)$$

$$+ \frac{\text{sign}(\beta_{si})}{\left(\sum_{r=1}^{K}|\beta_{ri}|\right)^2}\left\{\frac{\lambda\sum_{\substack{r=1\\r\neq s}}^{K}|\beta_{ri}|}{1-e^{-\lambda\frac{|\beta_{si}|}{\sum_{r=1}^{K}|\beta_{ri}|}}}e^{-\lambda\frac{|\beta_{si}|}{\sum_{r=1}^{K}|\beta_{ri}|}} - \lambda\sum_{\substack{k=1\\k\neq s}}^{K}\left[\frac{z_{ki}|\beta_{ki}|}{1-e^{-\lambda\frac{|\beta_{ki}|}{\sum_{r=1}^{K}|\beta_{ri}|}}}e^{-\lambda\frac{|\beta_{ki}|}{\sum_{r=1}^{K}|\beta_{ri}|}}\right]\right\}$$

$$- \frac{\lambda\text{sign}(\beta_{si})}{\left(\sum_{r=1}^{K}|\beta_{ri}|\right)^2}\left\{\sum_{\substack{r=1\\r\neq s}}^{K}\left[|\beta_{ri}|(1-z_{ri})\right]\right\}. \qquad \text{(A.17)}$$

Apêndice B

Demonstrações

Exibimos algumas demonstrações de casos em que as medidas resumo da Seção (2.2.2) podem ser consideradas equivalentes.

B.1 Média estrutural × medidas resumo

Proposição B.1: A média estrutural das estimativas das curvas – Equação (2.19) – é um caso particular da Equação (2.18) utilizando a medida resumo (2.15a), isto é, para $\dot{\beta} = \tilde{\beta}$.

Prova:
$$\hat{Y} = \frac{1}{m}\sum_{i=1}^{m}\hat{Y}_i \qquad \text{(estimativa por inteiro)}$$
$$= \frac{1}{m}\sum_{i=1}^{m}\mathbf{B}\hat{\beta}_{Z_i}$$
$$= \mathbf{B}\frac{1}{m}\sum_{i=1}^{m}\hat{\beta}_{Z_i}$$
$$= \mathbf{B}\tilde{\beta} \qquad \text{(resumo pelos coeficientes das bases).}$$

B.2 Equivalência entre medidas resumo para alguns casos

Proposição B.2: Se a base k é selecionada por todas as curvas, as medidas resumo ponderadas são equivalentes às suas versões não ponderadas, isto é,

i) (2.15b) = (2.15a);

ii) (2.16b) = (2.16a);

iii) (2.17b) = (2.17a).

Prova: Note que $Z_{ki} = 1$, $\forall i$, de modo que $Z_k = 1_m^T$.

i) Segue direto do fato de que $\sum_{i=1}^{m} Z_{ki} = m$, de modo que os denominadores de (2.15) são iguais;

ii) $\sum_{i=1}^{m}\left[Z_{ki} \sum_{r=1}^{K} Z_{ri} \right] = \sum_{i=1}^{m}\left[1 \sum_{r=1}^{K} Z_{ri} \right] = \sum_{i=1}^{m}\sum_{r=1}^{K} Z_{ri}$, de forma que os denominadores de (2.16) são iguais;

iii) $\sum_{i=1}^{m}\left[\frac{Z_{ki}}{\sum_{r=1}^{K} Z_{ri}} \right] = \sum_{i=1}^{m} \frac{1}{\sum_{r=1}^{K} Z_{ri}}$ e conseqüentemente os denominadores de (2.17) são iguais.

Proposição B.3: Se todas as curvas possuem um mesmo conjunto de bases em suas expansões, isto é, ou a base k está na representação de todas as curvas ou não está em nenhuma representação, (2.15) = (2.16) = (2.17).

Prova: Seja K_1 o número de bases usadas na expansão de cada curva, isto é,

$$\sum_{i=1}^{m} Z_{ki} = K_1, \quad \forall i.$$

O caso em que a base k não está na representação de nenhuma curva é trivial: $\tilde{\beta}_k = \tilde{\beta}_k^{'} = \tilde{\beta}_k^{''} = 0$. Para os demais casos, segue da Proposição B.2 que as versões ponderadas são equivalentes às não ponderadas. Portanto, basta mostrar que (2.16a) = (2.15a) e que (2.17a) = (2.15a). Lembre que $n(Z_k) = m$, de maneira que

$$\tilde{\beta}_k = \frac{1}{m}\sum_{i=1}^{m}\left[Z_{ki}\hat{\beta}_{ki}\right].$$

i) (2.16) = (2.15):

$$\tilde{\beta}_k^{'} = \frac{1}{n^{'}(Z_k)}\sum_{i=1}^{m}\left[Z_{ki}\hat{\beta}_{ki}\sum_{r=1}^{K}Z_{ri}\right] = \frac{1}{\sum_{i=1}^{m}\sum_{r=1}^{K}Z_{ri}}\sum_{i=1}^{m}\left[Z_{ki}\hat{\beta}_{ki}\sum_{r=1}^{K}Z_{ri}\right]$$

$$= \frac{1}{\sum_{i=1}^{m}K_1}\sum_{i=1}^{m}\left[Z_{ki}\hat{\beta}_{ki}K_1\right] = \frac{1}{mK_1}K_1\sum_{i=1}^{m}\left[Z_{ki}\hat{\beta}_{ki}\right] = \frac{1}{m}\sum_{i=1}^{m}\left[Z_{ki}\hat{\beta}_{ki}\right] = \tilde{\beta}_k;$$

ii) (2.17) = (2.15):

$$\tilde{\beta}_k^{''} = \frac{1}{n^{''}(Z_k)}\sum_{i=1}^{m}\left[\frac{Z_{ki}\hat{\beta}_{ki}}{\sum_{r=1}^{K}Z_{ri}}\right] = \frac{1}{\sum_{i=1}^{m}\frac{1}{\sum_{r=1}^{K}Z_{ri}}}\sum_{i=1}^{m}\left[Z_{ki}\hat{\beta}_{ki}\sum_{r=1}^{K}Z_{ri}\right]$$

$$= \frac{1}{\sum_{i=1}^{m}\frac{1}{K_1}}\sum_{i=1}^{m}\left[\frac{Z_{ki}\hat{\beta}_{ki}}{K_1}\right] = \frac{K_1}{m}\frac{1}{K_1}\sum_{i=1}^{m}\left[Z_{ki}\hat{\beta}_{ki}\right] = \frac{1}{m}\sum_{i=1}^{m}\left[Z_{ki}\hat{\beta}_{ki}\right] = \tilde{\beta}_k.$$

Corolário B.3: Se todas as curvas possuem todas as bases em suas expansões, todas as medidas resumo são equivalentes.

Prova: Como $Z_i = 1_k$, $\forall i$, isso é conseqüência direta da Proposição B.3.

Proposição B.4: Se as curvas cuja expansão inclui a base k possuem um mesmo número de bases em suas expansões, as versões ponderadas são todas equivalentes, isto é, (2.15b) = (2.16b) = (2.17b).

Prova: Seja I_k o conjunto de índices das curvas cuja expansão inclui a base k. Desse modo,

$$\sum_{r=1}^{K} Z_{ri} = \begin{cases} K_1, & \text{se } i \in I_k (Z_{ki} = 1); \\ 0, & \text{se } i \notin I_k (Z_{ki} = 0). \end{cases}$$

O caso em que a base k não está na representação de nenhuma curva é trivial: $\tilde{\beta}_k = \tilde{\beta}_k^{'} = \tilde{\beta}_k^{''} = 0$. Note que

$$\tilde{\beta}_k = \frac{\sum_{i=1}^{m} \left[Z_{ki} \hat{\beta}_{ki} \right]}{n(Z_k)} = \frac{\sum_{i=1}^{m} \left[Z_{ki} \hat{\beta}_{ki} \right]}{\sum_{i=1}^{m} Z_{ki}} = \frac{\sum_{i \in I_k} \left[Z_{ki} \hat{\beta}_{ki} \right] + \sum_{i \notin I_k} \left[Z_{ki} \hat{\beta}_{ki} \right]}{\sum_{i \in I_k} Z_{ki} + \sum_{i \notin I_k} Z_{ki}} = \frac{\sum_{i \in I_k} \left[Z_{ki} \hat{\beta}_{ki} \right]}{\sum_{i \in I_k} Z_{ki}},$$

pois $Z_{ki} = 0$ se $i \notin I_k$.

i) (2.16b) = (2.15b):

$$\tilde{\beta}_k^{'} = \frac{\sum_{i=1}^{m}\left[Z_{ki}\hat{\beta}_{ki}\sum_{r=1}^{K}Z_{ri}\right]}{n^{'}(Z_k)} = \frac{\sum_{i=1}^{m}\left[Z_{ki}\hat{\beta}_{ki}\sum_{r=1}^{K}Z_{ri}\right]}{\sum_{i=1}^{m}\left[Z_{ki}\sum_{r=1}^{K}Z_{ri}\right]} =$$

$$= \frac{\sum_{i\in I_k}\left[Z_{ki}\hat{\beta}_{ki}\sum_{r=1}^{K}Z_{ri}\right] + \sum_{i\notin I_k}\left[Z_{ki}\hat{\beta}_{ki}\sum_{r=1}^{K}Z_{ri}\right]}{\sum_{i\in I_k}\left[Z_{ki}\sum_{r=1}^{K}Z_{ri}\right] + \sum_{i\notin I_k}\left[Z_{ki}\sum_{r=1}^{K}Z_{ri}\right]} = \frac{\sum_{i\in I_k}\left[Z_{ki}\hat{\beta}_{ki}\sum_{r=1}^{K}Z_{ri}\right]}{\sum_{i\in I_k}\left[Z_{ki}\sum_{r=1}^{K}Z_{ri}\right]}$$

$$= \frac{\sum_{i\in I_k}\left[Z_{ki}\hat{\beta}_{ki}K_1\right]}{\sum_{i\in I_k}\left[Z_{ki}K_1\right]} = \frac{K_1\sum_{i\in I_k}\left[Z_{ki}\hat{\beta}_{ki}\right]}{K_1\sum_{i\in I_k}\left[Z_{ki}K_1\right]} = \frac{\sum_{i\in I_k}\left[Z_{ki}\hat{\beta}_{ki}\right]}{\sum_{i\in I_k}Z_{ki}} = \tilde{\beta}_k;$$

ii) (2.17b) = (2.15b):

$$\tilde{\beta}_k^{''} = \frac{\sum_{i=1}^{m}\left[\frac{Z_{ki}\hat{\beta}_{ki}}{\sum_{r=1}^{K}Z_{ri}}\right]}{n^{''}(Z_k)} = \frac{\sum_{i=1}^{m}\left[\frac{Z_{ki}\hat{\beta}_{ki}}{\sum_{r=1}^{K}Z_{ri}}\right]}{\sum_{i=1}^{m}\left[\frac{Z_{ki}}{\sum_{r=1}^{K}Z_{ri}}\right]} = \frac{\sum_{i\in I_k}\left[\frac{Z_{ki}\hat{\beta}_{ki}}{\sum_{r=1}^{K}Z_{ri}}\right] + \sum_{i\notin I_k}\left[\frac{Z_{ki}\hat{\beta}_{ki}}{\sum_{r=1}^{K}Z_{ri}}\right]}{\sum_{i\in I_k}\frac{Z_{ki}}{\sum_{r=1}^{K}Z_{ri}} + \sum_{i\notin I_k}\frac{Z_{ki}}{\sum_{r=1}^{K}Z_{ri}}}$$

$$= \frac{\sum_{i\in I_k}\left[\frac{Z_{ki}\hat{\beta}_{ki}}{\sum_{r=1}^{K}Z_{ri}}\right]}{\sum_{i\in I_k}\frac{Z_{ki}}{\sum_{r=1}^{K}Z_{ri}}} = \frac{\sum_{i\in I_k}\left[\frac{Z_{ki}\hat{\beta}_{ki}}{K_1}\right]}{\sum_{i\in I_k}\frac{Z_{ki}}{K_1}} = \frac{\frac{1}{K_1}\sum_{i\in I_k}\left[Z_{ki}\hat{\beta}_{ki}\right]}{\frac{1}{K_1}\sum_{i\in I_k}\left[Z_{ki}K_1\right]} = \frac{\sum_{i\in I_k}\left[Z_{ki}\hat{\beta}_{ki}\right]}{\sum_{i\in I_k}Z_{ki}} = \tilde{\beta}_k.$$

Proposição B.5: Se as expansões de cada curva possuem um mesmo número de bases:

a) As versões não ponderadas das medidas resumo são equivalentes;

b) As versões ponderadas das medidas resumo são equivalentes.

Prova: Seja K_1 o número de bases que entram na expansão de cada curva, ou seja,

$$\sum_{r=1}^{K} Z_{ri} = K_1, \quad \forall i .$$

Observe que $\tilde{\beta}_k = \frac{1}{m}\sum_{i=1}^{m}\left[Z_{ki}\hat{\beta}_{ki}\right]$.

a) (2.15a) = (2.16a) = (2.17a).

i) (2.16a) = (2.15a):

$$\tilde{\beta}_k^{`} = \frac{1}{\sum_{i=1}^{m}\sum_{r=1}^{K} Z_{ri}}\sum_{i=1}^{m}\left[Z_{ki}\hat{\beta}_{ki}\sum_{r=1}^{K} Z_{ri}\right] = \frac{1}{\sum_{i=1}^{m} K_1}\sum_{i=1}^{m}\left[Z_{ki}\hat{\beta}_{ki}K_1\right] =$$

$$= \frac{1}{mK_1}K_1\sum_{i=1}^{m}\left[Z_{ki}\hat{\beta}_{ki}\right] = \frac{1}{m}\sum_{i=1}^{m}\left[Z_{ki}\hat{\beta}_{ki}\right] = \tilde{\beta}_k ;$$

ii) (2.17a) = (2.15a):

$$\tilde{\beta}_k^{``} = \frac{1}{\sum_{i=1}^{m}\frac{1}{\sum_{r=1}^{K} Z_{ri}}}\sum_{i=1}^{m}\left[\frac{Z_{ki}\hat{\beta}_{ki}}{\sum_{r=1}^{K} Z_{ri}}\right] = \frac{1}{\sum_{i=1}^{m} K_1}\sum_{i=1}^{m}\left[\frac{Z_{ki}\hat{\beta}_{ki}}{K_1}\right] =$$

$$= \frac{K_1}{m}\frac{1}{K_1}\sum_{i=1}^{m}\left[Z_{ki}\hat{\beta}_{ki}\right] = \frac{1}{m}\sum_{i=1}^{m}\left[Z_{ki}\hat{\beta}_{ki}\right] = \tilde{\beta}_k .$$

b) (2.15b) = (2.16b) = (2.17b).

Se as expansões das curvas possuem um mesmo número de bases, uma dessas situações ocorre:

i) A base k não está na representação de nenhuma curva;

ii) As curvas cuja representação inclui a base k possuem K_1 bases em sua expansão.

Em ambas as situações, decorre diretamente da Proposição B.4 que (2.16b) = (2.15b) e (2.17b) = (2.15b).

Apêndice C

Implementações

Listamos a seguir as principais implementações utilizadas durante a dissertação.

C.1 Implementações em R

A versão em R, que é mais antiga que as versões em Ox, considera o critério de parada fixo e não inclui a proposta de registro do Algoritmo 3.1.

C.1.1 Bibliotecas

Arquivo cablm.txt

```
library(splines)
#MONTA MATRIX DE B-SPLINES
bs1 <- function(x, ndx){
      knots <- as.single(sort(c(rep(range(x), 4), quantile(x, seq(0, 1,
length = ndx + 2)[ - c(1, ndx + 2)])))
      x <- as.single(x)
      spline.des(knots, x, 4, 0 * x)$design
}

#DERIVADAS PARA lambda
f1.lambda.vetorial <- function(lambda){
      aux <- diag(rep(1,K))[,Z]
```

```
}

f1.lambda.vetorial.otim <- function(lambda){
      uns <- rep(1, sum(z))
      return( -(  uns%*%log( (1-exp(-lambda*abs(beta))) / exp(-
lambda*abs(beta)) ) - uns%*%(lambda*abs(beta))  ) )
}

f1.lambda <- function(lambda){
      soma <- sum(abs(beta))
      return( -(  sum(log( (1-exp(-lambda*abs(beta)/soma)) / exp(-
lambda*abs(beta/soma)) )) - sum(lambda*abs(beta)/soma)  ) )
}

#DERIVADAS PARA beta
#nao funciona pq beta eh a incognita: beta.grande <-
diag(rep(1,K))[,Z]%*%beta
f1.beta.vetorial <- function(beta){
      aux <- y.s-B[, Z]%*%beta #AQUI REALMENTE DEVE SER MATRICIAL
      uns <- rep(1, sum(z))
      return( -(-1/(2*sigma2)*t(aux)%*%aux+uns%*%log((1-exp(-
lambda*abs(beta)))/exp(-lambda*abs(beta)))-uns%*%(lambda*abs(beta))))
}

f1.betam <- function(beta){
      soma <- sum(abs(beta))
      aux <- y.s[1,]-B[, Z]%*%beta #AQUI REALMENTE DEVE SER MATRICIAL
      return( -(-1/(2*sigma2)*sum(aux*aux)+sum(log((1-exp(-
lambda*abs(beta)/soma))/exp(-lambda*abs(beta)/soma)))-
lambda*sum(abs(beta)/soma)))
}
```

Arquivo grad.txt

```
grr.antigo <- function(betacompleta){
      aux1 <- y.s-B%*%betacompleta
      aux2 <- sum(abs(betacompleta))
      aux3 <- exp(-lambda*abs(betacompleta)/aux2)
      aux4 <- sum(aux3*abs(betacompleta)/(1-aux3))
      gr <- rep(0,K)
      for(i in 1:K) gr[i] <- 1/sigma2*sum(aux1*B[,i]) +
lambda*sign(betacompleta[i])/(aux2^2) * ( exp(-
lambda*abs(betacompleta[i])/aux2)*(aux2-betacompleta[i])/(1-exp(-
lambda*abs(betacompleta[i])/aux2)) - aux4 + exp(-
lambda*abs(betacompleta[i])/aux2)*abs(betacompleta[i])/(1-exp(-
lambda*abs(betacompleta[i])/aux2)) )
      gr
}
```

```
grr <- function(betacompleta){
      aux0 <- sign(betacompleta)
      aux1 <- abs(betacompleta)
        aux2 <- sum(aux1)
        aux3 <- exp(-lambda*aux1/aux2)
      aux4 <- aux3*aux1/(1-aux3)

      as.vector(t(t(y.s-
B%*%betacompleta)%*%B)/sigma2+lambda*(aux3*aux0*(aux2-aux1)/((1-
aux3)*aux2^2)-aux0/(aux2^2)*(sum(aux4)-aux4)))
}
```

C.1.2 Rotina principal

Arquivo noitavom_betas.txt

```
#BIBLIOTECAS DE FUNCOES
source('cab1m.txt')
source('cab1m.txt')

#PARAMETROS INICIAIS
set.seed(1)
m <- 3
M <- .1
n <- 100
epsilon <- 10^(-5)

#GERA CURVAS
A<-1
B<-2
curva<-A

if(curva==A) nbumps<-3 else nbumps<-1

if(curva==A){
      x <- seq(0, 2*pi, length=n)
      y <- cos(x)+sin(2*x)
      nbumps<-3
}else{
      x<-seq(0,6,length=n)
      y<-4*exp(-x)-cos(x/2)
      nbumps<-1
}
K<-4*nbumps+3

est.y<-y.s<-matrix(0, m, n)

#ADICIONA RUIDO
PEQUENA<-1
MEDIA<-2
```

```
GRANDE<-3
var.curva<-PEQUENA
mostra.inter<-20 #de quantas em quantas interacoes imprimimos L

if(curva==A){
      #o criterio de parada depende da variancia do ruido
      if(var.curva==GRANDE){
            var.conver<-2
            crit.parada<-.03
      }else if(var.curva==MEDIA){
            var.conver<-4
            crit.parada<-.01
      }else if(var.curva==PEQUENA){
            var.conver<-10
            crit.parada<-.005
      }
}else{
        #o criterio de parada depende da variancia do ruido
        if(var.curva==GRANDE){
                var.conver<-2
                crit.parada<-.005
        }else if(var.curva==MEDIA){
                var.conver<-4
                crit.parada<-.003
        }else if(var.curva==PEQUENA){
                var.conver<-10
                crit.parada<-.001
        }
}

for(i in 1:m) y.s[i,] <- y+rnorm(n)/var.conver
y.s<-aux1
y<-aux[1,]
ylim=c(min(y.s)-.5,max(y.s)+.5)

par(mfrow=c(2,2))
plot(x,y,type='l',col=1,ylim=ylim,xlab='t',ylab='Y(Xt)')
for(i in 1:m) lines(x,y.s[i,],type='l',col=i+1,cex=.2)
plot(x,y,type='l',ylim=ylim)
legenda<-NULL

#GERA MATRIZ DE B-SPLINES
B <- bsl(x, max(3, K-4))

beta.l<-matrix(0,K,m)
y.scl<-matrix(0,m,n)
```

```
#PARA CADA CURVA
for(l in 1:m){
      #ESTIMATIVA INICIAL
      beta <- solve(t(B)%*%B)%*%t(B)%*%y.s[l,]
      y.sc <- B%*%beta
      erro <- y.s[l,]-y.sc
      sigma2 <- 1/n*sum(erro^2)

      intervalo.lambda <- c(0+epsilon, M)
      semente.beta <- rep(1, length(beta))

      teta <- rep(.5, K)

      L <- 1
      nloops <- 1000
      parada <- 1

      #INICIA ESTIMACAO ITERATIVA
      while((L<=nloops) && (parada>crit.parada)){
            if(!(L-trunc(L/mostra.inter)*mostra.inter)) print(c('L',L))
            z <- rep(0, K)
            while(sum(z)<3){
                  for(i in 1:K) z[i] <- rbinom(1, 1, teta[i])
            }
            Z <- z*seq(z)
            zz[L,] <- z

            #ESTIMACAO DE MINIMOS QUADRADOS
            beta <- solve(t(B[, Z])%*%B[, Z])%*%t(B[, Z])%*%y.s[l,]

            # APENAS PARA sigma2 NA ESTIMACAO DE lambda
            y.sc <- B[, Z]%*%beta
            erro <- y.s[l,]-y.sc
            sigma2 <- 1/n*t(erro)%*%erro

            #lambda <- optimize(f1.lambda, intervalo.lambda)$minimum
            lambda <- 1/(1-exp(-1))
            lambda <- .5*lambda

            #ESTIMACAO DE MAXIMA VEROSSIMILHANCA
            semente.beta<-beta
            beta <- nlm(f1.betam, semente.beta)$estimate

            y.sc <- B[, Z]%*%beta
            erro <- y.s[l,]-y.sc
            sigma2 <- 1/n*t(erro)%*%erro
            teta[Z] <- 1-exp(-lambda*abs(beta))
            parada <- t(y-y.sc)%*%(y-y.sc)/n
            L <- L+1
      }
      lines(x,y.sc,col=l+1)
      legenda<-c(legenda,length(beta)+L/1000)

      est.y[l,]<-y.sc #ESTIMATIVA POR INTEIRO
```

```
      #PARA MEDIDAS RESUMO
      beta.l[,l]<-diag(rep(1,K))[,Z]%*%beta

      y.sc1[l,]<-B%*%beta.l[,l]
}
#MEDIDAS RESUMO
beta.m<-medial(beta.l)
beta.mp<-rep(0,K)
for(i in 1:K) beta.mp[i]<-sum(beta.l[i,])/max(sum(beta.l[i,]!=0),1)

beta.mpo<-rep(0,K)
num.b<-rep(0,m)
for(i in 1:m) num.b[i]<-sum(beta.l[,i]!=0)
for(i                in                1:K)                beta.mpo[i]<-
beta.l[i,]%*%num.b/max(1,((beta.l[i,]!=0)%*%num.b))

beta.mpo.<-rep(0,K)
for(i in 1:K) beta.mpo.[i]<-beta.l[i,]%*%num.b/max(1,(sum(num.b)))

beta.mpo1<-rep(0,K)
for(i in 1:K) beta.mpo1[i]<-beta.l[i,]%*%(1/num.b)/sum(1/num.b)

beta.mpo1.<-rep(0,K)
for(i in 1:K){
      aux<-beta.l[i,]!=0
      beta.mpo1.[i]<-beta.l[i,]%*%(1/num.b)
      if(sum(aux)) beta.mpo1.[i]<-beta.mpo1.[i]/(aux%*%(1/num.b))
}

y.cm<-B%*%beta.m
y.cmp<-B%*%beta.mp
y.cmpo<-B%*%beta.mpo
y.cmpo.<-B%*%beta.mpo.
y.cmpo1<-B%*%beta.mpo1
y.cmpo1.<-B%*%beta.mpo1.

#EQM
eqm<-c(sum((y-y.cm)^2)/n,sum((y-y.cmp)^2)/n,sum((y-y.cmpo)^2)/n,sum((y-
y.cmpo.)^2)/n,sum((y-y.cmpo1)^2)/n,sum((y-y.cmpo1.)^2)/n)

pp<-c(0,-.5)
if (curva==B){
      pp<-c(2,3)
}
legend(pp[1],pp[2],as.character(legenda),col=seq(2,m+1),lty=rep(1,m))

x11()
ploto(x,matrix(c(y,y.cm,y.cmp,y.cmpo,y.cmpo1),byrow=T,nrow=5))
```

C.2 Implementações em Ox

As versões em Ox são as que utilizamos para simulação. As implementações do Algoritmo 3.1 são experimentais.

C.2.1 Biblioteca

Essa é a tradução para Ox da função splineDesign() do R, de autoria de Douglas M. Bates e William (Bill) N Venables (verificar).

Arquivo splinedesign.ox

```
#include <oxstd.h>
#include <oxfloat.h>

decl sp_order,     // ordem da spline
sp_ordm1,          // ordem - 1 (3 para spline cubica)
sp_nknots,         // numero de knots
sp_curs,           // posicao no vetor de knots
sp_boundary;       // knots[curs] <= x < knots[curs+1]
                        // exceto para o caso boundary
decl sp_ldel,      // diferencas 'a esquerda em knots
sp_rdel,           // diferencas 'a direita em knots
sp_knots,          // vector de knots
sp_coeff;          // coeficientes
decl val;
decl offsets;

set_cursor(x){// NAO PRECISA RETORNAR VALOR
      decl i;

      sp_curs = 0;       // nao assume x ordenado(?)
      sp_boundary = 0;

      for(i = 0; i < sp_nknots; i++){
            if(sp_knots[i]>=x) sp_curs = i;
            if(sp_knots[i]>x) break;
    }
    if(sp_curs>sp_nknots - sp_order){
            decl lastLegit = sp_nknots - sp_order;
            if(x == sp_knots[lastLegit]){
                  sp_boundary = 1;
                  sp_curs = lastLegit;
            }
    }
```

```
    return sp_curs;
}

diff_table(x, ndiff){
      decl i;

      for(i = 0; i < ndiff; i++) {
      sp_rdel[i] = sp_knots[sp_curs + i] - x;
            sp_ldel[i] = x - sp_knots[sp_curs - (i + 1)];
      }
}

basis_funcs(x, indice){  //b eh endereco
    decl j, r;
    decl saved, term;

    diff_table(x, sp_ordm1);

      val[indice][0] = 1; // warning: indexando matriz como vetor
    for(j = 1; j <= sp_ordm1; j++){
            saved = 0;
            for(r = 0; r < j; r++){
                term = val[indice][r] / (sp_rdel[r] + sp_ldel[j - 1 -
r]);
            val[indice][r] = saved + sp_rdel[r] * term;
                saved = sp_ldel[j - 1 - r] * term;
            }
            val[indice][j] = saved;
    }
}

spline_basis(knots, order, xvals){
      decl nk, nx, i, j;
    decl kk, xx;

      kk = knots;
      nk = columns(knots);
      xx = xvals;
      nx = columns(xvals);

      sp_order = order;
      sp_ordm1 = order - 1;
      sp_knots = kk;
      sp_nknots = nk;

      sp_rdel = sp_ldel = zeros(1, sp_ordm1);
      val = zeros(nx, sp_order);
      offsets = zeros(1, nx);

      for(i = 0; i < nx; i++){
            set_cursor(xx[i]);
            offsets[i] = sp_curs - sp_order;

            if (sp_curs < sp_order || sp_curs > (nk - sp_order)){
            for (j = 0; j < sp_order; j++) {
```

```
                        val[i][j] = M_NAN; // ou < . > (NaN)
                    }
                }
                else{
                    basis_funcs(xx[i], i); // aqui estah tentando enviar
endereco
                }
        }

        return val;
}

splineDesign(knots, x, ord){
      decl i;
      decl nk, nx;

      knots = sortr(knots); // ordena
      if((nk = columns(knots)) <= 0){
            print("Número de knots deve ser pelo menos a ordem\n");
            exit(1);
      }

      nx = columns(x);
      if(ord > nk || ord <1){
            print("Ordem deve estar entre 1 e o número de nknots\n");
            exit(1);
      }

      if(any(x .< knots[ord-1]) || any(x .> knots[nk+1-ord-1])){
            print("Os valores de x devem estar no intervalo
[knots[ordem], knots[nknots+1-ordem]]\n");
            exit(1);
      }
      decl temp = spline_basis(knots, ord, x);

      //PONDO B NO LUGAR REFERENTE AOS PONTOS (transforma em matriz
bloco-diagonal - pode ser otimizado)
      decl ncoef = nk - ord;
      decl design = zeros(nx,ncoef);
      decl ind=range(1,nx);
      for(i=1;i<ord;i++){
            ind=ind~range(1,nx);
      }
      ind=sortr(ind);
      for(i=0;i<columns(offsets);i++){
            ind=ind~offsets[i]+range(1,ord);
      }
      //shape pode servir para repetir e otimizar acima
      ind=(shape(ind,nx*ord,2));

      //temp eh nx x ord
      temp = shape(temp',1,nx*ord);
      decl j=0;
      for(i=0;i<nx*ord;i++){
            design[ind[i][0]-1][ind[i][1]-1] = temp[j++];
```

```
    }

    return design;

}
```

C.2.2 Rotina principal

Essa versão já considera o critério de parada flexível e pode fornecer, se especificado, os EQM para vários casos (como foi feito nas figuras 3.11 e 3.12).

Arquivo bs1win.ox

```
#include <oxstd.h>
#include <oxfloat.h>
#include <oxprob.h>
#import <maximize>
//#include </usr/local/lib/ox-3.20/packages/gnudraw/gnudraw.h>
//#include <h:/gnudraw/packages/gnudraw/gnudraw.h>
#include <../packages/gnudraw/gnudraw.h>
#include <splinedesign.ox>

enum {ESTIMADAS, BRUTAS, EMR};
enum {PEQUENA, MEDIA, GRANDE};
enum {CA, CB};
const decl opcao = EMR;
const decl var_curva = 2;
const decl curva = CB;
const decl var_conver = <10, 4, 2; 10, 4, 2>;  //1 / desvio-padrao do
ruido;

    //linhas: curvas; colunas: magnitude da variancia
const decl crit_parada = <.005, .01, .03; .001, .003, .005>;

decl K, in, sigma2, lambda;
decl y_s, B, Z, l;

// log-verossimilhanca
fmv(const vBeta, const adFuncao, const avDerivadas, const amHessiana){
    decl ci;
    decl dsoma = sumc(fabs(vBeta));
    decl vaux = y_s[l][]'-B[][Z]*vBeta;

    adFuncao[0] = -1/(2*sigma2)*sumc(vaux.^2) + sumc(log((1-exp(-
lambda*fabs(vBeta)/dsoma)) ./ exp(-lambda*fabs(vBeta)/dsoma))) -
lambda*sumc(fabs(vBeta)/dsoma);
```

```
        if(avDerivadas){
                decl vaux0 = vBeta .> 0 .? 1 .: vBeta .< 0 .? -1 .: 0; //(a
.> 0) - (a .< 0);
                decl vaux1 = fabs(vBeta);
                decl daux2 = sumc(vaux1);
                decl vaux3 = exp(-lambda*vaux1/daux2);
                decl vaux4 = vaux3.*vaux1./(1-vaux3);

                avDerivadas[0] = B[][Z]'*vaux/sigma2 +
lambda*(vaux3.*vaux0.*(daux2-vaux1)./((1-vaux3)*daux2^2)-
vaux0/(daux2^2).*(sumc(vaux4)-vaux4));
        }

        return 1; // SUCESSO
}

main(){
        decl knots;
        decl nbumps, ndx, ordem = 4;
        decl n;
        decl m, M, epsilon;
        decl x, y;
        decl est_y, y_sc;
        decl i, r, L, nloops;
        decl erro, teta, beta;
        decl z;
        decl semente_beta, intervalo_lambda;
        decl Lambda, Dim_beta, Parada, tetas, Sigma2, parada, max_beta;
        decl pos, betamv, valormaximo;
        decl beta_1, eqm, est_media;
        decl repita, tol;
        decl beta_cm, beta_cmp, beta_cmpo, beta_cmpo_, beta_cmpo1,
beta_cmpo1_, num_b;
        decl y_cm, y_cmp, y_cmpo, y_cmpo_, y_cmpo1, y_cmpo1_;
        decl y_c;
        decl tempo=timer();
        format("%#7.7g");

        n = 100;

        m = 10;
        M = .1;
        epsilon = 10^(-5);

        if(curva==CA){
                x = range(0, M_2PI, (M_2PI-0)/(n-1));
                y = cos(x) + sin(2*x);
                nbumps = 3;
        }
        else if(curva==CB){
                x = range(0, 6, (6-0)/(n-1));
                y = 4 * exp(-x) - cos(x / 2);
                nbumps = 1;
        }else{
                print("Codigo de curva invalido\n");
```

```
            exit(1);
      }

      K = 4 * nbumps + 3;

      ndx = max(3, K-4);
      knots = constant(min(x), 1, 3)~quantiler(x, range(0, 1, 1/(ndx +
2 - 1)))~constant(max(x), 1, 3);

      // monta matriz de B-splines
      B = splineDesign(knots, x, ordem);

      decl R = 1; // R>1 apenas quando queremos calcular o EQM para
varios casos
      eqm = zeros(6, R); // eqm para as medidas resumo
      decl conv = zeros(m, R);
      for(r=0;r<R;r++){ // fixa caso e aleatoriza semente
            println("r=",r);
            ranseed(r+1);

            // gera curvas
            y_s = y + rann(m, n) / var_conver[curva][var_curva];
            beta_1=zeros(K, m);

            //para cada curva
            for(l=0; l < m; l++){
                  println("l=",l);
                  repita=1; tol=1;

                  while(repita){ // criterio de parada flexivel
                        println("tol=",tol);

                        // estimacao inicial
                        beta = invertsym(B'*B)*B'*y_s[l][]';
            y_sc = (B*beta)';
                  erro = y_s[l][] - y_sc;
                    sigma2 = 1 / n * sumr(erro.^2);

                        intervalo_lambda = constant(0 + epsilon, 1, M);
                        semente_beta = constant(1, 1, columns(beta));

                        teta = constant(.5, 1, K);

                        L = 0;
                        nloops = 1000;
                        Lambda = Dim_beta = Parada = Sigma2 = zeros(1,
nloops);

                        parada = 1;
                        max_beta = 0;
                        tetas = zeros(nloops, K);

                        while((L < nloops) &&
(parada>crit_parada[curva][var_curva]*tol)){
                              if(!(imod(L,50))) println("L=",L);
```

```
                                z = zeros(1, K);
                                while(sumr(z) < 3)
                                      for(i = 0; i < K; i++) z[i] =
ranbinomial(1, 1, 1, teta[i]);
//                              z = zz[L][];

                                Dim_beta[L] = sumr(z);

                                pos = range(1, K);
                                Z = selectc(pos, z .* (pos)) - 1;

                                // estimacao via minimos quadrados
                                beta =
invertsym(B[][Z]'*B[][Z])*B[][Z]'*y_s[1][]';

                                //# APENAS PARA sigma2 NA ESTIMACAO DE
lambda
                          y_sc = (B[][Z]*beta)';
                              erro = y_s[1][] - y_sc;
                    sigma2 = 1 / n * sumr(erro.^2);
                                //# testar transposicao

                                lambda = 1/(1-exp(-1));       //#pode ser
delaracao externa ao laco
                                lambda = .5 * lambda;
                                Lambda[L] = lambda;
                                semente_beta = beta;

                                // estimacao via maxima verossimilhanca
                                MaxControl(2000, 0);    //num maximo de
iteracoes (default eh 1000), imprime resultado a cada 50 iteracoes
(default eh 0)

                                betamv = semente_beta;
                                MaxBFGS(fmv, &betamv, &valormaximo, 0,
0);

                        y_sc = (B[][Z]*betamv)';
                          erro = y_s[1][] - y_sc;

                                sigma2 = 1 / n * sumr(erro.^2);

                                Sigma2[L] = sigma2;

                                teta[Z] = 1 - exp(-lambda *
fabs(betamv));
                                tetas[L][] = teta;

                                parada = sumr((y-y_sc).^2) / n;

                                Parada[L] = parada;
                                L++;
                          }
                          println("L=",L-1);
```

```
      if(parada<=(crit_parada[curva][var_curva]*tol)){
                              repita=0;
                              conv[l][r]=1;
                        }
                        else{
                              tol*=1.3;
                        }
                        //repita=0; // siga sempre, para criterio de
parada fixo
                  }
                  println("sumz",sumr(z));
                  beta_l[][l] = diag(ones(1, K))[][Z]*betamv; //expande
dimensao de beta
                  // est_y[l][] = y_sc; //nao precisamos mais
            }

            // medidas resumo
            beta_cm = meanr(beta_l);

            //est_media = meanc(est_y); // nao precisamos mais
            beta_cmp = zeros(K,1);
            for(i=0; i<K; i++) beta_cmp[i] =
sumr(beta_l[i][])/max(1~sumr(beta_l[i][].!=0));

            beta_cmpo = zeros(K,1);
            num_b = zeros(m, 1);
            for (i = 0; i < m; i++) num_b[i] = sumc(beta_l[][i].!=0);
            for(i=0; i<K; i++) beta_cmpo[i] = beta_l[i][]*num_b /
max(1~(beta_l[i][].!=0)*num_b);

            beta_cmpo_ = zeros(K,1);
            for(i=0; i<K; i++) beta_cmpo_[i] = beta_l[i][]*num_b /
max(1~sumc(num_b));

            beta_cmpo1 = zeros(K,1);
            for(i=0; i<K; i++) beta_cmpo1[i] =
beta_l[i][]*sumr(num_b.^(-1)) / sumc(num_b.^(-1));

            beta_cmpo1_ = zeros(K,1);
            decl aux;
            for(i=0; i<K; i++){
                  aux=beta_l[i][].!=0;
                  beta_cmpo1_[i] = beta_l[i][]*sumr(num_b.^(-1));
                  if(sumr(aux)) beta_cmpo1_[i] /= aux*  (num_b.^(-1));
//                beta_cmpo1_[i] = beta_l[i][]*sumr(num_b.^(-1)) /
(beta_l[i][].!=0)*(num_b.^(-1)));
            }

            y_c =
(B*beta_cm)'|(B*beta_cmp)'|(B*beta_cmpo_)'|(B*beta_cmpo)'|(B*beta_cmpo1
)'|(B*beta_cmpo1_)';
```

```
            for(i=0;i<6;i++){
                  eqm[i][r]=sumr((y-y_c[i][]).^2)/n;
            }

            if(R==1){
                  print("e",sumr((meanc(y_s)-y_c[0][]).^2)/n);
                  print(eqm[<0,2,4>]);
            }

            //Draw(0,y|est_y[][]); //desenha todas estimadas de cada
curva
            //ShowDrawWindow();
            //  CloseDrawWindow();
      }
      // para graficos no R
      decl eqmm=meanr(eqm);
      decl somaconv=sumc(conv);

      decl eqmp=(eqm*((somaconv.==m)'))/sumr(somaconv.==m);
      println("tempo",(timer()-tempo)/100," seg");
}
```

C.2.3 Registro experimental

Essa versão é uma implementação experimental do Algoritmo 3.1. Listamos apenas as partes de bs1win.ox que precisam ser modificadas, substituindo com . . . o que permanece inalterado.

Arquivo bs1reg.ox

```
      // PASSO 0.1
      // OBTEM FORMA FUNCIONAL DAS CURVAS ATRAVES DO ALGORITMO 2.2
      decl R = 1;
      eqm = zeros(6, R);
      decl conv = zeros(m, R);
      for(r=0;r<R;r++){
            println("r=",r);
            y_s = loadmat("yreg2.mat"); // NOME DO ARQUIVO COM CURVAS A
SEREM REGISTRADAS

            beta_1=zeros(K, m);
            y_c = zeros(m, n);
            for(l=0; l < m; l++){
                  println("l=",l);
                  repita=1; tol=1;
                  while(repita){
                        println("tol=",tol);
```

```
                        .
                        .
                        .

                        while((L < nloops) &&
(parada>crit_parada[curva][var_curva]*tol/5)){
                              if(!(imod(L,50))) println("L=",L);
                              .
                              .
                              .
                              while(sumr(z) < 3) for(i = 0; i < K; i++)
z[i] = ranbinomial(1, 1, teta[i]);
                              pos = range(1, K);
                              Z = selectc(pos, z .* (pos)) - 1;

                              beta =
invertsym(B[][Z]'*B[][Z])*B[][Z]'*y_s[l][]';

                              //# APENAS PARA sigma2 NA ESTIMACAO DE
lambda
                        y_sc = (B[][Z]*beta)';
                              .
                              .
                              .
                              L++;
                        }
                        println("L=",L-1);
                        .
                        .
                        .
                  }
                  println("sumz",sumr(z));
                  beta_l[][l] = diag(ones(1, K))[][Z]*betamv; //expande
dimensao de beta
                  y_c[l][] = y_sc; //nao precisamos mais
            }

            // PASSO 0.2
            // NORMALIZA AS VERSOES FUNCIONAIS
            y_cn = ((y_c'-minc(y_c')).//(maxc(y_c')-minc(y_c')))';

            decl y_dados=y_s;

            // PASSOS 0.3 E 0.4
            decl landmark = y_cn[0][];
            decl delta=zeros(m,1);
            decl minimo;
            decl j;
            decl aux1;
            decl dif;
            decl y_cns=zeros(m,n);
            for(i=0;i<m;i++){
                  // OBTEM delta DA EQUACAO (3.4)
                  minimo=sumr((y_cn[i][]-landmark).^2);
                  delta[i]=0;
```

```
                for(j=0;j<n-1;j++){

      aux1=dropc(y_cn[i][],range(0,j))~dropc(y_cn[i][],range(j+1,n-1));
                        dif=sumr((aux1-landmark).^2);
                        if(dif<minimo){
                              minimo=dif;
                              delta[i]=j+1;
                        }
                }
                // TRANSLADA AS CURVAS NAO NORMALIZADAS
                if(!delta[i]){
                        y_cns[i][]=y_cn[i][];
                }
                else{
                        y_cns[i][]=dropc(y_cn[i][],range(0,delta[i]-
1))~dropc(y_cn[i][],range(delta[i],n-1));
                        y_s[i][]=dropc(y_s[i][],range(0,delta[i]-
1))~dropc(y_s[i][],range(delta[i],n-1));
                }
          }
          //y_s estah registrado

          /*Draw(0,y_c);
          Draw(1,y_cn);
          Draw(2,y_cns);
          ShowDrawWindow();*/

          y_sn = ((y_s'-minc(y_s')./(maxc(y_s')-minc(y_s')))';
//dados normalizados
          median = meanr(y_sn); // media estrutural normalizada

          // APLIQUE O ALGORITMO 2.2 (PROGRAMA bs1win.ox)
          beta_1=zeros(K, m);
          y_c = zeros(m, n);
          for(l=0; l < m; l++){
                println("l=",l);
                repita=1; tol=1;
                while(repita){
                        println("tol=",tol);
                        .
                        .
                        .
                        while((L < nloops) &&
(parada>crit_parada[curva][var_curva]*tol*5)){
                                if(!(imod(L,50))) println("L=",L);
                                .
                                .
                                .
                                L++;
                        }
                        .
                        .
                        .
                }
```

```
                    .
                    .
                    .
                }
                // medidas resumo
                .
                .
                .
                Draw(0,y_c|y);
                Draw(1,y_s|y);
                Draw(2,y_dados|y);
                y_c =
(B*beta_cm)'|(B*beta_cmp)'|(B*beta_cmpo_)'|(B*beta_cmpo)'|(B*beta_cmpo1
)'|(B*beta_cmpo1_)';
                Draw(3,y_c|median|y);
                ShowDrawWindow();
                .
                .
                .
        }
        .
        .
        .
}
```

C.3 Implementação em Matlab

Essa é a versão em Matlab para o cálculo da matriz de *B-splines*. Ela foi retirada de Eilers and Marx (1996) e considera apenas o caso particular em que o *grid* de nós é uniforme (eqüiespaçado). Nós a utilizamos apenas para avaliação da velocidade de cálculo em comparação com o Ox e o R e da precisão numérica em comparação com esses *softwares*, tendo como referência a precisão da versão em Maple. Como ela apresentou alguns erros quando executando na versão 6.1 do Matlab (devido a algumas multiplicações inválidas entre matrizes), apresentamos a versão modificada a seguir.

Não garantimos que a versão modificada nem a original calculem corretamente a matriz de B-*splines*. O que fizemos foi apenas para garantir que o programa forneça uma matriz tetra-diagonal tanto no Matlab quanto em outras plataformas para comparação da estabilidade numérica e tempo de computação de um mesmo algoritmo. Como já foi comentado, o algoritmo que testamos e confirmamos sua corretude é aquele da versãos em R e a nossa tradução para Ox, ambos com *grid* variável.

Arquivo bsmatlab.txt

```
function B = bspline(x, xl, xr, ndx, bdeg)
      dx = (xr - xl) / ndx; % O GRID UNIFORME; xl = min(x), lr = max(x)
      t = xl + dx * [-bdeg:ndx-1];
      T = ones(size(x))' * t; % O INICIO DE CADA INTERVALO
      X = x' * ones(size(t)); % VALORES EM QUE AVALIAMOS A MATRIZ
      P = (X - T) / dx; % OS COEFICIENTES DA RECURSAO
      B = (T <= X) & (X < (T+dx)); % EM QUE INTERVALO ESTAMOS AVALIANDO
      r = [2:length(t) 1];
      for k = 1:bdeg % PARA O GRAU ESPECIFICADO (3 NO NOSSO CASO)
            B = (P .* B + (k + 1 - P) .* B(:, r)) / k;
      end;
end;
```

C.4 Implementação em Maple

Essa é a tradução para Maple da versão da rotina da Seção C.3. Portanto, considera apenas o caso particular em que o *grid* de nós é uniforme. Nós a utilizamos apenas para avaliação da precisão numérica das versões em Ox e em R, montando a matriz de B-*splines* com variáveis simbólicas e avaliando-a, ao final, com mantissa de 100 dígitos.

Arquivo bsmaple.mws

```
bs1 := proc(x, xl, xr, ndx, bdeg)
   local i, j, k,
         n_x, dx,
         t, ones_t, ones_x, B1, r,
         T, X, P, B:

   with(linalg): # PACOTE PARA OPERAÇÕES COM MATRIZES

   n_x := LinearAlgebra[Dimension](x): # TAMANHO DE x
   dx := (xr-xl)/ndx: # O GRID UNIFORME
   ones_t := <seq(1, i = 1..bdeg+ndx)>: # VETOR DE UNS

   t:=convert(ones_t + <dx*seq(i,i=-bdeg..ndx-1)>,matrix):
                                                  # VETOR DE UNS
   ones_x := convert(<seq(1, i = 1..n_x)>, matrix):
                                                  # VETOR DE UNS
   ones_t := convert(ones_t, matrix);

   T := multiply(ones_x, transpose(t)):
                              # O INÍCIO DE CADA INTERVALO
   X := multiply(convert(x, matrix), transpose(ones_t)):
                              # VALORES ONDE AVALIAMOS A MATRIZ
   P := evalm((X-T)/dx):      # COEFICIENTES DA RECURSÃO
```

```
   B := matrix(n_x, bdeg+ndx, 0):
   for i from 1 to n_x do
      for j from 1 to bdeg+ndx do
         if (is(T[i,j] <= X[i,j]) and is(X[i,j] < T[i,j]+dx)) then
            B[i,j] := 1: # EM QUE INTERVALO ESTAMOS AVALIANDO
         end if:
      end do:
   end do:

   B1 := <seq(B[i,1], i = 1..10)>:
        # PRIMEIRA LINHA DE B, NECESSÁRIA PQ O LAÇO ABAIXO ALTERA-A (*)

   r := <seq(i, i = 2..bdeg+ndx), 1>:
   for k from 1 to bdeg do
      for i from 1 to n_x do
         for j from 1 to bdeg+ndx-1 do
            B[i,j] := (P[i,j]*B[i,j] + (k+1-P[i,j])*B[i,r[j]])/k:
         end do:
            B[i,j] := (P[i,j]*B[i,j] + (k+1-P[i,j])*B1[i])/k: # DE (*)
      end do:
   end do:

   return(evalm(B)):
end proc:
```

Glossário

Esse glossário exibe alguns termos utilizados no texto cujo significado não foi explicitamente declarado ou enfatizado e a definição do modo como consideramos alguns deles.

$(x)_+$: $\max(x, 0)$

$<\beta, x>_\lambda$: produto interno entre β e x considerando apenas as suas λ primeiras posições $(\sum_{j=1}^{\lambda} \beta_j x_j)$

$\|.\|_1$: norma L_1

$a..b$: $a, a+1, a+2, ..., b-1, b$

absolutamente continuamente diferenciáveis, curvas: curvas cujas derivadas são absolutamente contínuas

AIC: critério de informação de Akaike (Akaike *information criterion*)

bandwidth: largura da banda ou do intervalo

bump: saliência

$C^v[a,b]$: classe das funções v vezes continuamente diferenciáveis em $[a,b]$

CV: validação cruzada (*cross-validation*)

DD: diferenças divididas

EQM: erro quadrático médio

EQM combinado: EQM considerando como referência uma curva que não a verdadeira

feature: saliência, nuance

grid: grade ou malha onde são definidos os *knots*

$\imath(P)$: função indicadora da propriedade P

knots: nós

$\mathcal{L}_2[a,b]$: classe das funções cujo quadrado é integrável em $[a,b]$

$\mathcal{N}(\mu, \sigma^2)$: distribuição normal com média μ e variância σ^2

nuance: o mesmo que propriedades; *bump*

online: medido continuamente

$\Re$: corpo dos reais

roughness: quantidade de curvatura, oscilação, em oposição a suavidade

shift: translação

sign(x): função sinal de x

SOC: sistema ortonormal completo

spline suave: *smoothing spline*

SQRP: soma de quadrados dos resíduos penalizada

$\mathcal{U}(a, b)$: distribuição uniforme com parâmetros a e b

$\mathcal{W}_2^v[a,b]$: classe das funções v-1 vezes absolutamente continuamente diferenciáveis com $D^v f \in \mathcal{L}_2[a,b]$, espaço de Sobolev de ordem v

Bibliografia

Advanced Topics in computer Graphics in http://www-courses.cs.uiuc.edu/~cs319/bsplines.pdf

Akaike, H. (1974). A new look at the statistic model identification. *IEEE Transactions and Automatic Control, AC-19(6)*, 716-723.

Anselone, P.M. and Laurent, P.J. (1968). A General Method for Construction of Interpolating or Smoothing Spline Functions. *Numerische Mathematik, 12*, 66-82.

B-spline Basis Functions: Definition in http://www.cs.mtu.edu/~shene/courses/cs3621/notes/spline/bspline-basis.html

Bates, D.M. and Venables, W.N. *SplineDesign*. Source code from http://cran.r-project.org.

Bos, Charles. Graphics package Gnudraw version 2002. Downloaded from www.tinbergen.nl/~cbos/gnudraw.html

Celeux, G. and Dielbolt, J. (1985). The SEM algorithm: a probabilistic teacher algorithm derived from the EM algorithm for the mixture problem. *Comp. Statist. Quart., 2*, 73-82.

Celeux, G. Chauveau and Diebolt, J.(1996). Stochastic version of the EM algorithm: Experimental study in the mixture case. *Journal of Statistical Computation and Simulation, 55(4)*: 287-314.

Craven, P and Wahba, G. (1979). Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of the generalized cross-validation. *Numerische Mathematik*, 31, 377-403.

De Boor, C. (1978) *A Practical Guide to Splines*. New York: Springer Verlag.

Dempster, A.P., Laird, N.M. and Rubin, D.B. (1977). Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society Series B, 39*, 1-38.

Dias, R. and Gamerman, D. (2002). A Bayesian approach to hybrid splines nonparametric regression. *Journal of Statistical Computation and Simulation, 72*, 285-297.

Doornik, J.A. (2001). *Ox: An Object-Oriented Matrix Language* (4th edition), London: Timberlake Consultants Press.

Efron, B. and Tibshirani, R.J. (1993). *An Introduction to the Bootstrap*. London, U.K.: Chapman & Hall.

Eilers, P. H. C. and Marx, B. D. (1996). Flexible Smoothing with B-splines and Penalties. *Statistical Science, 11(2)*: 89-121.

Eubank, R.L. (1988). *Spline Smoohing and Nonparametric Regression*. New York.

Fortran Source Codes in http://www.psc.edu/~burkardt/src/spline/spline.html

Green, P.J. and Silverman, B.W. (1994). *Nonparametric Regression and Generalized Linear Models: A Roughness Penalty Approach*. London: Chapman and Hall.

Gnuplot version 1998. Graphics platform Gnuplot for windows. Downloaded from www.gnuplot.info/

Marschner, I. (2001). On stochastic versions of the EM algorithm. *Biometrika, 88(1)*, 281-286.

Ramsay, J.O. (2001a). *A Guide do Curve Registration* in ftp://ego.psych.mcgill.ca/pub/ramsay/FDAfuns/Rguide.pdf

Ramsay, J.O. (2001b). *Matlab and S-plus Functions for Functional Data Analysis* in ftp://ego.psych.mcgill.ca/pub/ramsay/FDAfuns/Mguide.pdf

Ramsay, J.O. and Li, Xiaochun (1998). Curve Registration. *Journal of the Royal Statistical Society Series B, 60(2)*: 351-363.

Ramsay, J.O. and Silverman, B.W. (1997). *Functional Data Analysis*. New York: Springer.

Schoenberg, I.J. (1946). Contributions to the problem of approximation of equidistant data by analytic functions. *Quart. Appl. Math. 4*, 45-99 and 112-141.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society B, 36*, 111-147.

Stone, M. (1977). Asymptotics for and against cross-validation. *Biometrika, 64(1)*:29-35.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B, 58(1)*: 267-288.

Wegman, E.J. and Wright, I.W. (1983). Splines in Statistics. *Journal of the American Statistical Assossiation, 78(382)*: 351-365.

Xiang, D. and Wahba, G. (1996). A generalized approximate cross validation for smoothing splines with non-Gaussian data. *Statistica Sinica, 6*, 675-692.

Índice Alfabético