

Estimation of knots location and number in the splines regression models using an optimization approach

Alberto Rodrigues Ferreira

DISSERTATION PRESENTED
TO THE
INSTITUTE OF MATHEMATICS AND STATISTICS
OF
UNIVERSITY OF SÃO PAULO
TO
OBTAIN THE DEGREE
OF
MASTER IN SCIENCE

Program: Statistics

Advisor: Prof. Dr. Florencia Graciela Leonardi

Co-advisor: Prof. Dr. Alex Rodrigo dos Santos Sousa

During the development of this work the author received financial support from CNPq

São Paulo, February, 2022

Estimation of knots location and number in the splines regression models using an optimization approach

This is the original version of the dissertation of the
Msc. candidate Alberto Rodrigues Ferreira
as presented to the Judging Committee.

Estimation of knots location and number in the splines regression models using an optimization approach

This version of the dissertation contains the changes suggested by the Judging Committee during the defence of the original version of the work, made on 04/15/2022. A copy of the original version is available at Institute of Mathematics and Statistics of the University of São Paulo.

Judging Committee:

- Prof. Dr. Florencia Graciela Leonardi - Universidade de São Paulo
- Prof. Dr. Ronaldo Dias - Universidade Estadual de Campinas
- Prof. Dr. Mariela Sued - Universidad de Buenos Aires

Acknowledgments

I would like to thank my advisor Prof. Dr. Florencia Leonardi, for providing guidance, support, and feedback throughout this important master's project. Without this help, the dissertation would be much more difficult. To my co-advisor, Prof. Dr. Alex Sousa, who helped me in essential parts of the project and was always willing to collaborate.

For the members of the judging committee, Prof. Ronaldo, Prof. Mariela and Prof. Florencia, thank you for your availability and your future suggestions in our work, I am absolutely sure that they will be very important to improve our proposed method more and more.

I would also like to thank my friends Alex, Gabi and Pedro from my master's degree. You definitely made the journey easier during the difficult courses we had together. To my friends Daniele, Diego, Maraca, Daniel, Matheus, Suellem, Yohana, Anderson, Iury, Natália, Rebecca, Rafael, Thainá, and Deni, thanks for the help, funny conversations, advice, and academic partnerships.

To my professors at USP, where I learned a lot about statistics, you gave me essential academic advice and made my master's degree an incredible experience, thank you very much, especially to professors Alexandre Patriota, Florencia Leonardi, Silvia Nagib, Luis Gustavo, Victor Fossaluza and Carlos Hitoshi .

To my professors at UFC, where I obtained the basic knowledge of statistics and started to learn machine learning, an area that I want to work all my life, thank you very much, especially to Professors Juvêncio Santos, Luis Gustavo, Mauricio Mota, Gualberto Montalvo, Silvia, Ronald, Tiberius, and Leandro.

Abstract

FERREIRA, A. R., LEONARDI, F. G., SOUSA, A. R. S **Estimation of knots location and number in the splines regression models using an optimization approach**. 2022. Dissertation (Master's degree) - Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2022.

In many practical problems related to supervised statistical learning, we are interested in predicting a continuous target. Frequently, the relationship between the explanatory variable and the target variable is nonlinear, so models that introduce nonlinearity for this purpose tend to obtain better performances in general. A statistical model that addresses this problem called the regression splines model has received considerable attention in recent years. This is due to its great predictive power and good fits incorporated by its flexibility. However, the splines regression model has a significant disadvantage: one of its main components, called knots, related to the change points, are usually chosen before the estimation process. They are considered pre-specified values, which in some situations can present severe problems in practical problems. In this work, we propose a new methodology that tries to solve this considering the knots location and knots number as parameters, and we solve this problem as an optimization approach using the nonlinear optimization algorithm BFGS. Furthermore, we introduce new regularization methods to penalize variables with irrelevant knots and avoid overfitting. The proposed methodology obtained many advantages compared to the approach used in the literature, such as automatic estimation of the number and location of knots, regularization methods that avoids overfitting, and selection of irrelevant knots. Our approach obtained several gains in predictive performance and knots estimation in the simulations, thus obtaining better results than the usual procedure.

Keywords: splines regression model, knots location estimation, knots number estimation, regularization methods, BFGS.

Contents

List of Figures	vii
List of Tables	ix
1 Introduction	1
2 Models Based on Knots	3
2.1 Fitting additive models	3
2.2 Piecewise Polynomials	5
2.3 Regression Splines	6
2.4 Disadvantages of Regression Splines	9
2.4.1 Manual choose	10
2.4.2 Uniform choose	11
2.5 Disadvantages of Linear Splines	12
2.5.1 Manual choose	13
2.5.2 Uniform choose	15
2.6 Disadvantages on Cubic Splines	17
2.6.1 Manual choose	19
2.6.2 Uniform choose	20
3 Estimation of the Number and Location of Knots	23
3.1 Cost functions	23
3.2 Motivation	26
3.2.1 Newton's Method vs Gradient Descent: Computational Comparison	27
3.3 Quasi Newton Methods	28
3.4 The BFGS algorithm	33
3.5 Estimation of Number of Knots	41
3.6 Summary of the parameter estimation procedure	48
3.7 Loss functions comparasion	48
3.7.1 Knot number estimation and hyperparameter chosen	49
3.7.2 Predictive performance	56
4 Predictive Performance with p variables	59
4.1 Simulations	61
4.1.1 Predictive results with $\sigma^2 = 1$	62
4.1.2 Predictive results with $\sigma^2 = 2$	63

4.1.3	Predictive results with $\sigma^2 = 3$	63
4.2	Performance on Real Data	63
4.2.1	Airfoil Self-Noise Data Set	64
4.2.2	Auto MPG Data Set	64
4.2.3	Concrete Compressive Strength Data Set	64
4.2.4	Combined Cycle Power Plant Data Set	65
4.2.5	QSAR aquatic toxicity Data Set	65
4.2.6	Real estate valuation data set Data Set	66
4.2.7	Predictive Results	66
5	Conclusion	69
A	Code	71
	Bibliography	73

List of Figures

2.1	Simulated nonlinear fit	5
2.2	Comparison between piecewise and polynomial regression	6
2.3	Regression splines fitted with two knots: $t_{11} = 18$ and $t_{12} = 85$	7
2.4	Proper fit with fewer parameters	11
2.5	Incompatible fit using uniform knots choice	12
2.6	Linear Splines Simulated	13
2.7	Linear Splines with Manual Choice First Situation	14
2.8	Linear Splines with Manual Choice Second Situation	14
2.9	Linear Splines with Manual Choice Third Situation	15
2.10	Linear Splines with Uniform Choice First Situation	16
2.11	Linear Splines with Uniform Choice Second Situation	16
2.12	Linear Splines with Uniform Choice Third Situation	17
2.13	Cubic Splines Simulated	18
2.14	Cubic Splines with Manual Choice First Situation	19
2.15	Cubic Splines with Manual Choice Second Situation	19
2.16	Cubic Splines with Manual Choice Third Situation	20
2.17	Cubic Splines with Uniform Choice First Situation	21
2.18	Cubic Splines with Uniform Choice Second Situation	21
2.19	Cubic Splines with Uniform Choice Third Situation	22
3.1	Estimated curves using $J_1(\theta)$ with different λ values	24
3.2	Estimated curves using $J_2(\theta)$ with different λ values	25
3.3	Level curves comparison between gradient descent and Newton's method	27
3.4	θ estimate over the iterations	32
3.5	$\sin(\theta)$ over the iterations	32
3.6	Generated curve example	39
3.7	Estimated curves at four iterations	40
3.8	Knot convergence over the iterations	40
3.9	Cost function over the iterations	41
3.10	Generated curve with one knot and $K=1$	42
3.11	Generated curve with two knots and $K=1$	43
3.12	Generated curve with three knots and $K=1$	44
3.13	Generated curve with one knot and $K=3$	45
3.14	Generated curve with two knots and $K=3$	46

3.15	Generated curve with three knots and $K=3$	47
3.16	$\hat{\alpha}$ proportion comparison with one knot and $K = 1$	49
3.17	$\hat{\alpha}$ proportion comparison with two knots and $K = 1$	51
3.18	$\hat{\alpha}$ proportion comparison with three knots and $K = 1$	52
3.19	$\hat{\alpha}$ proportion comparison with one knot and $K = 3$	53
3.20	$\hat{\alpha}$ proportion comparison with two knots and $K = 3$	54
3.21	$\hat{\alpha}$ proportion comparison with three knots and $K = 3$	55
4.1	x_1 form	61
4.2	x_2 form	61
4.3	x_3 form	62

List of Tables

3.1	Average cost functions for $J_1(\theta)$	42
3.2	Average cost functions for $J_2(\theta)$	42
3.3	Average cost functions for $J_1(\theta)$	43
3.4	Average cost functions for $J_2(\theta)$	43
3.5	Average cost functions for $J_1(\theta)$	44
3.6	Average cost functions for $J_2(\theta)$	44
3.7	Average cost functions for $J_1(\theta)$	45
3.8	Average cost functions for $J_2(\theta)$	45
3.9	Average cost functions for $J_1(\theta)$	46
3.10	Average cost functions for $J_2(\theta)$	46
3.11	Average cost functions for $J_1(\theta)$	47
3.12	Average cost functions for $J_2(\theta)$	47
3.13	λ proportion chosen for $J_1(\theta)$	50
3.14	λ proportion chosen for $J_2(\theta)$	50
3.15	λ proportion chosen for $J_1(\theta)$	51
3.16	λ proportion chosen for $J_2(\theta)$	51
3.17	λ proportion chosen for $J_1(\theta)$	52
3.18	λ proportion chosen for $J_2(\theta)$	52
3.19	λ proportion chosen for $J_1(\theta)$	53
3.20	λ proportion chosen for $J_2(\theta)$	53
3.21	λ proportion chosen for $J_1(\theta)$	54
3.22	λ proportion chosen for $J_2(\theta)$	54
3.23	λ proportion chosen for $J_1(\theta)$	55
3.24	λ proportion chosen for $J_2(\theta)$	55
3.25	Predictive performance comparison between cost functions	56
3.26	Predictive performance comparison between cost functions	56
3.27	Predictive performance comparison between cost functions	56
3.28	Predictive performance comparison between cost functions	56
3.29	Predictive performance comparison between cost functions	56
3.30	Predictive performance comparison between cost functions	57
4.1	Simulations predictive results with $\sigma^2 = 1$	62
4.2	Simulations predictive results with $\sigma^2 = 2$	63
4.3	Simulations predictive results with $\sigma^2 = 3$	63

4.4	Details airfoil dataset	64
4.5	Details auto mpg dataset	64
4.6	Details concrete dataset	65
4.7	Details cycle power dataset	65
4.8	Details qsar dataset	66
4.9	Details valuation dataset	66
4.10	Real data predictive results	66
4.11	Predictice performance comparison	67

Chapter 1

Introduction

In many practical problems related to supervised statistical learning, we are interested in predicting a continuous target. Often the relationship between the explanatory variable and the target variable is non-linear. Hence, models that introduce nonlinearity for this purpose tend to obtain better performances in general. A statistical model that addresses this type of problem called the regression splines model has received considerable attention in recent years. It is a non-parametric model and tries to balance the interpretability and flexibility provided by additive models[10], which are a generalization of linear models that allow non-linear relationships between the explanatory variable and the response variable. This is due to its great predictive power and good fits due to its flexibility.

Also, regression splines is an extension of polynomial regression models. It involves splitting the range of each explanatory variable into a few different regions. Within each region, a polynomial function is fitted to the data [14]. However, we have to worry about overfitting because if the model is too flexible, the estimates can have big variance and not predict well observations of a set that it has not yet seen. Because of this, regularization methods are of fundamental importance to have a trade-off between bias and model variance.

The splines regression model has a significant disadvantage: one of its main components, called knots, related to the change points are usually chosen before the estimation process. They are considered pre-specified values, which in some situations can present severe problems in practical problems. In this work, we propose a new methodology that tries to solve this considering the knots location and knots number as parameters, and we solve this problem as an optimization approach using the non-linear optimization algorithm BFGS. Furthermore, we introduce new regularization methods to penalize variables with irrelevant knots and avoid overfitting.

The proposed methodology obtained many advantages compared to the approach used in the literature, such as automatic location knot estimation, automatic number knot estimation, regularization method that avoids overfitting or underfitting, and selection of irrelevant knots. In the simulations performed, our approach obtained several gains in predictive performance and knots estimation, thus obtaining good results compared to the approach used in the usual literature and many well-known predictive models in the literature.

This dissertation has the following structure: chapter 1 introduces some additive models, including the spline regression model, which will be the focus of the dissertation. It also presents the main methods for choosing knots and its disadvantages. Chapter 3 presents the proposed cost functions, our main contributions to the splines regression model, the motivation behind the estimation method used, and the knots number estimation method. Chapter 4 tests our proposed methods in simulations and several real data sets how the proposed methods behave against several machine learning algorithms known in the literature.

Chapter 2

Models Based on Knots

Statistical models are often a powerful tool for various purposes, such as making predictions, obtaining estimates, checking which variables are significant to the problem, and understanding the relationship between the explanatory and response variables.

Frequently, the relationship between the explanatory variable and the response variable is non-linear, and thus some traditional models, despite having the advantages of simplicity, may fail to obtain a proper fit. Therefore, a flexible alternative for this objective is to use additive models [10], which is a class of models defined by:

$$Y_i = \beta_0 + \sum_{j=1}^p f_j(x_{ij}) + \epsilon_i, \forall i = 1, \dots, n$$

where the errors have $E[\epsilon_i] = 0, \forall i = 1, \dots, n$.

Additive models are non-parametric models, since the functions $f_j, j = 1, \dots, p$ are not specified. In practice, these functions are smooth for each variable, and the parameters can be estimated by the backfitting algorithm or simply by the least squares method, depending on the functions f_j used in the functional form of the additive model.

When we fit a linear regression model, we usually do not believe that the model is correct. Instead, we believe that it will be a simple approximation to the true model and that we can uncover the important predictors and their roles using this approximation [10]. Additive models are more general approximations where we can obtain relations more realistic in most of the datasets.

2.1 Fitting additive models

There are several ways to approach the formulation and estimation of additive models. Normally, these methods change through the constraints and the smoothness of the functions in the model. A few approaches that handle the estimation can be as follows:

1. The simplest possible model using the additive models approach is the multiple linear regression model with several possible tools such as confidence intervals, prediction intervals, hypothesis tests, and estimates of the variances of well-defined estimators, which can be used when it is a reasonable model. It is a simple but powerful modeling tool as a first attempt at modeling data and perform predictions.
2. More general versions than multiple linear regression can be incorporated. These approaches can add flexibility and predictive power and can be classified as parametric non-linear additive models. Some examples are the addition of non-linear explanatory variable transformations, such as logarithmic function, square root function, and mainly polynomials, which are the

most used.

Another possibility would be an orthogonal polynomial approach, which creates a set of polynomials as a function of the explanatory variables to avoid approximate multicollinearity.

3. Also, there are models called generic splines that use a particular set of functions. Each specific model uses a different type of base. For example, the splines regression fits a polynomial regression in some regions of the explanatory variable range. This model uses an important component called knots. In the literature, knots are pre-chosen, which can be a significant disadvantage. This dissertation will explore this model more in the next sections. The prevalent choice is B-Splines[4].
4. The most general algorithm for estimating additive models allows us to estimate each function separately by any smoother. Some examples of these models are cubic smoothing splines, locally weighted and kernel smoothers. The algorithm that estimates these models parameters is called backfitting, which uses conditional expectations and an iterative process.

The main idea is to estimate each function by fitting the new target like being the residuals. For example, fitting the target $Y_i - \beta_0 - \sum_{\substack{j=1 \\ j \neq k}}^p f_j(x_{ij})$ as a function of the variable x_{ij} to estimate $f_k, \forall k = 1, \dots, p$

The algorithm called backfitting has an objective to estimate the parameters of each function $f_j, \forall j = 1, \dots, p$. However, the biggest problem in this algorithm is identifiability, which occurs because we can add or subtract a constant from functions f and still obtain the same value of the functional form. Thus, a way to avoid this problem is to fix the estimate of $\beta_0 = \bar{Y}$ and add the following constraint $\sum_{i=1}^n f_j(x_{ij}) = 0$. More details about the mathematical tricks of the backfitting and estimations of specific models can be found in [7],[13],[10].

The pseudocode of the backfitting algorithm is given below:

Algorithm 1 Backfitting algorithm

Input: Training data X, y .

Initialize: $\beta_0 = \bar{y}$ and generate initial guesses for $\hat{f}_j(X), \forall j = 1, \dots, p$.

while \hat{f}_j does not converge $\forall k = 1, \dots, p$ **do**

for $k = 1, \dots, p$ **do**

 1. Compute $\tilde{Y}_i = Y_i - \hat{\beta}_0 - \sum_{\substack{j=1 \\ j \neq k}}^p \hat{f}_j(x_{ij}), \forall i = 1, \dots, n$

 2. Estimate f_k with the target \tilde{Y} and the explanatory variable X_j .

 3. Uptade $\hat{f}_k(X) = \hat{f}_k(X) - \frac{1}{n} \sum_{i=1}^n \hat{f}_k(x_{ij})$

end for

end while

2.2 Piecewise Polynomials

Over the years, the amount of data has only grown significantly. Statistical models must follow this evolution to obtain a good fit and relevant results, such as predictions, model selection, interpretability, confidence intervals, and hypothesis tests.

One possible model that is a particular case of the non-parametric additive models is piecewise polynomial models. Instead of fitting a polynomial model to the entire dataset, a polynomial model is fitted to subsets of data to be more flexible and predict better future observations.

We start simply with a polynomial regression, for example, with a cubic degree.

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \epsilon_i \quad (2.1)$$

However, the coefficients $\beta_0, \beta_1, \beta_2, \beta_3$ change by some specified value of x . More specifically:

$$y_i = \begin{cases} \beta_{01} + \beta_{11}x_i + \beta_{21}x_i^2 + \beta_{31}x_i^3 + \epsilon_i & \text{if } x_i \leq c \\ \beta_{02} + \beta_{12}x_i + \beta_{22}x_i^2 + \beta_{32}x_i^3 + \epsilon_i & \text{if } x_i > c \end{cases} \quad (2.2)$$

In other words, we fit two different polynomial functions to the data, one in the subset of observations with $x_i \leq c$, and one in the subset of the observations with $x_i > c$. The first polynomial function has coefficients $\beta_{01}, \beta_{11}, \beta_{21}, \beta_{31}$, and the second has the coefficients $\beta_{02}, \beta_{12}, \beta_{22}, \beta_{32}$ [14]. These polynomial models can be fitted simply using the least squares estimator, thus obtaining estimates for each subset of the data. The change points are called knots. In the previous example, they are equal to c and are an important component in this type of model and the models studied in this dissertation. From now on, all models presented that have these change points will be referenced in this way.

An important observation is that using more knots can lead to very flexible models since we will fit polynomial statistical models by regions, making the fit as flexible as possible. In practice, the knots number is pre-specified and may vary depending on the data set. This approach in additive models will be explained in the following sections of this dissertation.

For instance, 200 observations of the curve in the figure below were generated, and then we compared how a polynomial regression and a piecewise polynomial behave for non-linear relationships.

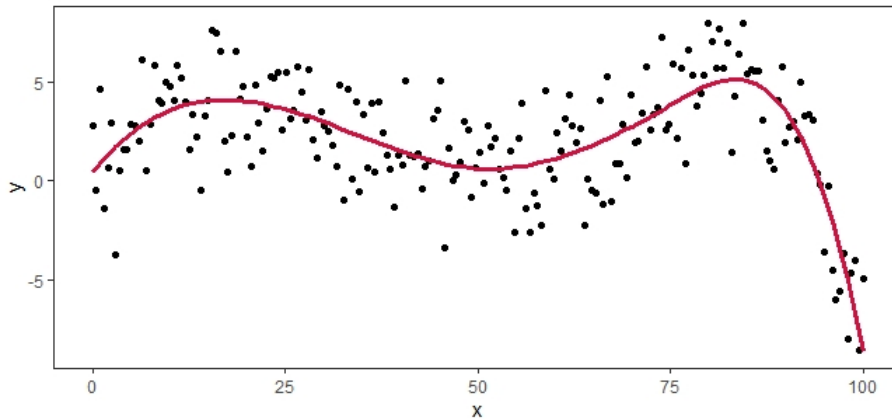


Figure 2.1: *Simulated nonlinear fit*

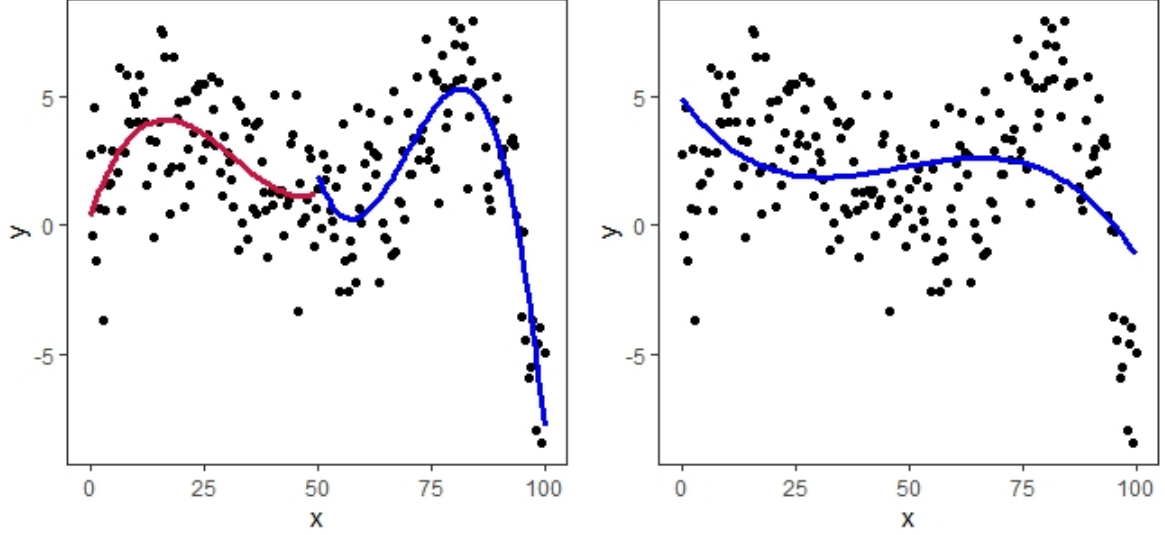


Figure 2.2: *Comparison between piecewise and polynomial regression*

The Figure 2.2 shows the piecewise model(left side) with $c=50$ and polynomial regression(right side) fitted the simulated data. We can see that the piecewise polynomial model obtains a better fit than a single polynomial model, which is expected in non-linear situations since, as previously mentioned, this model is more flexible and can capture more non-linear fit possibilities.

The biggest problem with this model is that the piecewise model fitted is not continuous and has many more parameters than necessary, which makes this model not attractive.

One way to solve this problem is to use the same idea of fitting polynomials in regions of the range of the explanatory variables, but now with the constraint that the fit is continuous. This is the primary motivation for constructing the Regression Splines statistical model, which will be widely studied in this dissertation.

2.3 Regression Splines

The Splines Regression model can be considered the evolution of the piecewise polynomial models presented in the previous section. This is due to the reason of having the same idea of the piecewise models, which is fitting polynomials in some regions of the interval for each numerical explanatory variable. However, now with the condition of being a continuous fit and because of this, it is likely to estimate fewer parameters and has less computational cost.

Also, just as piecewise models have pre-specified knots, spline regression models make use of this unique component, but now with a big difference, these knots are explicitly used in the functional form of the splines regression. This way, continuity is guaranteed through functions that involve these change points called truncated power basis. The splines regression model is defined as follows:

$$Y_i = \beta_0 + \sum_{j=1}^p f_j(x_{ij}) + \epsilon_i, \forall i = 1, 2, \dots, n \quad (2.3)$$

where the $f_j(x_{ij})$ are given bellow

$$f_j(x_{ij}) = \sum_{k=1}^{K_j} \beta_{(j,k)} x_{ij}^k + \sum_{m=1}^{\alpha_j} \beta_{(j,m+K_j)} (x_{ij} - t_{jm})^{K_j} I(x_{ij} > t_{jm}), \quad (2.4)$$

where p is the total number of explanatory variables, K_j is the polynomial degree of each variable and α_j is the knots number of each explanatory variable with index j and ϵ_i is the source of variation typically associated with the error that cannot be controlled, with $E[\epsilon_i] = 0, \forall i = 1, \dots, n$ and I is the indicator function.

The parameters will be defined by the vector $\theta = [\beta_0, \beta_{(1,1)}, \dots, \beta_{(p, \alpha_p + K)}] \in R^{\sum_{j=1}^p [\alpha_j + K_j] + 1}$. The knots are pre-specified and denoted by t_{jm} for j -th explanatory variable and m -th knot in this specific variable. For example, if we consider a single numerical explanatory variable, $K=3$ and two knots, the model given in (2.3) can be summarized by:

$$Y_i = \beta_0 + f_1(x_{i1}) + \epsilon_i \quad (2.5)$$

where

$$f_1(x_{i1}) = \beta_{(1,1)}x_{i1} + \beta_{(1,2)}x_{i1}^2 + \beta_{(1,3)}x_{i1}^3 + \beta_{(1,4)}(x_{i1} - t_{11})^3 I(x_{i1} > t_{11}) + \beta_{(1,5)}(x_{i1} - t_{12})^3 I(x_{i1} > t_{12}) \quad (2.6)$$

A possible example of a fit with two knots is the figure below.

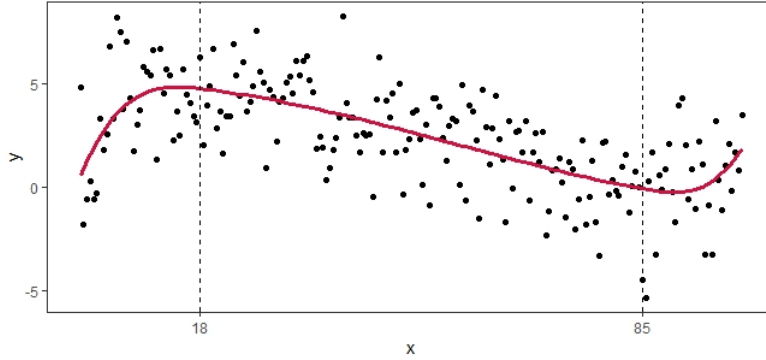


Figure 2.3: Regression splines fitted with two knots: $t_{11} = 18$ and $t_{12} = 85$

The knots are specified before the parameter estimation, in this simulation case, the values of knots are respectively $t_{11} = 10$ and $t_{12} = 85$. With this type of approach, we can see a fit that makes sense, and it is pretty reasonable since only the change points in the estimated curve are close to the previously established knots.

An important component in the splines regression model is the second sum of the equation given in (2.4). We can realize that in each function f_j , there is a polynomial regression with the truncated power basis. Therefore, it is essential to verify what happens with the addition of these functions, the estimates of their respective parameters and interpretations that we can provide to clearly explain the likely increase in flexibility and predictive power.

In practice, the splines regression model fits polynomials in pre-established regions through the knots. The best way to realize this is through parameter estimates. Returning to the example given in (2.5), with the fit made we will have the following estimates $\hat{\beta}_0, \hat{\beta}_{(1,1)}, \hat{\beta}_{(1,2)}, \hat{\beta}_{(1,3)}, \hat{\beta}_{(1,4)}, \hat{\beta}_{(1,5)}$.

In this case, we have 2 knots, so we will have three regions. The first region is given by the observations that are smaller than the first knot(t_{11}). The second region is given by the observations that are between the first(t_{11}) and the second knot(t_{12}) and the third given by the observations that are larger than the second knot(t_{12}). In this way, we can write the estimates for each region

to understand better how the fit is obtained. So we have:

$$\hat{y}_i = \begin{cases} \hat{\beta}_0 + \hat{\beta}_{(1,1)}x_i + \hat{\beta}_{(1,2)}x_i^2 + \hat{\beta}_{(1,3)}x_i^3 & \text{if } x_i \leq 12 \\ \hat{\beta}_0 + \hat{\beta}_{(1,1)}x_i + \hat{\beta}_{(1,2)}x_i^2 + \hat{\beta}_{(1,3)}x_i^3 + \hat{\beta}_{(1,4)}(x_i - 12)^3 & \text{if } 12 < x_i \leq 88 \\ \hat{\beta}_0 + \hat{\beta}_{(1,1)}x_i + \hat{\beta}_{(1,2)}x_i^2 + \hat{\beta}_{(1,3)}x_i^3 + \hat{\beta}_{(1,4)}(x_i - 12)^3 + \hat{\beta}_{(1,5)}(x_i - 88)^3 & \text{if } x_i > 88 \end{cases} \quad (2.7)$$

Through the expression (2.7) we can see that we have different prediction functions for each of the three regions. In other words, the splines regression model estimates its parameters based on observations in different regions in order to increase flexibility and predictive power. Furthermore, we can see that the higher the values of the explanatory variable, an estimate is added to predict the observations.

It may be more natural for some people to use the piecewise model rather than the spline regression model. However, the piecewise model does not guarantee continuity. It has many more parameters than the spline regression model because it fits a polynomial by regions separately and does not have the advantages of the spline regression model discussed in the following sections.

In practice, we have more than one explanatory variable and consequently more parameters to estimate in the splines regression model. The number of prediction equations grows with the increase in the knots number of the variables and, consequently, the flexibility of the model and its predictions.

Let us consider two explanatory variables, the first with only one knot and the second with three knots. In this way, the functional form of the model would be given by:

$$Y_i = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \epsilon_i \quad (2.8)$$

where

$$f_1(x_{i1}) = \beta_{(1,1)}x_{i1} + \beta_{(1,2)}(x_{i1} - t_{11})I(x_{i1} > t_{11}) \quad (2.9)$$

and

$$f_2(x_{i2}) = \beta_{(2,1)}x_{i2} + \beta_{(2,2)}(x_{i2} - t_{21})I(x_{i2} > t_{21}) + \beta_{(2,3)}(x_{i2} - t_{22})I(x_{i2} > t_{22}) \quad (2.10)$$

$$+ \beta_{(2,4)}(x_{i2} - t_{23})I(x_{i2} > t_{23}) \quad (2.11)$$

As in the previous example, knots are pre-specified before the estimation process, in this case $t_{11} = 88$ while $t_{21} = 19$, $t_{22} = 44$ and $t_{23} = 78$. Now, since we have eight different regions in our set, we will have eight prediction equations based on the knots number of the two variables. The number of regions for the generic case of p variables is calculated by $\prod_{j=1}^p (\alpha_j + 1)$.

The prediction equations provided for each region are given below:

$$\hat{y}_i = \begin{cases} \hat{\beta}_0 + \hat{\beta}_{(1,1)}x_{i1} + \hat{\beta}_{(2,1)}x_{i2} & \text{if } x_{i1} \leq 88 \text{ and } x_{i2} \leq 19 \\ \hat{\beta}_0 + \hat{\beta}_{(1,1)}x_{i1} + \hat{\beta}_{(2,1)}x_{i2} + \hat{\beta}_{(1,2)}(x_{i1} - 88) & \text{if } x_{i1} > 88 \text{ and } x_{i2} \leq 19 \\ \hat{\beta}_0 + \hat{\beta}_{(1,1)}x_{i1} + \hat{\beta}_{(2,1)}x_{i2} + \hat{\beta}_{(2,2)}(x_{i2} - 19) & \text{if } x_{i1} \leq 88 \text{ and } 19 < x_{i2} \leq 44 \\ \hat{\beta}_0 + \hat{\beta}_{(1,1)}x_{i1} + \hat{\beta}_{(2,1)}x_{i2} + \hat{\beta}_{(1,2)}(x_{i1} - 88) + \hat{\beta}_{(2,2)}(x_{i2} - 19) & \text{if } x_{i1} > 88 \text{ and } 19 < x_{i2} \leq 44 \\ \hat{\beta}_0 + \hat{\beta}_{(1,1)}x_{i1} + \hat{\beta}_{(2,1)}x_{i2} + \hat{\beta}_{(2,2)}(x_{i2} - 19) + \hat{\beta}_{(2,3)}(x_{i2} - 44) & \text{if } x_{i1} \leq 88 \text{ and } 44 < x_{i2} \leq 78 \\ \hat{\beta}_0 + \hat{\beta}_{(1,1)}x_{i1} + \hat{\beta}_{(2,1)}x_{i2} + \hat{\beta}_{(1,2)}(x_{i1} - 88) + \hat{\beta}_{(2,2)}(x_{i2} - 19) & \text{if } x_{i1} > 88 \text{ and } 44 < x_{i2} \leq 78 \\ \quad + \hat{\beta}_{(2,3)}(x_{i2} - 44) & \text{if } x_{i1} > 88 \text{ and } 44 < x_{i2} \leq 78 \\ \hat{\beta}_0 + \hat{\beta}_{(1,1)}x_{i1} + \hat{\beta}_{(2,1)}x_{i2} + \hat{\beta}_{(2,2)}(x_{i2} - 19) + \hat{\beta}_{(2,3)}(x_{i2} - 44) & \text{if } x_{i1} \leq 88 \text{ and } x_{i2} > 78 \\ \quad + \hat{\beta}_{(2,4)}(x_{i2} - 78) & \text{if } x_{i1} \leq 88 \text{ and } x_{i2} > 78 \\ \hat{\beta}_0 + \hat{\beta}_{(1,1)}x_{i1} + \hat{\beta}_{(2,1)}x_{i2} + \hat{\beta}_{(1,2)}(x_{i1} - 88) + \hat{\beta}_{(2,2)}(x_{i2} - 19) & \text{if } x_{i1} > 88 \text{ and } x_{i2} > 78 \\ \quad + \hat{\beta}_{(2,3)}(x_{i2} - 44) + \hat{\beta}_{(2,4)}(x_{i2} - 78) & \text{if } x_{i1} > 88 \text{ and } x_{i2} > 78 \end{cases} \quad (2.12)$$

The prediction equations for each region possibly provide more flexible predictions that contemplate more possibilities of the non-linear relationship between the variables. Thus, eight prediction equations for each region may seem a bit exaggerated. However, in some situations, it becomes necessary due to the complexity of the data set involved, and it is crucial to have some model selection criteria for this case or it can lead to the problem of overfitting.

One way to solve this issue is through regularization methods. Among the most commonly used are those used in ridge regression[12] through a L2 regularization, lasso regression[20] through a L1 regularization and the elastic net[21] that combines the two regularizations with some extra advantages over the lasso and ridge.

The least-squares estimation method uses no regularization and can lead to overfitting. Therefore, using some regularizations is vital for this type of estimation to avoid overfitting, where we have many parameters to be estimated. One of the goals of this dissertation is to propose flexible regularization alternatives for this case. In addition to other advantages, more details will be explained in detail in the following sections.

2.4 Disadvantages of Regression Splines

Despite often improving the quality of the fit when there are non-linear relationships between the explanatory variable and the response variable, the splines regression model can present some problems in data sets in practice. It is essential to verify in which situations we can use this statistical model without having negative impacts or minimizing them as much as possible.

Often, statistical models that are more flexible than traditional models are likely to pay the price for nonlinearity, which is expected because no model is perfect and can be used in 100% of situations. The problem that can occur most in practice is overfitting, which is the model capacity to fit beyond what is necessary to the data set and not understand the behavior of these same data. This is quite common in models with many parameters and when there is no regularization on the estimation process for this type of situation. Furthermore, with the increase in the number of parameters, another prevalent problem in this situation is the increase in the variance of the estimates and, consequently, wrong predictions, as it leads to the prediction intervals becoming much wider than they should be.

In the splines regression model, the knots number chosen for each variable is of fundamental im-

portance, as the knots number increases the number of parameters, which can lead to a wrong specification of the model and consequent overfitting. Furthermore, if the knots number chosen is less than necessary or in deficient regions, underfitting can occur, which is the model capacity of not capturing the behavior of the data and not generate good fits or predictions.[19] Thus, to solve these problems referred to as overfitting or underfitting, the model selection area is beneficial to select an ideal knots number in practice.

Finally, the main disadvantage of the spline regression model is often related to the fact that the knots are pre-specified before the parameter estimation. They are not considered parameters, but they are considered "easy" values to be obtained in the usual literature, which is not always true. A wrong choice of the location of the knots, together with the number of the knots, can generate fits that do not represent reality and lead to problems in practice.

This dissertation's main contributions and motivations, which will be explained in detail in the following sections, try to solve some problems that will be discussed to make the studied models even better and more robust to real dataset situations. Therefore, we can summarize the disadvantages of the splines regression model, and then we will explore each of them in examples in situations where the model is negatively affected and the possible solutions proposed by this dissertation.

1. Manual choose of the knots number
2. Manual choose of the knots location
3. Increase in estimated standard errors
4. Overfitting/Underfitting possibilities

There are two most common practices in choosing the number and knots location. The first is simply manually choosing these values and checking which situations are better. The second, popularly used in statistical software such as the R language, is to use the knots in uniform regions of each variable and the choice of knots number is not well defined.

2.4.1 Manual choose

A first idea to specify the knots location would be through an expert manually checking each scatter plot between the explanatory and response variables. A possible choice would be a value close to the change points of the behavior present in the graph. However, human choices have limitations, even if made by specialists, and differences between decimal digits can make a big difference in the quality of the fit model.

For example, if the possible range of an explanatory variable is between 0 and 1, and there is only one change point. Decimal places limit the choice made by the human, and there is no way to have reasonable confidence in this choice, even if made by someone who perfectly understands the nature of the sample data.

In addition, an idea for manual choice of the knots number would be to check how many change points there are in the scatter plot. However, generally in the splines regression model, as we estimate a polynomial fit in regions with the continuity constraint, there is a good possibility of needing fewer knots to estimate the dataset correctly. For example, if a graph perfectly shows six change points, it may only need to take three knots to fit the model with fewer parameters properly and possibly have better interpretations and predictions.

The following example shows this perfectly. The model that generated the data has four knots, but an estimated model with two knots fits the data correctly, at least visually.

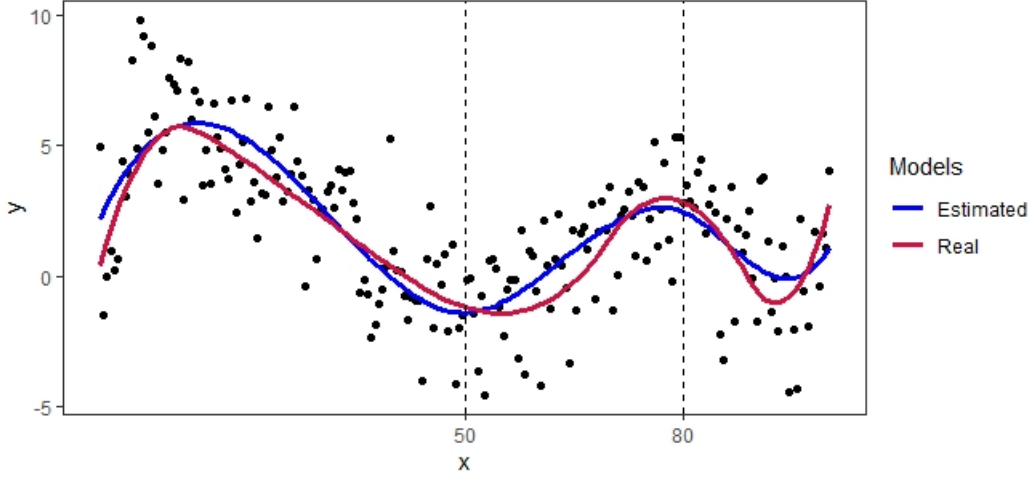


Figure 2.4: *Proper fit with fewer parameters*

More specifically, the real model has the following functional form:

$$Y_i = 0.41 + 0.98x_{i1} - 0.05x_{i1}^2 + 0.0002x_{i1}^3 + 0.03(x_{i1} - 12.83)^3 - 0.01(x_{i1} - 28.68)^3 \quad (2.13)$$

$$-0.05(x_{i1} - 69.86)^3 + 0.09(x_{i1} - 88.70)^3 + \epsilon_i \quad (2.14)$$

whereas the estimated model has the following functional form:

$$\hat{y}_i = 2.18 + 0.60x_{i1} - 0.03x_{i1}^2 + 0.0003x_{i1}^3 - 0.0007(x_{i1} - 50)^3 + 0.002(x_{i1} - 80)^3 \quad (2.15)$$

If this happens with more variables, more unnecessary parameters will be added to the estimated model and consequently can lead to overfitting. A proper procedure would be to avoid this type of problem, thus discarding parameters that do not contribute to the adjustment.

Furthermore, the manual choice would have to be performed for each numerical variable, and nowadays, there are many datasets in practice that have many variables. Therefore, in some situations, this procedure manually is impractical.

2.4.2 Uniform choose

The second procedure used to estimate the knots location is performed through a uniform choice of the range of possible values of the explanatory variable and a manual choice of the knots number. This method suffers from the same problems as the first method given in Section 2.4.1. It considers the proper choice of knots, which may not always be easy in practice.

Furthermore, the uniform choice of knots can lead to some problems, such as a wrong model specification. It is not difficult to imagine situations in which we have knots in regions that are not uniform. The simplest example would be if all change points were located at the beginning or end of the data. The next example illustrates this idea very well:

More specifically, the real model has the following functional form:

$$Y_i = 0.73 + 0.03x_{i1} - 0.9(x_{i1} - 92.06) + \epsilon_i \quad (2.16)$$

while the estimated model has the following functional form:

$$\hat{y}_i = 1.09 + 0.03x_{i1} - 0.05(x_{i1} - 50) \quad (2.17)$$

In Figure 2.5 there is only a single change point that is very near the end. In this case, the uniform

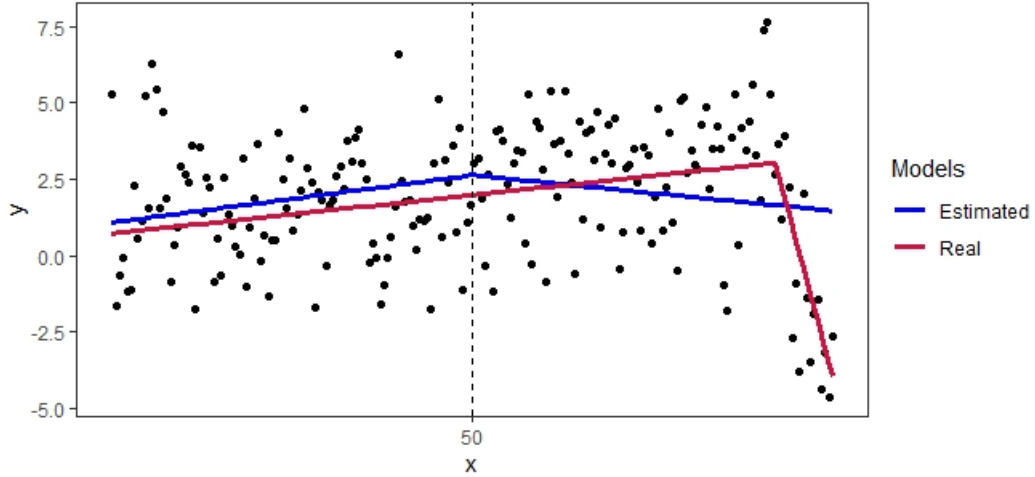


Figure 2.5: *Incompatible fit using uniform knots choice*

choice of knot location, even if the knots number is appropriately chosen, will probably suffer from some adjustment problem, as it should lead to an inadequate model specification.

This procedure is even more problematic to choose the number of knots when we have many explanatory variables. A possible commonly used strategy is choosing some possible values and testing which one best fits the problem. In [14], the author says that "One option is to try out different numbers of knots and see which produces the best looking curve", using cross-validation. This approach can lead to a simplistic or unrealistic fit and occasionally lead to overfitting or underfitting problems.

Even if the fit is reasonable, either by obtaining good predictions or having a low loss function, intuitively, there are ways to improve the fit and consequently improve the predictions considerably. Therefore, we will see some examples using these approaches, and then we will make some considerations and possible improvements that can be made.

Unlike the previous method, choosing knots uniformly can provide better results when there is not much information about the change points or difficult choices. On the other hand, this procedure depends on a reasonable choice of the knots number.

2.5 Disadvantages of Linear Splines

We will see how the mentioned disadvantages can affect some simulated data using $K=1$. In this case, we will have linear splines, which is linear regression by parts. We will look at how poor knot numbers and location choices can negatively impact model fit.

Let us look at three different situations: the first will be when the knots number is much larger than it should be, the second will be when the knots number chosen is close to the real knots number, and the third will be when the knots number chosen is equal to the real knots number. For each one, we will see the two methods mentioned: the manual choice and the uniform choice of the knots location, explained in Sections 2.4.1 and 2.4.2.

The simulated dataset will be divided into training and testing in all analyses, with 140 observations for training and 60 for testing to calculate the mean square error and standard deviation using the generated and estimated models. The generated model will always have two knots, and with that, we will compare in which situations the approach can be used and when it is likely to be the wrong choice. Also, in the graphs presented, black dots refer to training observations, while

green dots refer to test observations, and the knots location will be represented in the figure as dashed lines. As examples, we will consider just a single explanatory variable to analyze how the usual approach fits these models.

The analysis will compare the mean square error (MSE) and standard deviation for the test set of the generating model with the estimated model. In addition, we will compare the curves visually to understand if, in fact, the estimated model adequately understood the behavior of the data and consequently making realistic predictions. Also, all pre-established knots choices will be based on possible situations in practice, chosen based on some software, by specialists, or by someone trying to understand the splines regression model.

The model that generated the data is provided below:

$$Y_i = 1.48 + 0.19x_{i1} - 0.28(x_{i1} - 18.21)I(x_{i1} > 18.21) + 0.32(x_{i1} - 88.81)I(x_{i1} > 88.81) + \epsilon_i \quad (2.18)$$

where the errors have $\epsilon_i \sim \text{Normal}(0, 2)$.

The model curve that generated the data is given below, as well as the training and testing data.

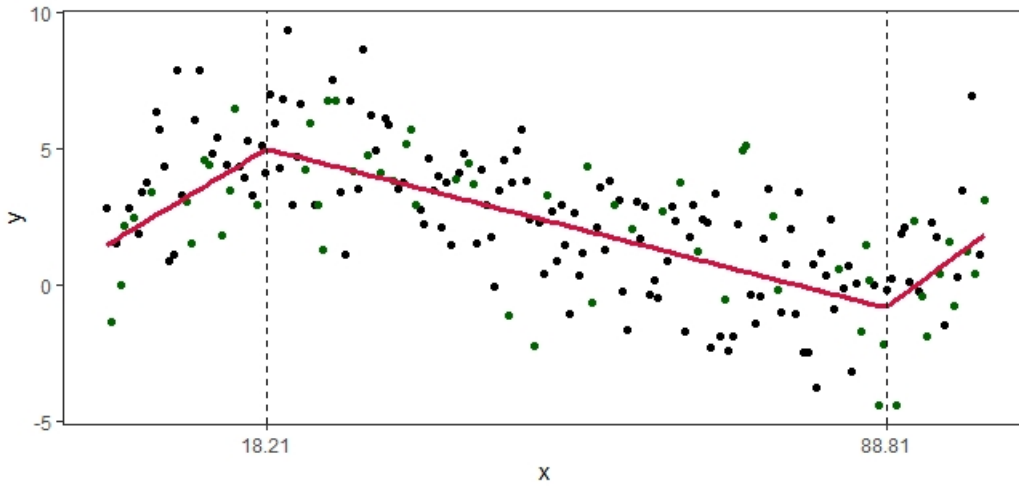


Figure 2.6: *Linear Splines Simulated*

For this model, the pre-specified knots are equal to 18.21 and 88.81, as shown in the figure. The MSE and standard deviation for the test set are equal to 3.84 and 5.31, respectively. Therefore, the metrics of all models estimated with $K=1$ will be compared with these values. We will look at several examples of how the two methods used in the literature behave with the generated data.

2.5.1 Manual choose

For the manual choice method, we will see the three situations mentioned above.

The first situation will be analyzed when the knots number is greater than two.

In Figure 2.7, the knots number chosen for the fit was eight, six knots more than the real knots number. The pre-specified knot locations are 10, 22, 30, 41, 50, 71, 81, and 90. The MSE and the standard deviation of the test set for the estimated model are equal to 4.88 and 6.42, respectively. These metrics are more or less a unit away from the generated model metrics, and despite obtaining an estimated curve close to the real curve, improvements should be made, such as decreasing the number of parameters and choosing the knots location more conveniently, thus improving the predictions.

The second situation will be when the estimated knots number is close to the number of the

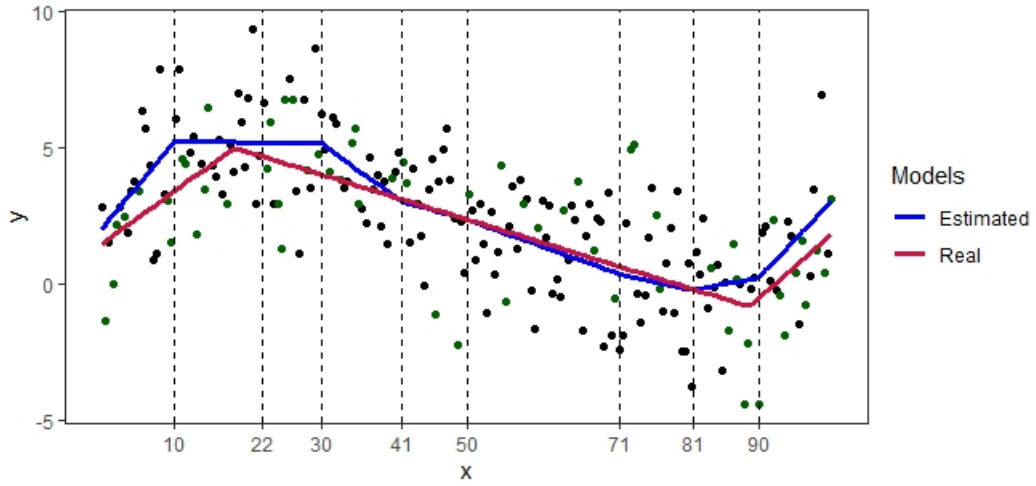


Figure 2.7: *Linear Splines with Manual Choice First Situation*

real knots.

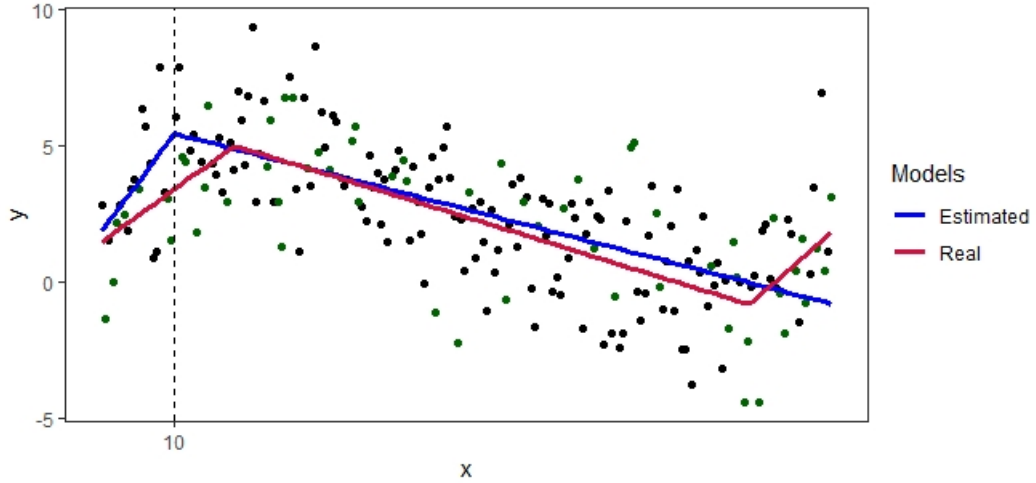


Figure 2.8: *Linear Splines with Manual Choice Second Situation*

In Figure 2.8, the knots number chosen for the fit was one, one knot less than the actual knots number. The pre-specified knot location is 10, a reasonable choice based visually on the change points of the scatter plot. The MSE and standard deviation for the test set of the estimated model are equal to 4.58 and 5.94, respectively. The test MSE of the estimated model is a little higher than the real model, and the standard deviation is relatively close to the standard deviation of the generated model. However, the estimated curve presents a visible error: it did not detect the second point of change of the real model, and it will probably return more considerable errors when the explanatory variable is close to this region. Therefore, if the estimated model had correctly estimated the knots number, it would perform better for the simulated data.

The third situation will be when the knots number in the estimated model is equal to the knots number in the real model.

In Figure 2.9, the knots number chosen for the fit was two, the same knots number as the real model. The locations of the pre-specified knots are at 30 and 63, plausible choices for this case. The MSE and standard deviation for the estimated model test set are equal to 5.2 and 7.31, respectively. The test MSE of the estimated model is much larger than the real model, and the standard deviation for the test set is much larger than the standard deviation (exactly two units larger) of the real model. Despite the knots number in the estimated model being the same as in the real model, the

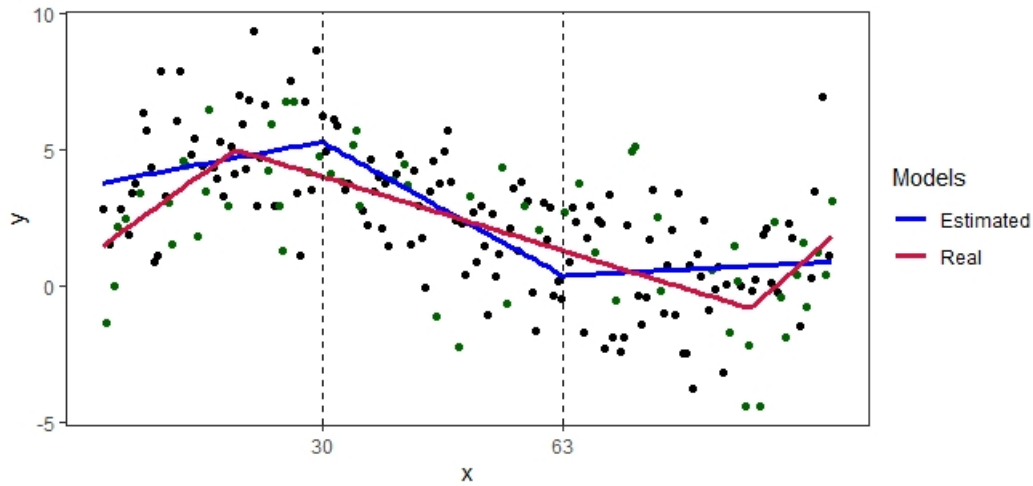


Figure 2.9: *Linear Splines with Manual Choice Third Situation*

locations made the fit much worse, causing the model not to perform predictions efficiently. So, this is an excellent example that even with the knots number being adequate, it is useless if the knots location is terrible.

In conclusion, when we are dealing with linear splines, the manual choice of both the location and the knots number is essential because, through it, the fit may be able to present an adequate performance or not. Furthermore, even if reasonable choices are made, they have limitations and cause serious problems. Therefore, a methodology capable of overcoming this problem must undoubtedly be considered to make predictions more suitable for practical problems.

2.5.2 Uniform choose

Now, let us look at the three situations again with the uniform choice method.

The first situation will be when the knots number used is much higher than the knots number in the real model.

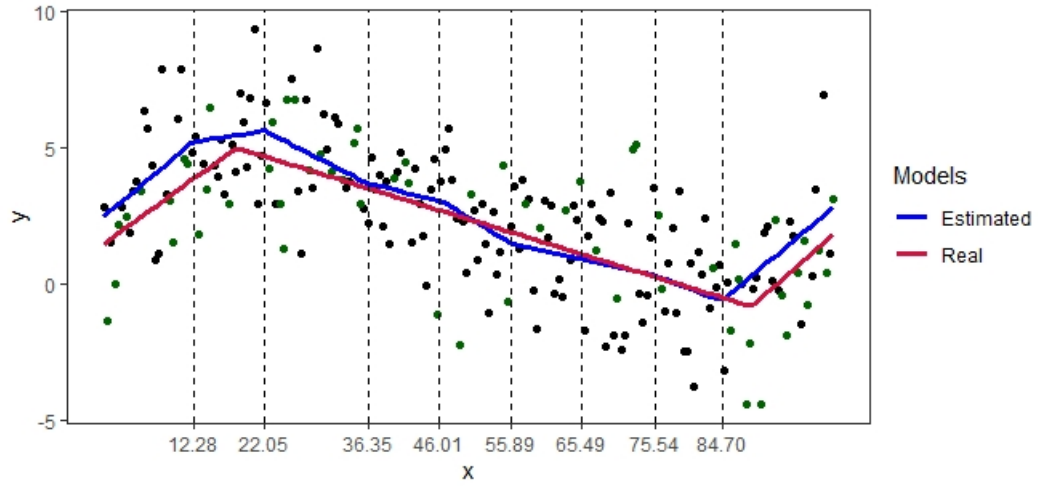


Figure 2.10: *Linear Splines with Uniform Choice First Situation*

In Figure 2.10, the knots number chosen for the fit was eight, six knots more than the actual model's knots number. The locations of the pre-specified knots are: 12.28, 22.05, 36.35, 46.01, 55.89, 65.49, 75.54, and 84.70. The MSE and standard deviation for the test set of the estimated model are equal to 4.98 and 6.64, respectively. The test MSE of the estimated model is greater than the real model, with more than one unit of difference, and the standard deviation for the test set is greater than the standard deviation of the real model, with more than one unit of difference as well. Although the estimated curve is relatively close to the real curve, the knots number in the estimated model is much higher than the real model, leading to some problems such as interpretability and an increase in the variance of the estimate. Therefore, if the model had the most suitable knots number, the performance could improve.

The second situation will be when the knots number of the estimated model is close to the real model.

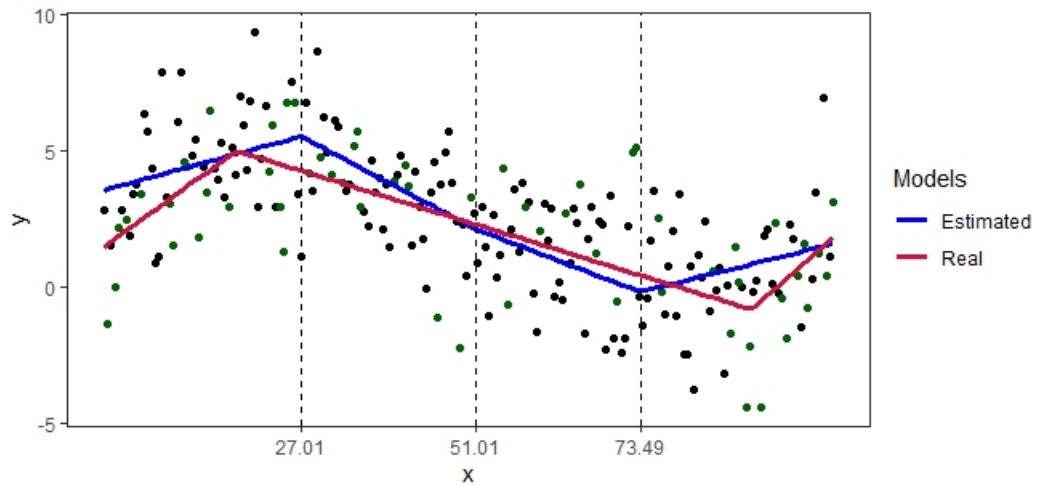


Figure 2.11: *Linear Splines with Uniform Choice Second Situation*

In Figure 2.11, the knots number chosen for the fit was three, only one knot more than the actual model's knots number. The locations of the pre-specified knots are 27.01, 51.01, and 73.49. The MSE and standard deviation for the test set of the estimated model are equal to 5.36 and 7.82, respectively. The test MSE of the estimated model is much larger than the real model, about one and a half units larger, and the standard deviation for the test set is much larger than the standard deviation of the real model, with more than two units larger as well. Although the knots

number in the estimated model is close to the real model, the knots location made the fit worse, leading to some problems with error variance. Therefore, even the knots number of the estimated model is adequate, and the performance could improve with the knots location being more assertive.

The third situation will be when the knots number used is equal to the knots number in the real model.

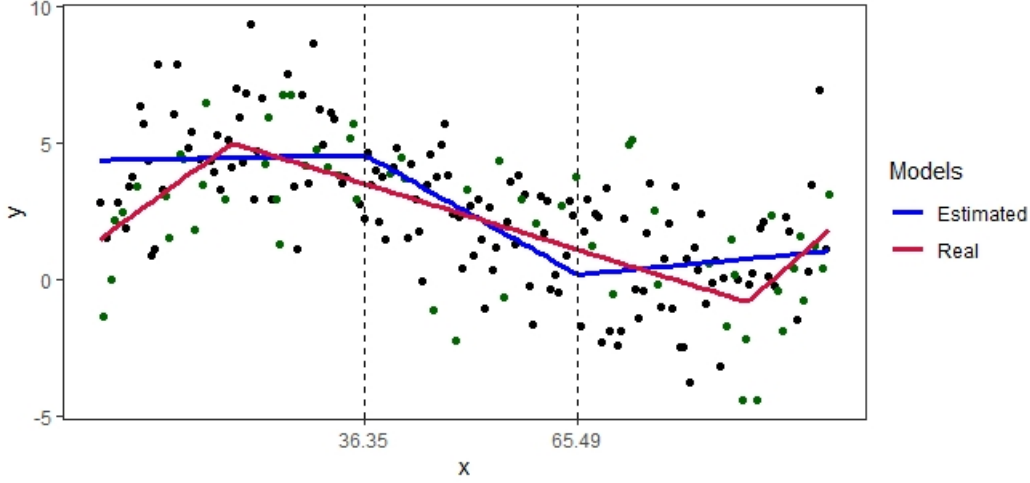


Figure 2.12: *Linear Splines with Uniform Choice Third Situation*

In Figure 2.12, the knots number chosen for the fit was two, the same knots number as the knots number in the real model. The pre-specified knot locations are 36.35 and 65.49. The MSE and standard deviation for the estimated model test set are equal to 5.53 and 7.97, respectively. The estimated model's test MSE is much larger than the real model, almost two units larger, and the standard deviation for the test set is much larger than the standard deviation of the actual model, with more than two and a half units larger than the standard deviation of the real model. Therefore, although the knots number in the estimated model is the same as the real model, the knots location did not achieve good performance, leading to some error variance problems. So this is another example where knot location negatively impacts even the number of estimated model knots is adequate. Again, predictive performance could improve if knots locations were more accurate.

These examples showed that uniform choices for this type of model do not work very well, as they make a big difference in the fit and can affect predictions. The results showed that the error and standard deviation metrics were not close enough to be considered satisfactory. Therefore, we have the intuition that the choice of location and knots number can be improved.

In summary, six examples of the linear spline regression model were shown where six different situations were verified: three different situations where the knots number used in comparison to the knots number in the real model and also using two methods, the first choosing the knots location manually and the second choosing the knots location uniformly. These examples can happen in practice, and we must be cautious when making this choice, as the model can perform poorly because of poor choices. This dissertation proposes a new methodology to consider knots and the knots number as the parameters. This way, it will obtain them numerically concerning a specific cost function. More details will be covered in the methodology proposed in the next section.

2.6 Disadvantages on Cubic Splines

We will see how the mentioned disadvantages can affect some simulated data using $K = 3$. In this case, we will have cubic splines. Let us see how poor knot number and location choices can nega-

tively impact model fit.

As in the previous section, we will see the same three situations illustrated: when the knots number is much larger than it should be when the knots number chosen is close to the real knots number, and when the knots number chosen is equal to the real knots number. We will see the two methods mentioned for each: the manual choice and the uniform choice of knot location explained in sections 2.4.1 and 2.4.2.

The simulated dataset will be divided into training and testing in all analyses, with 140 observations for training and 60 for testing to calculate mean square error and standard deviation using the estimated model. The generated model will always have two knots, and with that, let us compare in which situations the approach can be used and when it is likely to be the wrong choice. Also, in the graphs presented, black dots refer to training observations while green dots refer to test observations, and the knots location will be represented in the figure as dashed lines.

The analysis will compare the mean square error (MSE) and standard deviation for the test set of the generation model with the estimated model. Also, let us compare the curves visually to understand if, in fact, the estimated model adequately understood the behavior of the data and therefore making realistic predictions. In addition, all pre-set knots choices will be based on possible situations in practice, chosen based on some software or reasonable user choices.

The fit that generated the data is provided below:

$$Y_i = 0.33 + 0.56x_{i1} - 0.02x_{i1}^2 + 0.02(x_{i1} - 13.64)^3 I(x_{i1} > 13.64) + 0.01(x_{i1} - 82.63)^3 I(x_{i1} > 82.63) + \epsilon_i \quad (2.19)$$

where the errors have $\epsilon_i \sim Normal(0, 2)$.

The fit curve that generated the data is given below, as well as the training and testing data.

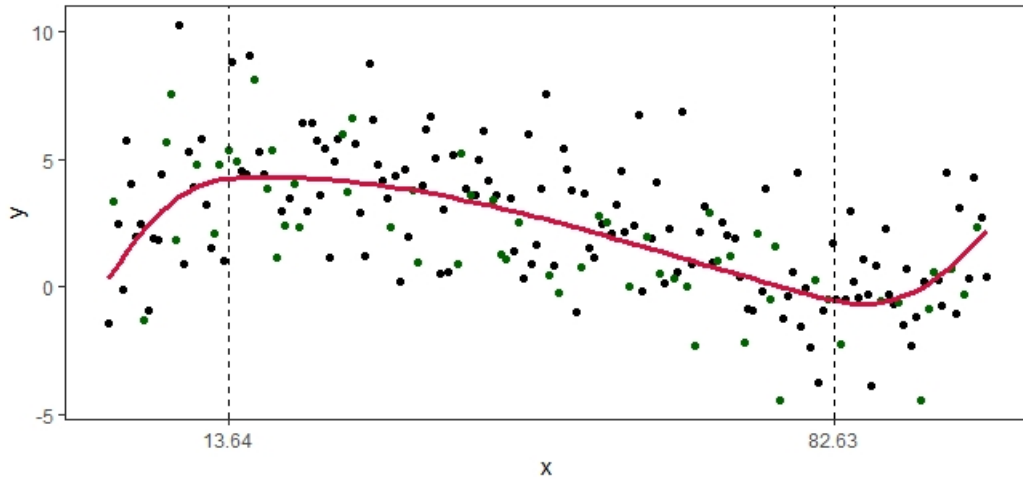


Figure 2.13: *Cubic Splines Simulated*

For this fit, the pre-specified knots are equal to 13.64 and 82.63, as shown in the Figure 2.13. The MSE and standard deviation for the test set are equal to 3.68 and 4.94, respectively. Therefore, the metrics of all models estimated with $K=3$ will be compared with these values. We will look at several examples of how the two methods used in the literature behave with the generated data.

2.6.1 Manual choose

For the manual choice method, we will see the three situations mentioned above. The first situation will be analyzed when the knots number is greater than two.

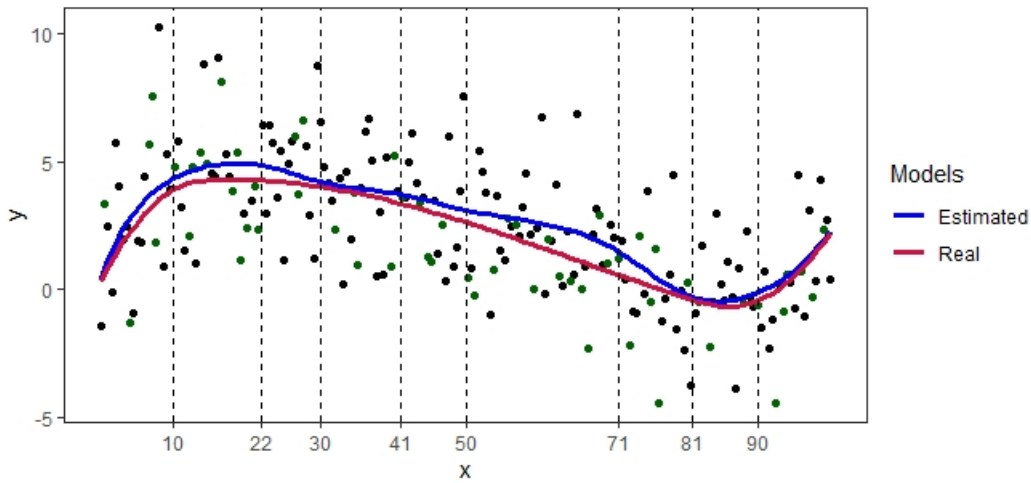


Figure 2.14: *Cubic Splines with Manual Choice First Situation*

In Figure 2.14, the knots number chosen was eight, six more than the knots number in the real model. The locations of the pre-specified knots are at 10, 22, 30, 41, 50, 71, 81 and 90. The MSE and standard deviation in the test set were 4.42 and 5.7, respectively. Both metrics are larger than the real model but not so much. This is due to the flexibility introduced by the nonlinearity, even if the number and knots location are not adequate. However, there are problems with this approach: first, there are more parameters than necessary. In the estimated model, there are twelve parameters, while in the real model, there are only six parameters, twice as many parameters, which leads to prediction variance problems. Also, this can lead to overfitting if we consider that there will be more variables and consequently more parameters to be estimated.

The second situation will be when the estimated knots number is close to the number of the real knots.

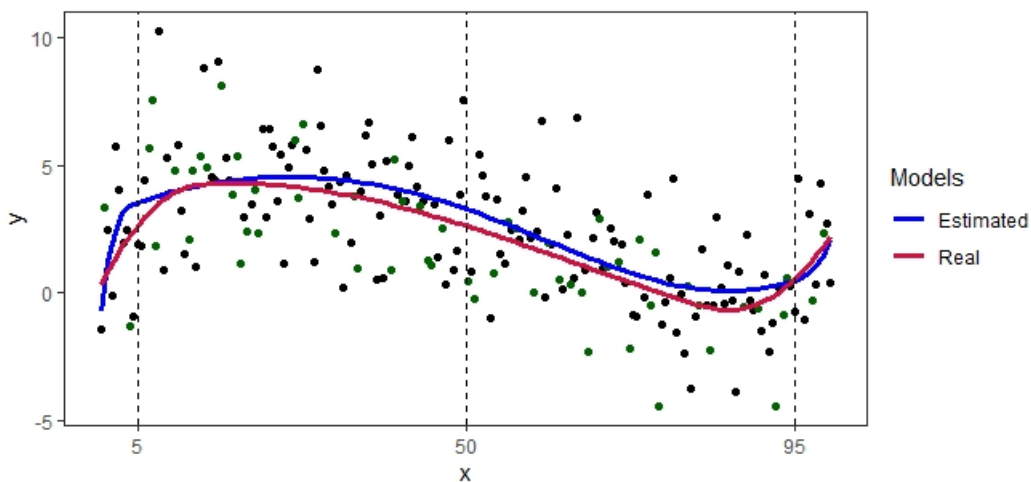


Figure 2.15: *Cubic Splines with Manual Choice Second Situation*

In Figure 2.15, the knots number chosen was three, one more than the actual model knots. The pre-specified knot locations are 5, 50, and 95. The MSE and standard deviation in the test set were 4.45 and 5.86, respectively. The estimated curve is close to the curve that generated the data and

has excellent performance. However, it may be possible for the fit to be much better with the knots number better. Also, knots were not well chosen for the problem, e.g. knot at 50 is unnecessary. So it is intuitive what changes can be made to improve predictive performance.

The third situation will be when the knots number in the estimated model is equal to the the knots number in the real model.

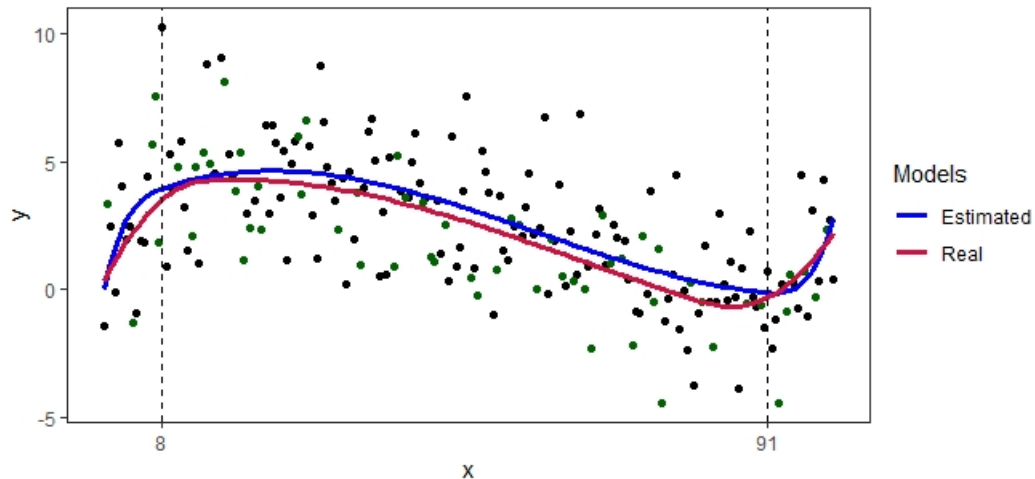


Figure 2.16: *Cubic Splines with Manual Choice Third Situation*

In the Figure 2.16, the knots number chosen was two, the same knots number as the model that generated the data. The pre-specified knot locations are 8 and 91. The MSE and standard deviation for the test set were 4.28 and 5.65, respectively. The estimated model performs well in the predictive sense, and the estimated curve is very close to the real curve. The flexibility helped the model enough not to need the knot locations close to the actual knots. That way, if the model had the pre-specified knots closer to the real knots, the predictions would be even better.

For being suitable for non-linear problems, the model in question generally presents better results than linear splines. Although the number and knots location are not assertive, they are comfortable allocating any amount of knots and presenting a result that is not bad.

2.6.2 Uniform choose

Let us look at the three situations again with the uniform choice method.

The first situation will be when the knots number used is much higher than the knots number in the real model.

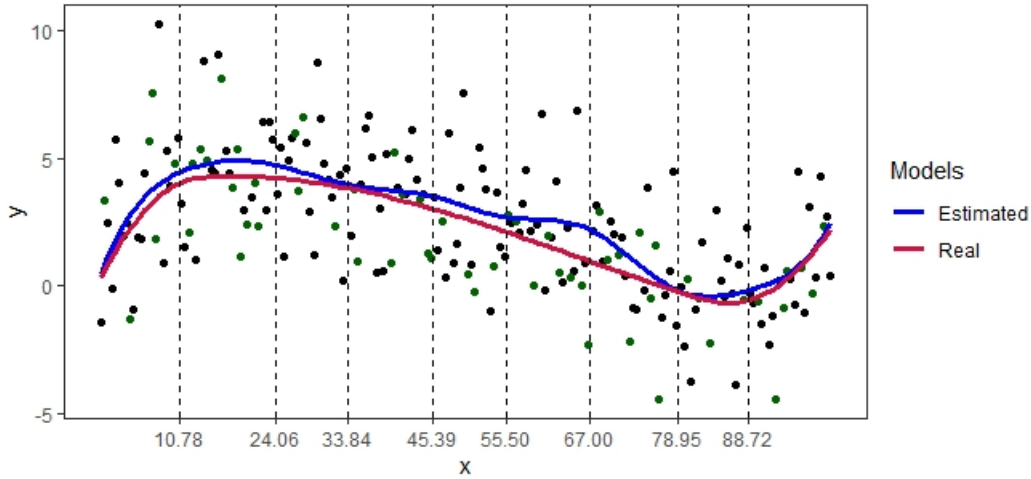


Figure 2.17: *Cubic Splines with Uniform Choice First Situation*

In the Figure 2.17, the knots number chosen was eight, six knots more than the model that generated the data. Pre-specified knot locations are 10.78, 24.06, 33.84, 45.39, 55.50, 67.00, 78.95, and 88.72. The MSE and standard deviation for the test set are 4.42 and 5.63, respectively. These metrics indicate that the estimated model obtained an adequate predictive performance. Despite having an excellent approximation to the real model, there are some problems, such as increasing unnecessary parameters. The estimated model uses twelve parameters, while the real model uses only six, twice as much. The excess of estimated parameters can lead to overfitting on some occasions, especially when we have more parameters of other variables. Also, the variance of predictions is constantly a problem when we have more parameters than necessary.

The second situation will be when the estimated knots number is close to the number of the real knots.

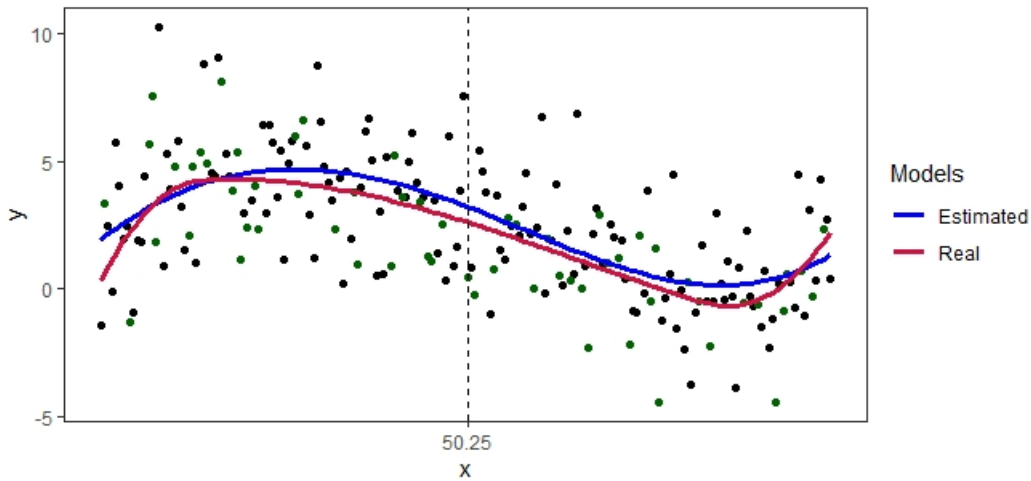


Figure 2.18: *Cubic Splines with Uniform Choice Second Situation*

In the Figure 2.18, the knots number chosen was one, one knot less than the model that generated the data. The pre-specified knot location is at 50.25. The MSE and standard deviation for the test set are 4.36 and 5.79, respectively. Close to the metrics of the real model, thus obtaining stunning results, even with one not less than the real model. In this case, the nonlinearity helps a lot the performance of the model, making it not necessary that the knots number is equal to the real model to obtain satisfactory performance.

The third situation will be when the knots number in the estimated model is equal to the knots number in the real model.

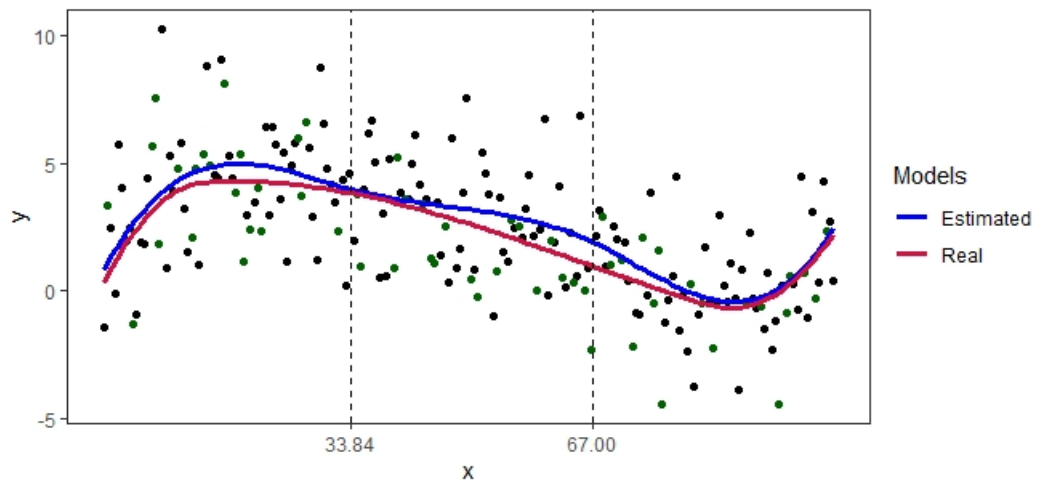


Figure 2.19: *Cubic Splines with Uniform Choice Third Situation*

In the Figure 2.19, the knots number chosen was two, the same knots number in the model that generated the data. The locations of the pre-specified knots are at 33.84 and 67.00. The MSE and standard deviation for the test set are 4.35 and 5.6, respectively. The nonlinearity allows for adequate performance, even with the knots location not being close to real knots.

Therefore, there is always a doubt whether it is possible to improve this specific component or not. The knots location does not significantly worsen the performance of the model when it is nonlinear, but in some cases, it can make much difference, so the ideal choice of the knots location is essential for the spline regression model and should be studied in detail to obtain significant improvements in the predictions.

In this dissertation, we want to improve the performance of the regression splines model to present even better results. We developed a method for this purpose considering the number and knots location as parameters, covered in detail in the next section.

Chapter 3

Estimation of the Number and Location of Knots

The splines regression model has numerous advantages in its use for prediction purposes. However, as seen in Section 2.4, some clear disadvantages can make its use difficult or inefficient. Traditional methods use location of knots as fixed values, pre-specified before the estimation process. These methods can lead to a wrong specification of the model and consequently lead to overfitting or models that can be improved.

This dissertation proposes an increase in the flexibility of the splines regression model to work around the problems of the usual methodology. The most significant difference in the methodology of this dissertation is that we consider both the location of knots and the number of knots of each variable as parameters. In this way, we have the chance to improve predictions and parameter interpretations significantly. Also, we introduce new cost functions for this problem to solve the bias and variance model problem and, consequently, solve the overfitting/underfitting.

The functional form of the proposed model is the same defined in (2.3). However, now we have more parameters to estimate due to the number and location of knots are parameters.

The parameters will be defined by the vector $\theta = [\beta_0, \beta_{(1,1)}, \dots, \beta_{(p, \alpha_p + K)}, t_{11}, \dots, t_{p\alpha_p}, \alpha_1, \dots, \alpha_p] \in R^{\sum_{j=1}^p [2\alpha_j + K_j] + 1} XN^p$. The knots are denoted by t_{jm} for j-th explanatory variable and m-th knot in this specific variable. Now knots will be estimated like any other parameter. We believe that this approach should lead to improvements in predictions and interpretability of the change points.

3.1 Cost functions

As with any complex model, there is a need to select the number of knots appropriately and penalize variables that have a lot of irrelevant knots or that harm the modeling in some way. If we simply choose the number of knots without a regularization, the model will choose as many knots as possible, leading to overfitting.

Therefore, it is essential to study some regularizations to make the model viable and choose the bias and variance accordingly. We propose two regularization functions for a quadratic cost function, chosen based on analytic facilities.

The two cost functions are provided below:

$$J_1(\theta) = \frac{1}{n} \sum_{i=1}^n \left[y_i - \beta_0 - \sum_{j=1}^p f_j(x_{ij}) \right]^2 + \sum_{j=1}^p \alpha_j \lambda_j \sum_{m=1}^{\alpha_j} |\beta_{(j,m+K)}| \quad (3.1)$$

and

$$J_2(\theta) = \frac{1}{n} \sum_{i=1}^n \left[y_i - \beta_0 - \sum_{j=1}^p f_j(x_{ij}) \right]^2 + \frac{\lambda \sum_{j=1}^p \alpha_j}{\log(n)} \quad (3.2)$$

The two functions have a quadratic loss, but they have different regularizations, suitable for each situation we are dealing with. The first cost function given in (3.1) deals in more detail with bias and variance of the model, as hyperparameters control these components. The hyperparameters $\lambda_j, \forall j = 1, \dots, p$ deal directly with each variable, and we can have different biases and variances for each different combination of these hyperparameters. Note that this regularization includes the numbers of knots denoted by α_j , such that variables containing many knots are penalized accordingly and penalize the $\beta_{(j,m+K)} \forall j = 1, \dots, p, m = 1, \dots, \alpha_j$ by the L1 regularization. It is important to note that the only parameters that are penalized are related to knots, which are the parameters that multiply the base truncation functions, as discussed in [17].

Furthermore, the behavior of the regularization is very similar to other regularizations known in the literature. If $\lambda \rightarrow 0$, the cost function penalizes fewer variables with many knots and tends to choose as many knots as possible, leading to overfitting. In this case, the model will have less bias and a greater variance. On the other hand, if $\lambda \rightarrow \infty$ the cost function penalizes the variables with many knots and tends to choose the smallest number of knots possible. As we are dealing with an optimization problem to compensate for this value, the first part of the sum would tend to zero and thus cause underfitting. In this case, the model will have a higher bias and a lower variance.

To illustrate, let us show an example with three knots and then estimate the model with three different λ , equal to 0.01, 10, and 1000 to see how the estimation is performed based on the bias and variance of each curve.

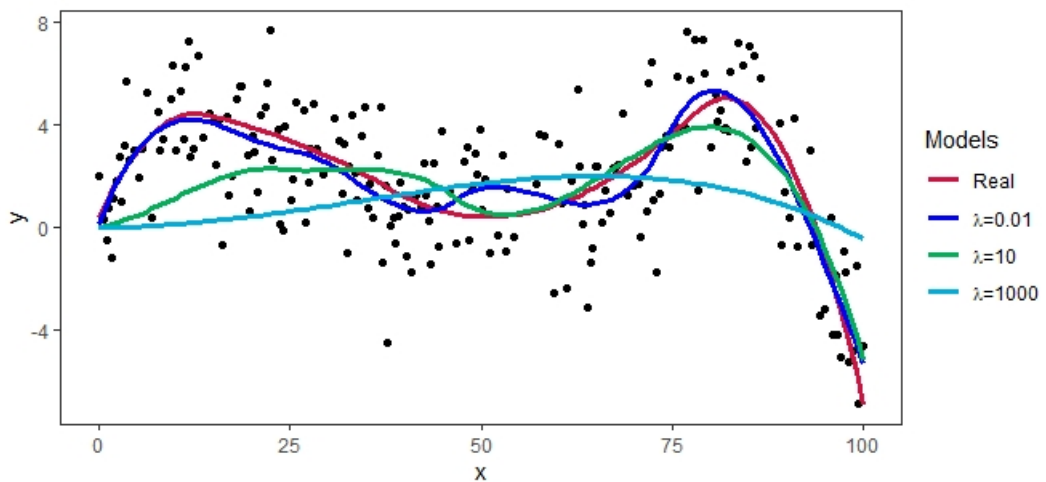


Figure 3.1: Estimated curves using $J_1(\theta)$ with different λ values

In Figure 3.1 we can see that the estimation does exactly what was mentioned in relation to bias and variance. The higher the λ , the more the model has a higher bias and lower variance. The curve in the dark blue, with $\lambda = 0.01$ shows a fair estimate of the real model, with slight bias and a larger variance, since the green curve with $\lambda = 10$ dramatically reduces the bias in the price of increasing

the variance, the light blue curve, with $\lambda = 1000$, is almost a straight line, with low variance and high bias, thus occurring underfitting.

The second cost function given in (3.2), introduced in [19], deals directly with the total number of knots and with a single hyperparameter λ . In this way, this cost function tries to select the model based only on the number of knots and not deals with bias and variance of the model. It is divided by $\log(n)$ because the scale of the regularization matches the scale of the quadratic cost function and penalizes the model accordingly. Also, in the same way as the last regularization, the higher the λ , the less the cost function penalizes variables with many knots. Therefore, if $\lambda \rightarrow 0$ the cost function penalizes less the variables with many knots. On the other hand, if $\lambda \rightarrow \infty$, the cost function penalizes more the variables with many knots.

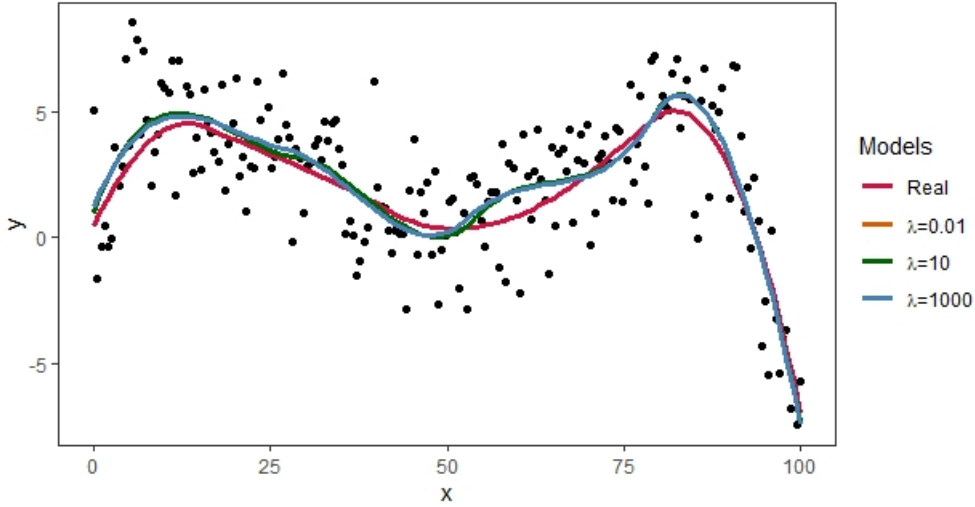


Figure 3.2: Estimated curves using $J_2(\theta)$ with different λ values

In Figure 3.2, as mentioned earlier this regularization deals exclusively with the selection of the number of knots, and as the regularization does not deal with the parameter estimates if we change the value of λ , it will not have much difference in the parameter estimates. Consequently, we will have similar estimated curves.

The first attempt to estimate the parameters is to find some analytical solution through the derivatives. There are three possibilities for the two cost functions of the derivative to any parameter for the two cost functions.

For $J_1(\theta)$, we have:

1. $2 \sum_{i=1}^n \left[\left(y_i - \beta_0 - \sum_{j=1}^p f_j(x_{ij}) \right) x_{ij}^k \right]$
2. $2 \sum_{i=1}^n \left[\left(y_i - \beta_0 - \sum_{j=1}^p f_j(x_{ij}) \right) (x_{ij} - t_{jm})^{K_j} I(x_{ij} > t_{jm}) \right] + \alpha_j \lambda_j \text{signal}(\beta_{(j,m+K_j)})$
3. $-2 \sum_{i=1}^n \left[\left(y_i - \beta_0 - \sum_{j=1}^p f_j(x_{ij}) \right) K_j \beta_{(j,m+K_j)} (x_{ij} - t_{jm})^{K_j-1} I(x_{ij} > t_{jm}) \right]$

The signal function is the derivative of the module function.

For $J_2(\theta)$, we have:

1. $2 \sum_{i=1}^n \left[\left(y_i - \beta_0 - \sum_{j=1}^p f_j(x_{ij}) \right) x_{ij}^k \right]$

2. $2 \sum_{i=1}^n \left[\left(y_i - \beta_0 - \sum_{j=1}^p f_j(x_{ij}) \right) (x_{ij} - t_{jm})^{K_j} I(x_{ij} > t_{jm}) \right]$
3. $-2 \sum_{i=1}^n \left[\left(y_i - \beta_0 - \sum_{j=1}^p f_j(x_{ij}) \right) K_j \beta_{(j,m+K_j)} (x_{ij} - t_{jm})^{K_j-1} I(x_{ij} > t_{jm}) \right]$

Unfortunately, these functions have no analytical solution and need to be solved numerically through some optimization algorithm. Several algorithms can perform this task, such as Newton's Method, Quasi-Newton methods such as BFGS, L-BFGS-B, Broyden, Broyden family, DFP, and SR1, and methods based only on the gradient as a linear approximation such as the gradient descent and conjugate gradient method.

These methods have advantages and disadvantages like any standard algorithm. However, for our problem of minimizing the cost functions given in (3.1) and (3.2), there is an algorithm that stands out from the rest and therefore will be used for our purpose, which will be explained in detail the reasons why we decided to use it instead of any other optimization algorithm.

3.2 Motivation

The first alternative to solve our optimization problem would be the gradient descent optimization algorithm, widely used as a first attempt to optimization problems. It is a standard first-order iterative optimization algorithm. We will denote θ our vector of m parameters. The iterative estimation process is provided below:

$$\theta_{t+1} = \theta_t - \eta \nabla J(\theta_t) \quad (3.3)$$

$$= \theta_t - \eta I_m \nabla J(\theta_t) \quad (3.4)$$

where t represents the iteration, $\eta > 0$ represents the learning rate and $\nabla J(\theta_t)$ represents the vector of the first derivative of the cost function with respect to θ_t .

Its great advantage is often easy to implement because it is a first-order algorithm. However, the gradient descent algorithm has some significant disadvantages:

1. It is a first-order approximation algorithm.
2. It does not consider the Hessian matrix.
3. In practice, it is quite inefficient, even with an adequate learning rate.

We are only working with minimal local information in each iteration by approaching our objective function linearly at each point, so we have to be careful and restrict ourselves to small steps in each iteration. Perhaps we can improve our optimization process by obtaining more local information from the cost function at each iteration to take more accurate steps. The natural solution would be to check the second-order behavior of the cost function.

The best known and widely studied second-order algorithm for this purpose is Newton's Method, whose main added component in relation to gradient descent is the inverse of the Hessian matrix, thus using a quadratic approximation and consequently using more local information for each iteration. It is a better approximation than simply the linear approximation provided by the gradient descent algorithm. The estimation process provided by its damped version is given below:

$$\theta_{t+1} = \theta_t - \eta [\nabla^2 J(\theta_t)]^{-1} \nabla J(\theta_t) \quad (3.5)$$

where t represents the iteration of the estimation process, η is the learning rate, $\nabla^2 J(\theta_t)$ is the Hessian matrix, matrix of second derivatives of $J(\theta_t)$ with respect to θ_t and $\nabla J(\theta_t)$ is the usual gradient of the cost function with respect to θ_t .

The main advantages of Newton's method over gradient descent are that its convergence is superior, quadratic, whereas gradient descent is linear. However, there are still two disadvantages:

1. Calculation of the Hessian matrix in each iteration
2. High computational cost in each iteration

In our case, the exact calculation of the Hessian matrix is challenging. There are many different combinations of parameters, which makes Newton's method impractical. Furthermore, if it were possible to calculate this matrix somehow, we would have to calculate it and then invert it in each iteration and still calculate the gradient. As the number of parameters increases, this process becomes quite computationally costly.

These two methods can potentially be used for various problems. However, they are hardly practical for our problem, but they can help choose another non-linear optimization algorithm using classical optimization ideas. The main component of this goal is the Hessian matrix, which is highly impractical to calculate. Let us see how the two methods behave in computational cost to help us choose a method that fits appropriately for estimating the parameters of our model in question.

3.2.1 Newton's Method vs Gradient Descent: Computational Comparison

There are two ways to evaluate these methods: their efficiency in finding the optimal solution as quickly as possible and the computational time needed to find the solution for each iteration. First, as already mentioned, Newton's Method is much more efficient than the gradient descent algorithm since it uses a better approximation at each iteration, so it will probably reach the optimal solution in much fewer iterations than the gradient descent.

For example, using an arbitrary cost function to find the ideal two-dimensional solution, we have the following level curve results for both methods:

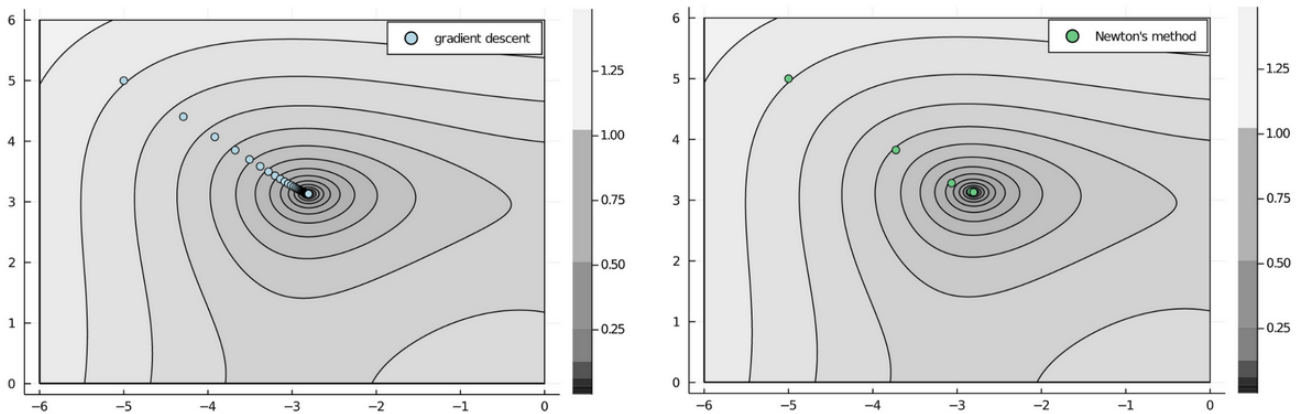


Figure 3.3: *Level curves comparison between gradient descent and Newton's method*

The initial guess was at $\theta_0 = (-5, 5)$, a reasonable choice of the learning rate for the gradient descent algorithm and the undamped version of Newton's Method ($\eta = 1$), we have the following: the gradient descent took 229 iterations to reach the optimal solution, while Newton's method took only six iterations. This is an example where the second approximation made a lot of difference, expected from most optimization problems: Newton's method is much more efficient than gradient descent.

Newton's method is expected to take more time in the calculations for each iteration, as it makes a better approximation, which makes total sense. On the other hand, the gradient descent algorithm

only calculates the gradient in each iteration, which makes an optimization algorithm with much less computational time, which can sometimes be the most viable for some problems. Let us look carefully at the computational complexity of each method.

For the gradient descent algorithm, we have to calculate only the gradient at each iteration, which takes m operations. For Newton's method, in addition to calculating the same gradient as the gradient descent, we have to calculate the Hessian matrix, as it is a symmetric matrix, we have to calculate only half of the matrix values along with the diagonal elements that are precisely: $\frac{m^2-m}{2} + m = \frac{m^2+m}{2} = \frac{m(m+1)}{2}$, which can be considered a relatively large amount for our problem. For example, if we have six explanatory variables, each with 2 knots and we consider $K_j = 3$, we have $\sum_{j=1}^p [2\alpha_j + K_j] + p + 1 = 6 * (2 * 2 + 3) + 6 + 1 = 49$ parameters to estimate, and this would be a more straightforward problem, with few knots in the variables, so we would have $\frac{49*50}{2} = 1225$ operations only to calculate the Hessian in each iteration.

In addition, we have the computation of the inverse of the Hessian matrix, which gives approximately more m^3 operations. In the previous example, we would have to calculate more $49^3 = 117649$ operations.

Thus, the computational complexities of each method are given below:

1. Gradient Descent: $O(m)$, due to the gradient vector addition.
2. Newton's Method: $O(m^3)$, due to the calculation of the Hessian matrix, then invert it.

Therefore, an optimization method that is as efficient as Newton's method but not as computationally expensive is an excellent choice for our problem. Fortunately, there is a class of optimization methods for this purpose called Quasi-Newton methods, which will be widely studied and applied to our optimization problems.

3.3 Quasi Newton Methods

Quasi-Newton methods are among the most widely used methods for non-linear optimization. They are incorporated in many software libraries, and they are effective in solving a wide variety of small to mid-size problems, in particular when the Hessian is hard to compute[9]. They are an optimization model class that aims to be as efficient as Newton's method but without the difficulties faced by this method. The great trick of this class of models is that instead of calculating the Hessian matrix, quasi Newton methods simply approximate it, doing much fewer operations and therefore being computationally faster.

For our problem, being fast is of fundamental importance, as in most cases, we will have a considerable amount of parameters to estimate. Therefore, this method fits our estimation problem very well mainly for two reasons:

1. It solves our difficulty in calculating the Hessian matrix by approximating it.
2. It is as efficient as Newton's method and requires less computational time in each iteration.

Let us describe the iterative estimation process in detail. The quasi Newton method makes use of the second-order approximation, also used by the Newton method, but now we will replace the exact Hessian matrix $\nabla^2 J(\theta_t)$ with its approximate matrix B_t . Therefore, we will make use of the Taylor series with multiple variables. Our objective is to approximate our cost function $J(\theta)$ in the second order to obtain the iteration process of the estimation of the quasi-Newton methods.

We approximate our cost function around d .

$$J(\theta_t + d) \approx m_t(d) \tag{3.6}$$

where $m_t(d)$ is a second order approximation, provided below:

$$m_t(d) = J(\theta_t) + \nabla J(\theta_t)^T d + \frac{1}{2} d^T B_t d \quad (3.7)$$

To find our estimation process, we have to derive $m_t(d)$ with respect to d and equal to 0, so we will approximate the minimum and discover the iterative process.

Let us recall some derivatives involving matrices that we will use in the next step to accomplish this. The derivatives used will be:

$$\frac{\partial \nabla J(\theta_t)^T d}{\partial d} = \frac{\partial d^T \nabla J(\theta_t)}{\partial d} = \nabla J(\theta_t) \quad (3.8)$$

and

$$\frac{\partial d^T B_t d}{\partial d} = (B_t + B_t^T) d = 2B_t d \quad (3.9)$$

The Hessian matrix is symmetric, and therefore the quasi Newton methods also require B_t to be symmetric, and each algorithm varies in how it requires it to be symmetric.

Thus, we have the following expression for the derivative of $m_t(d)$:

$$\frac{\partial m_t(d)}{\partial d} = \nabla J(\theta_t) + B_t d \quad (3.10)$$

Equating the expression (3.10) to 0:

$$\frac{\partial m_t(d)}{\partial d} = 0 \Leftrightarrow \nabla J(\theta_t) + B_t d = 0 \quad (3.11)$$

and isolating d , the iteration will be given by:

$$d = -B_t^{-1} \nabla J(\theta_t) \quad (3.12)$$

The iteration of the quasi Newton methods is almost identical to the Newton's method, except that we now have the matrix that approximates the Hessian matrix. Finally, the iterative procedure of a quasi Newton algorithm is given by:

$$\theta_{t+1} = \theta_t - \eta_t B_t^{-1} \nabla J(\theta_t) \quad (3.13)$$

The η_t is found by the backtracking line search method [16]. The backtracking line search starts with a large η_t estimate and iteratively shrinks it. The shrinking continues until a value is found that is small enough to provide a decrease in the objective function that adequately matches the decrease that is expected to be achieved, based on the local function gradient $\nabla J(\theta_t)$.

The numerous quasi-Newton methods differ in how to estimate B_t . Moreover, B_t is also estimated iteratively, after estimating θ_t . There are several advantages to this approach. First, an approximation B_t can be found using only first-derivative information. Second, the complexity time of the quasi Newton methods is only $O(m^2)$ while Newton's method is $O(m^3)$. Therefore, the time to reach the minimum will be faster and, consequently, more efficient in most situations. There are also disadvantages, but they are minor. The methods do not converge quadratically, but they can converge superlinearly. At the precision of computer arithmetic, there is not much practical difference between these two convergence rates [9].

Now that we obtain the generic iterative estimation process for quasi Newton methods, we would like to provide the advantages in detail over Newton's method and construct the approximation

of B_t reasonably enough to estimate θ_t plausibly. These advantages are on the ease of calculation in each iteration and the iterative approximation of B_t , so the algorithms of this class will be as efficient as Newton's method and adequately solve our non-linear optimization problem.

Therefore, these requirements can be summarized in two main objectives:

1. The calculation of $B_t^{-1}\nabla J(\theta_t)$ should be easier to calculate than Newton's method, so we have a computational advantage.
2. The second-order approximation $m_t(d)$ should capture the curvature information of $J(\theta)$ along the iterations.

The first requirement concerns the computational time needed in each iteration to be easier than Newton's method. The second considers the approximation of the exact Hessian matrix to obtain the information necessary for our estimation without calculating it.

We will focus on the second requirement, while the computational advantages will be explained soon.

The main idea for this condition is to construct a procedure such that $m_{t+1}(d)$ corresponds to the gradient of $J(\theta_{t+1})$ of the last two iterations using second order approximation given by $J(\theta_{t+1} + d) \approx m_{t+1}(d)$. Therefore, we would like the gradient of $m_{t+1}(d)$ at point $\mathbf{0}$ to be exactly equal to the gradient of our cost function $J(\theta_{t+1})$ and also, we wish that the gradient of $m_{t+1}(d)$ corresponds to the gradient of $J(\theta_t)$, which is exactly a previous iteration, and we do this by evaluating efficiently in the corresponding update given by $\eta_t B_t^{-1}\nabla J(\theta_t)$.

Specifically, we require the following two conditions:

1. $\nabla m_{t+1}(\mathbf{0}) = \nabla J(\theta_{t+1})$
2. $\nabla m_{t+1}(\eta_t B_t^{-1}\nabla J(\theta_t)) = \nabla J(\theta_t)$

Let us carefully check what these conditions imply in the iterative estimation process.

Using (3.7) and (3.10) we have that the first condition implies:

$$\nabla m_{t+1}(\mathbf{0}) = \nabla J(\theta_{t+1}) + B_{t+1}\mathbf{0} = \nabla J(\theta_{t+1}) \quad (3.14)$$

Therefore, the first condition is automatically satisfied.

The second condition is given by:

$$\nabla m_{t+1}(\eta_t B_t^{-1}\nabla J(\theta_t)) = \nabla J(\theta_{t+1}) + B_{t+1}\eta_t B_t^{-1}\nabla J(\theta_t) = \nabla J(\theta_t) \quad (3.15)$$

$$= \nabla J(\theta_{t+1}) - \nabla J(\theta_t) = -B_{t+1}\eta_t B_t^{-1}\nabla J(\theta_t) \quad (3.16)$$

Using the iterative estimation process given in (3.13), we have the following result:

$$\theta_{t+1} = \theta_t - \eta_t B_t^{-1}\nabla J(\theta_t) \Leftrightarrow \theta_{t+1} - \theta_t = -\eta_t B_t^{-1}\nabla J(\theta_t) \quad (3.17)$$

Thus, replacing (3.17) in (3.15) we have:

$$\nabla J(\theta_{t+1}) - \nabla J(\theta_t) = -B_{t+1}\eta_t B_t^{-1}\nabla J(\theta_t) \quad (3.18)$$

$$\nabla J(\theta_{t+1}) - \nabla J(\theta_t) = B_{t+1}(\theta_{t+1} - \theta_t) \quad (3.19)$$

$$\bar{J}_t = B_{t+1}\bar{\theta}_t \quad (3.20)$$

Where $\bar{J}_t = \nabla J(\theta_{t+1}) - \nabla J(\theta_t)$ and $\bar{\theta}_t = (\theta_{t+1} - \theta_t)$.

That is a non-trivial condition, and therefore it should be used as a constraint in obtaining the matrix B_{t+1} by all the algorithms of the quasi-Newton class. This condition is popularly known in non-linear optimization by the secant equation. Furthermore, the secant equation requires that

B_{t+1} be positive definite, and each method enforces it differently.

A specific condition to make B_{t+1} positive defined can be satisfied if:

$$\bar{\theta}_t^T B_{t+1} \bar{\theta}_t = \bar{\theta}_t^T \bar{J}_t > 0 \quad (3.21)$$

This condition is known as the curvature condition. We are using that $\bar{\theta}_t$ is a particular case of a non-zero vector and satisfies a particular condition to be positive definite. Therefore, each quasi Newton method requires that B_{t+1} be positive definite so that it is also symmetric requiring (3.21) to be satisfied.

The simplest case for the condition (3.19) is when we are estimating only one parameter ($m = 1$), and for this case, we can obtain an exact approximation for the Hessian matrix given by the secant equation:

$$J'(\theta_t) - J'(\theta_{t-1}) = J''(\theta_t)(\theta_t - \theta_{t-1}) \quad (3.22)$$

Isolating $J''(\theta_t)$, we obtain:

$$J''(\theta_t) = \frac{J'(\theta_t) - J'(\theta_{t-1})}{\theta_t - \theta_{t-1}} \quad (3.23)$$

Therefore, if we consider $\eta_t = 1$ the iterative estimation process provided in (3.13) is given by:

$$\theta_{t+1} = \theta_t - \frac{J'(\theta_t)}{J''(\theta_t)} \quad (3.24)$$

Replacing (3.22) in (3.24), we obtain the following iterative process:

$$\theta_{t+1} = \theta_t - \frac{J'(\theta_t)(\theta_t - \theta_{t-1})}{J'(\theta_t) - J'(\theta_{t-1})} \quad (3.25)$$

Thus, we use only the gradient and estimates from the last two iterations and efficiently obtain a second-order estimation process. However, it takes two initial guesses instead of just one, which is not a problem.

Let us apply this method to minimize the function $J(\theta) = \sin(\theta)$. Thus, the specific estimation process for this function is given by:

$$\theta_{t+1} = \theta_t - \frac{\cos(\theta_t)(\theta_t - \theta_{t-1})}{\cos(\theta_t) - \cos(\theta_{t-1})} \quad (3.26)$$

The first two initial guesses are $\theta_1 = 0$ and $\theta_2 = -1$.

Estimates of θ over the iterations are given below:

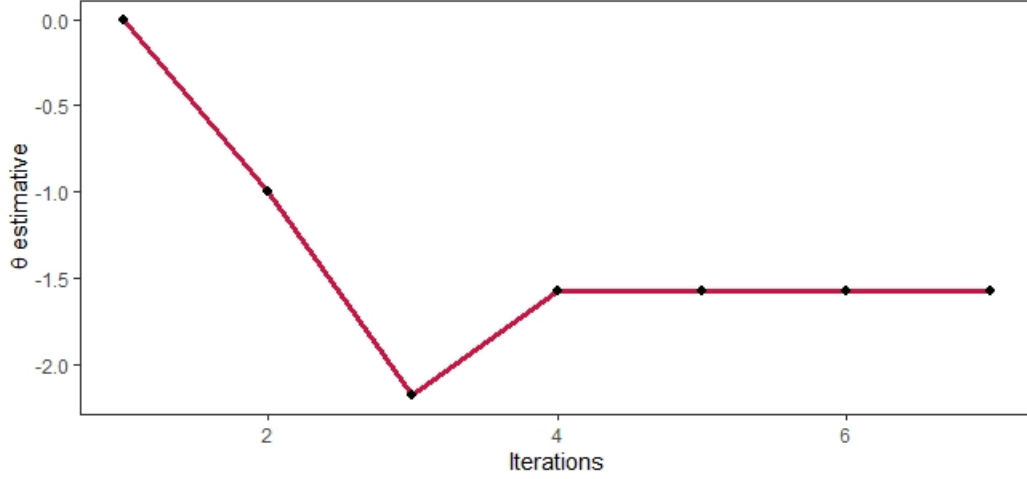


Figure 3.4: θ estimate over the iterations

The estimate converged at $\hat{\theta} = -1.570796 \approx -\frac{\pi}{2}$.

The values of $\sin(\theta)$ over the course of the estimates are given below:

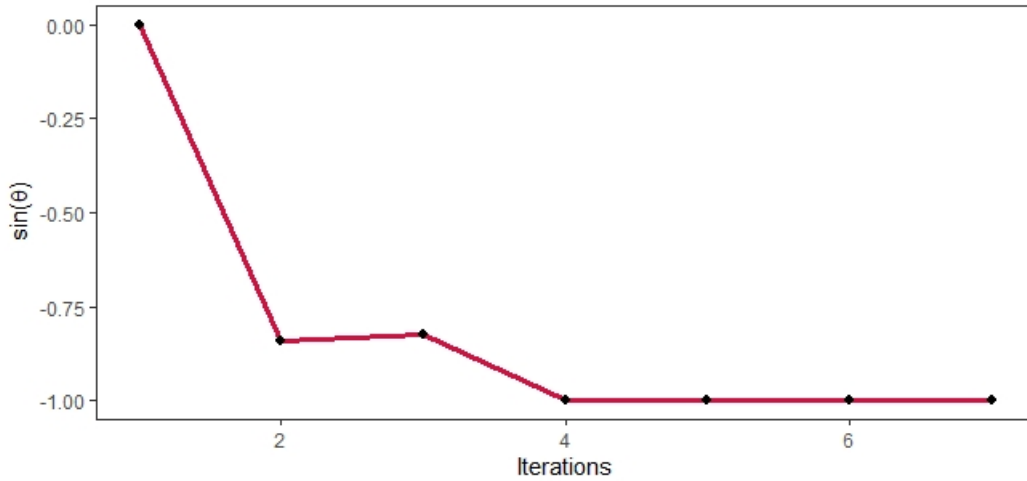


Figure 3.5: $\sin(\theta)$ over the iterations

Since $\sin(\theta) \in [-1, 1], \forall \theta \in R$, then the minimum was found correctly.

Nevertheless, this method cannot be generalized to the case of $m > 1$ dimensions. This is because in the general case, we have to estimate all the elements of the matrix B_t which are exactly $\frac{m(m+1)}{2}$ elements, but the constraint (3.19) has m equations, so this is a system of equations not determined, since we have more elements to estimate than equations available. In the case of $m = 1$, we have $\frac{1(1+1)}{2} = 1$, and we also only have one equation to solve it without problems. In general, we cannot solve this problem precisely for more than one dimension.

Fortunately, the quasi-Newton methods here handle very well to solve this problem of estimating B_t properly. Each method has its particularities and properties to estimate it, differentiating each class algorithm. Additionally, all algorithms usually are faster computationally than Newton's method because they only use gradient information from the two iterations to estimate it.

Another remarkable observation to make is about the convergence of B_t . An excellent wish for our problem is that as B_t passes through the iterations B_t stays close enough to the exact Hessian

matrix. Thus, it would indicate that the estimate of θ was adequate for our problem. For a quadratic convex $J(\theta)$, B_t should approach the exact Hessian as $t \rightarrow \infty$ (i.e., as we apply our iterative evaluation update for many points and many gradients, approaching the minimum). In practice, what can typically be proved [8], [2], [6] is that for a quadratic convex $J(\theta)$, the quasi Newton method gives the exact minimum and the exact Hessian in n steps (in exact arithmetic), which is a powerful and highly beneficial mathematical result.

More specifically, each algorithm has its speed of convergence, situations for the proper use, and estimation of B_t . The generic algorithm is given below so that we will then provide details about a particular one.

Algorithm 2 Generic quasi-Newton

Input: Initial guess θ_1 , a nonsingular B_1 (often the choice is $B_1 = I_m$), and a termination tolerance $\varepsilon > 0$.

$t = 1$

while $\|\nabla J(\theta_t)\| > \varepsilon$ **do**

1. Compute $B_t^{-1}\nabla J(\theta_t)$ by some quasi Newton algorithm.
2. Perform a practical line-search for the minimization to find η_t .
3. Estimate θ_{t+1} iteratively through $\theta_{t+1} = \theta_t - \eta_t B_t^{-1}\nabla J(\theta_t)$
4. Compute the new approximate Hessian B_{t+1} according to the specified rule.
5. $t = t + 1$

end while

Let us carefully analyze a particular quasi Newton algorithm for handling the estimation.

3.4 The BFGS algorithm

For estimating the parameters of our model, we chose the most popular method that best fits our cost function. The most popular quasi-Newton algorithm is the BFGS method, named for its discoverers Broyden, Fletcher, Goldfarb, and Shanno [3], [5], [8], [18] who developed it independently.

It is based on the estimation of B_t through the last two iterations using only information from the estimates and the gradient, ensuring optimal properties, such as B_t being definite positive and minimizing $J(\theta)$ adequately, by superlinear convergence.

We will now detail this algorithm, a particular case of quasi-Newton methods. Therefore, we will need the first advantage of the quasi Newton methods mentioned earlier, which is: The calculation of $B_t^{-1}\nabla J(\theta_t)$ should be easier to calculate than Newton's method, so we have a computational advantage. This component can be summarized in calculating B_t^{-1} because $\nabla J(\theta_t)$ is easier to compute, as there is a fast and straightforward function, requiring only m inputs.

Therefore, our focus will be to define how the BFGS algorithm estimates B_t^{-1} iteratively. Another desired property is that B_{t+1} is close enough to B_t . The most usual way to do this is through a normal matrix. We will then do this by imposing the restrictions of quasi-Newton methods.

Thus, a first attempt is to elaborate the optimization problem written below:

$$B_{t+1} = \underset{B}{\operatorname{argmin}} \|B - B_t\| \quad (3.27)$$

subject to:

$$B = B^T \text{ and } \bar{J}_t = B\bar{\theta}_t \quad (3.28)$$

Which also ensures that B is positive definite. Details will be explained shortly. The choice of the norm is of fundamental importance, through it which we can differentiate the quasi Newton methods. BFGS uses a specific normal called the weighted Frobenius norm, which uses the Frobenius norm. The definitions are provided below.

Frobenius norm:

$$\|A\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^m A_{ij}^2} \quad (3.29)$$

Weighted Frobenius norm:

$$\|A\|_W = \left\| W^{1/2} A W^{1/2} \right\|_F \quad (3.30)$$

The weight matrix W can be chosen as any matrix satisfying the relation $W\bar{J}_t = \bar{\theta}_t$. The BFGS chooses a specific W provided in [9] and [15] derive the iterative process solving (3.27), (3.28) and arrive at the following result for B_{t+1} :

$$B_{t+1} = B_t + U_t + V_t \quad (3.31)$$

where the matrices U_t and V_t have rank 1, and therefore there is a result that guarantees that they can be written as follows:

$$U_t = au_t u_t^T \text{ and } V_t = bv_t v_t^T \quad (3.32)$$

where $u^T v = 0$ (linearly independent) and a, b are scalar and will be found shortly. This process is known in the literature as rank 2 update, which guarantees both symmetry and definite positivity that will be discussed soon.

Replacing (3.32) in (3.31), we have the following:

$$B_{t+1} = B_t + au_t u_t^T + bv_t v_t^T \quad (3.33)$$

Furthermore, we have to impose the quasi-Newton condition (3.19) in (3.33). Therefore, we have:

$$\bar{J}_t = B_{t+1} \bar{\theta}_t \quad (3.34)$$

$$\bar{J}_t = (B_t + au_t u_t^T + bv_t v_t^T) \bar{\theta}_t \quad (3.35)$$

$$\bar{J}_t = B_t \bar{\theta}_t + au_t u_t^T \bar{\theta}_t + bv_t v_t^T \bar{\theta}_t \quad (3.36)$$

The BFGS uses particular choices for u_t and v_t , specifically it uses $u_t = \bar{J}_t$ and $v_t = B_t \bar{\theta}_t$. So, replacing we have the following:

$$\bar{J}_t = B_t \bar{\theta}_t + a \bar{J}_t \bar{J}_t^T \bar{\theta}_t + b B_t \bar{\theta}_t \bar{\theta}_t^T B_t^T \bar{\theta}_t \quad (3.37)$$

Organizing the terms to find a and b , we obtain the following result:

$$\bar{J}_t - a\bar{J}_t\bar{J}_t^T\bar{\theta}_t = B_t\bar{\theta}_t + bB_t\bar{\theta}_t\bar{\theta}_t^TB_t^T\bar{\theta}_t \quad (3.38)$$

$$\bar{J}_t[1 - a\bar{J}_t^T\bar{\theta}_t] = B_t\bar{\theta}_t[1 + b\bar{\theta}_t^TB_t^T\bar{\theta}_t] \quad (3.39)$$

Therefore, the values of a and b result in:

$$\begin{aligned} 1. \quad a &= \frac{1}{\bar{J}_t^T\bar{\theta}_t} \\ 2. \quad b &= -\frac{1}{\bar{\theta}_t^TB_t^T\bar{\theta}_t} \end{aligned}$$

Finally, with all this mathematical procedure, replacing the values of u_t , v_t , a and b in (3.33), we arrive at the final result of the iterative process of B_{t+1} .

$$B_{t+1} = B_t + \frac{\bar{J}_t\bar{J}_t^T}{\bar{J}_t^T\bar{\theta}_t} - \frac{B_t\bar{\theta}_t\bar{\theta}_t^TB_t^T}{\bar{\theta}_t^TB_t^T\bar{\theta}_t} \quad (3.40)$$

We can see that we only use gradient and estimation information. Therefore, this makes the estimation process faster than the usual procedure.

Let us show a significant result of the iterative process of B_{t+1} given in [9]:

Let B_t be a symmetric positive-definite matrix, and that B_{t+1} is obtained from B_t using the BFGS update given in (3.40). Then B_{t+1} is positive definite if and only if $\bar{\theta}_t^T\bar{J}_t > 0$.

BFGS requires an initial guess for B_1 , which is often assigned the identity matrix, which is symmetric and positive definite. Also, B_{t+1} is symmetric in all iterations because it will always be a sum of symmetric and positive definitive matrices due to the previous result because $\bar{\theta}_t^T\bar{J}_t > 0$ occurs in all iterations.. Consequently, B_{t+1}^{-1} will always exist because it will always be symmetric and positive definite. We are just using a known result in matrix algebra.

More specifically :

If A is a positive definite symmetric matrix, then A^{-1} always exists and is also a positive definite symmetric matrix.

Therefore, we never need to worry about B_{t+1} having the inverse, as it will always exist in all iterations. Now, we need to quickly find B_{t+1}^{-1} to compute in each iteration. Fortunately, there is a mathematical formula to calculate it related to the iterative process. This formula is known in matrix algebra as The Woodbury matrix identity [11], given below:

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1} \quad (3.41)$$

Let us organize the iterative process of B_{t+1} to fit (3.41) so that we can calculate its inverse.

$$B_{t+1} = B_t + \frac{\bar{J}_t\bar{J}_t^T}{\bar{J}_t^T\bar{\theta}_t} - \frac{B_t\bar{\theta}_t\bar{\theta}_t^TB_t^T}{\bar{\theta}_t^TB_t^T\bar{\theta}_t} \quad (3.42)$$

$$= B_t + [B_t\bar{\theta}_t, \bar{J}_t] \begin{bmatrix} -\frac{1}{\bar{\theta}_t^TB_t^T\bar{\theta}_t} & 0 \\ 0 & \frac{1}{\bar{J}_t^T\bar{\theta}_t} \end{bmatrix} \begin{bmatrix} \bar{\theta}_t^TB_t^T \\ \bar{J}_t^T \end{bmatrix} \quad (3.43)$$

$$= A + UCV, \quad (3.44)$$

where $A = B_t$, $U = [B_t \bar{\theta}_t, \bar{J}_t]$, $V = \begin{bmatrix} \bar{\theta}_t^T B_t^T \\ \bar{J}_t^T \end{bmatrix}$ and $C = \begin{bmatrix} -\frac{1}{\bar{\theta}_t^T B_t^T \bar{\theta}_t} & 0 \\ 0 & \frac{1}{\bar{J}_t^T \bar{\theta}_t} \end{bmatrix}$

Now, with the determination of the necessary matrices, let us use The Woodbury matrix identity (3.41).

$$B_{t+1}^{-1} = (A + UCV)^{-1} \quad (3.45)$$

$$= A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1} \quad (3.46)$$

First, solving $A^{-1}U$ and VA^{-1} :

$$A^{-1}U = B_t^{-1}[B_t \bar{\theta}_t, \bar{J}_t] \quad (3.47)$$

$$= [B_t^{-1}B_t \bar{\theta}_t, B_t^{-1}\bar{J}_t] \quad (3.48)$$

$$= [\bar{\theta}_t, B_t^{-1}\bar{J}_t] \quad (3.49)$$

$$VA^{-1} = \begin{bmatrix} \bar{\theta}_t^T B_t^T \\ \bar{J}_t^T \end{bmatrix} B_t^{-1} \quad (3.50)$$

$$= \begin{bmatrix} \bar{\theta}_t^T B_t^T B_t^{-1} \\ \bar{J}_t^T B_t^{-1} \end{bmatrix} \quad (3.51)$$

$$= \begin{bmatrix} \bar{\theta}_t^T \\ \bar{J}_t^T B_t^{-1} \end{bmatrix} \quad (3.52)$$

Now we solve $VA^{-1}U$ and C^{-1} , and then add them up:

$$VA^{-1}U = \begin{bmatrix} \bar{\theta}_t^T B_t^T \\ \bar{J}_t^T \end{bmatrix} B_t^{-1}[B_t \bar{\theta}_t, \bar{J}_t] \quad (3.53)$$

$$= \begin{bmatrix} \bar{\theta}_t^T B_t^T B_t^{-1} \\ \bar{J}_t^T B_t^{-1} \end{bmatrix} [B_t \bar{\theta}_t, \bar{J}_t] \quad (3.54)$$

$$= \begin{bmatrix} \bar{\theta}_t^T \\ \bar{J}_t^T B_t^{-1} \end{bmatrix} [B_t \bar{\theta}_t, \bar{J}_t] \quad (3.55)$$

$$= \begin{bmatrix} \bar{\theta}_t^T B_t \bar{\theta}_t & \bar{\theta}_t^T \bar{J}_t \\ \bar{J}_t^T B_t^{-1} B_t \bar{\theta}_t & \bar{J}_t^T B_t^{-1} \bar{J}_t \end{bmatrix} \quad (3.56)$$

$$= \begin{bmatrix} \bar{\theta}_t^T B_t \bar{\theta}_t & \bar{\theta}_t^T \bar{J}_t \\ \bar{J}_t^T \bar{\theta}_t & \bar{J}_t^T B_t^{-1} \bar{J}_t \end{bmatrix} \quad (3.57)$$

Let us use the following matrix result to calculate the inverse of a 2x2 dimensional matrix to calculate C^{-1} and $(C^{-1} + VA^{-1}U)^{-1}$:

If a matrix D has dimension 2x2 and $\det(D) \neq 0$, then:

$$D^{-1} = \frac{1}{\det(D)} \begin{bmatrix} d_{22} & -d_{12} \\ -d_{21} & d_{11} \end{bmatrix} \text{ for } D = \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix}$$

Using that $\det(C) = -\frac{1}{\bar{\theta}_t^T B_t^T \bar{\theta}_t \bar{J}_t^T \bar{\theta}_t}$, we have:

$$C^{-1} = \begin{bmatrix} -\frac{1}{\bar{\theta}_t^T B_t^T \bar{\theta}_t} & 0 \\ 0 & \frac{1}{\bar{J}_t^T \bar{\theta}_t} \end{bmatrix}^{-1} \quad (3.58)$$

$$= \frac{1}{\det(C)} \begin{bmatrix} \frac{1}{\bar{J}_t^T \bar{\theta}_t} & 0 \\ 0 & -\frac{1}{\bar{\theta}_t^T B_t^T \bar{\theta}_t} \end{bmatrix} \quad (3.59)$$

$$= \bar{\theta}_t^T B_t^T \bar{\theta}_t \bar{J}_t^T \bar{\theta}_t \begin{bmatrix} \frac{1}{\bar{J}_t^T \bar{\theta}_t} & 0 \\ 0 & -\frac{1}{\bar{\theta}_t^T B_t^T \bar{\theta}_t} \end{bmatrix} \quad (3.60)$$

$$= \begin{bmatrix} \frac{-\bar{\theta}_t^T B_t^T \bar{\theta}_t \bar{J}_t^T \bar{\theta}_t}{\bar{J}_t^T \bar{\theta}_t} & 0 \\ 0 & \frac{\bar{\theta}_t^T B_t^T \bar{\theta}_t \bar{J}_t^T \bar{\theta}_t}{\bar{\theta}_t^T B_t^T \bar{\theta}_t} \end{bmatrix} \quad (3.61)$$

$$= \begin{bmatrix} -\bar{\theta}_t^T B_t^T \bar{\theta}_t & 0 \\ 0 & \bar{J}_t^T \bar{\theta}_t \end{bmatrix} \quad (3.62)$$

$$C^{-1} + VA^{-1}U = \begin{bmatrix} -\bar{\theta}_t^T B_t^T \bar{\theta}_t & 0 \\ 0 & \bar{J}_t^T \bar{\theta}_t \end{bmatrix} + \begin{bmatrix} \bar{\theta}_t^T B_t \bar{\theta}_t & \bar{\theta}_t^T \bar{J}_t \\ \bar{J}_t^T \bar{\theta}_t & \bar{J}_t^T B_t^{-1} \bar{J}_t \end{bmatrix} \quad (3.63)$$

$$= \begin{bmatrix} 0 & \bar{\theta}_t^T \bar{J}_t \\ \bar{J}_t^T \bar{\theta}_t & \bar{J}_t^T \bar{\theta}_t + \bar{J}_t^T B_t^{-1} \bar{J}_t \end{bmatrix} \quad (3.64)$$

Now, let us calculate $(C^{-1} + VA^{-1}U)^{-1}$:

Using $\det(C^{-1} + VA^{-1}U) = -\bar{\theta}_t^T \bar{J}_t \bar{J}_t^T \bar{\theta}_t$, we have:

$$(C^{-1} + VA^{-1}U)^{-1} = \begin{bmatrix} 0 & \bar{\theta}_t^T \bar{J}_t \\ \bar{J}_t^T \bar{\theta}_t & \bar{J}_t^T \bar{\theta}_t + \bar{J}_t^T B_t^{-1} \bar{J}_t \end{bmatrix}^{-1} \quad (3.65)$$

$$= \frac{1}{\det(C^{-1} + VA^{-1}U)} \begin{bmatrix} \bar{J}_t^T \bar{\theta}_t + \bar{J}_t^T B_t^{-1} \bar{J}_t & -\bar{\theta}_t^T \bar{J}_t \\ -\bar{J}_t^T \bar{\theta}_t & 0 \end{bmatrix} \quad (3.66)$$

$$= -\frac{1}{\bar{\theta}_t^T \bar{J}_t \bar{J}_t^T \bar{\theta}_t} \begin{bmatrix} \bar{J}_t^T \bar{\theta}_t + \bar{J}_t^T B_t^{-1} \bar{J}_t & -\bar{\theta}_t^T \bar{J}_t \\ -\bar{J}_t^T \bar{\theta}_t & 0 \end{bmatrix} \quad (3.67)$$

$$= \begin{bmatrix} -\frac{\bar{J}_t^T \bar{\theta}_t + \bar{J}_t^T B_t^{-1} \bar{J}_t}{\bar{\theta}_t^T \bar{J}_t \bar{J}_t^T \bar{\theta}_t} & \frac{\bar{\theta}_t^T \bar{J}_t}{\bar{\theta}_t^T \bar{J}_t \bar{J}_t^T \bar{\theta}_t} \\ \frac{\bar{J}_t^T \bar{\theta}_t}{\bar{\theta}_t^T \bar{J}_t \bar{J}_t^T \bar{\theta}_t} & 0 \end{bmatrix} \quad (3.68)$$

Finally, we will put all the parts together and calculate B_{t+1}^{-1} :

$$B_{t+1}^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1} \quad (3.69)$$

$$= B_t^{-1} - [\bar{\theta}_t, B_t^{-1} \bar{J}_t] \begin{bmatrix} -\frac{\bar{J}_t^T \bar{\theta}_t + \bar{J}_t^T B_t^{-1} \bar{J}_t}{\bar{\theta}_t^T \bar{J}_t \bar{J}_t^T \bar{\theta}_t} & \frac{\bar{\theta}_t^T \bar{J}_t}{\bar{\theta}_t^T \bar{J}_t \bar{J}_t^T \bar{\theta}_t} \\ \frac{\bar{J}_t^T \bar{\theta}_t}{\bar{\theta}_t^T \bar{J}_t \bar{J}_t^T \bar{\theta}_t} & 0 \end{bmatrix} \begin{bmatrix} \bar{\theta}_t^T \\ \bar{J}_t^T B_t^{-1} \end{bmatrix} \quad (3.70)$$

$$= B_t^{-1} - \left[-\frac{\bar{\theta}_t(\bar{J}_t^T \bar{\theta}_t + \bar{J}_t^T B_t^{-1} \bar{J}_t)}{\bar{\theta}_t^T \bar{J}_t \bar{J}_t^T \bar{\theta}_t} + \frac{B_t^{-1} \bar{J}_t \bar{J}_t^T \bar{\theta}_t}{\bar{\theta}_t^T \bar{J}_t \bar{J}_t^T \bar{\theta}_t}, \frac{\bar{\theta}_t \bar{\theta}_t^T \bar{J}_t}{\bar{\theta}_t^T \bar{J}_t \bar{J}_t^T \bar{\theta}_t} \right] \begin{bmatrix} \bar{\theta}_t^T \\ \bar{J}_t^T B_t^{-1} \end{bmatrix} \quad (3.71)$$

$$= B_t^{-1} - \left(-\frac{\bar{\theta}_t(\bar{J}_t^T \bar{\theta}_t + \bar{J}_t^T B_t^{-1} \bar{J}_t)}{\bar{\theta}_t^T \bar{J}_t \bar{J}_t^T \bar{\theta}_t} + \frac{B_t^{-1} \bar{J}_t \bar{J}_t^T \bar{\theta}_t}{\bar{\theta}_t^T \bar{J}_t \bar{J}_t^T \bar{\theta}_t} \right) \bar{\theta}_t^T - \frac{\bar{\theta}_t \bar{\theta}_t^T \bar{J}_t \bar{J}_t^T B_t^{-1}}{\bar{\theta}_t^T \bar{J}_t \bar{J}_t^T \bar{\theta}_t} \quad (3.72)$$

$$= B_t^{-1} + \frac{\bar{\theta}_t \bar{J}_t^T \bar{\theta}_t \bar{\theta}_t^T}{\bar{\theta}_t^T \bar{J}_t \bar{J}_t^T \bar{\theta}_t} + \frac{\bar{\theta}_t \bar{J}_t^T B_t^{-1} \bar{J}_t \bar{\theta}_t^T}{\bar{\theta}_t^T \bar{J}_t \bar{J}_t^T \bar{\theta}_t} - \frac{B_t^{-1} \bar{J}_t \bar{J}_t^T \bar{\theta}_t \bar{\theta}_t^T}{\bar{\theta}_t^T \bar{J}_t \bar{J}_t^T \bar{\theta}_t} - \frac{\bar{\theta}_t \bar{\theta}_t^T \bar{J}_t \bar{J}_t^T B_t^{-1}}{\bar{\theta}_t^T \bar{J}_t \bar{J}_t^T \bar{\theta}_t} \quad (3.73)$$

We can reorganize (3.73) to make it easier to calculate and make the procedure faster. We can note that $\bar{\theta}_t^T \bar{J}_t$ is a scalar and therefore $\bar{\theta}_t^T \bar{J}_t \bar{J}_t^T \bar{\theta}_t = \bar{\theta}_t^T \bar{J}_t (\bar{\theta}_t^T \bar{J}_t)^T = (\bar{\theta}_t^T \bar{J}_t)^2$.

Therefore:

$$B_t^{-1} = I_m B_t^{-1} I_m \quad (3.74)$$

$$\frac{\bar{\theta}_t \bar{J}_t^T \bar{\theta}_t \bar{\theta}_t^T}{\bar{\theta}_t^T \bar{J}_t \bar{J}_t^T \bar{\theta}_t} = \frac{\bar{\theta}_t \bar{J}_t^T \bar{\theta}_t \bar{\theta}_t^T}{(\bar{\theta}_t^T \bar{J}_t)^2} = \frac{\bar{\theta}_t \bar{\theta}_t^T}{\bar{\theta}_t^T \bar{J}_t} \quad (3.75)$$

$$\frac{\bar{\theta}_t \bar{J}_t^T B_t^{-1} \bar{J}_t \bar{\theta}_t^T}{\bar{\theta}_t^T \bar{J}_t \bar{J}_t^T \bar{\theta}_t} = \frac{\bar{\theta}_t \bar{J}_t^T}{\bar{\theta}_t^T \bar{J}_t} B_t^{-1} \frac{\bar{J}_t \bar{\theta}_t^T}{\bar{\theta}_t^T \bar{J}_t} \quad (3.76)$$

$$\frac{B_t^{-1} \bar{J}_t \bar{J}_t^T \bar{\theta}_t \bar{\theta}_t^T}{\bar{\theta}_t^T \bar{J}_t \bar{J}_t^T \bar{\theta}_t} = \frac{B_t^{-1} \bar{J}_t \bar{J}_t^T \bar{\theta}_t \bar{\theta}_t^T}{(\bar{\theta}_t^T \bar{J}_t)^2} = \frac{B_t^{-1} \bar{J}_t \bar{\theta}_t^T}{\bar{\theta}_t^T \bar{J}_t} = I_m B_t^{-1} \frac{\bar{J}_t \bar{\theta}_t^T}{\bar{\theta}_t^T \bar{J}_t} \quad (3.77)$$

$$\frac{\bar{\theta}_t \bar{\theta}_t^T \bar{J}_t \bar{J}_t^T B_t^{-1}}{\bar{\theta}_t^T \bar{J}_t \bar{J}_t^T \bar{\theta}_t} = \frac{\bar{\theta}_t \bar{\theta}_t^T \bar{J}_t \bar{J}_t^T B_t^{-1}}{(\bar{\theta}_t^T \bar{J}_t)^2} = \frac{\bar{\theta}_t \bar{J}_t^T B_t^{-1}}{\bar{\theta}_t^T \bar{J}_t} = \frac{\bar{\theta}_t \bar{J}_t^T}{\bar{\theta}_t^T \bar{J}_t} B_t^{-1} I_m \quad (3.78)$$

Finally, using (3.74), (3.75), (3.76), (3.77), (3.78), we can put B_t^{-1} in evidence and thus arrive at the final iterative process for B_{t+1}^{-1} .

$$B_{t+1}^{-1} = \left[I_m - \frac{\bar{\theta}_t \bar{J}_t^T}{\bar{\theta}_t^T \bar{J}_t} \right] B_t^{-1} \left[I_m - \frac{\bar{J}_t \bar{\theta}_t^T}{\bar{\theta}_t^T \bar{J}_t} \right] + \frac{\bar{\theta}_t \bar{\theta}_t^T}{\bar{\theta}_t^T \bar{J}_t} \quad (3.79)$$

In this way, we can calculate B_{t+1}^{-1} from B_t^{-1} and the gradient information and estimates of the last two iterations. A fundamental observation to be made is its great computational advantage over

Newton's method, we calculate B_t^{-1} without ever calculate B_t . Consequently, this makes BFGS much faster over the iterations.

Thus, after all the mathematical procedures for constructing the algorithm, we now have how to write the pseudo-code for the BFGS implementation, given below:

Algorithm 3 The BFGS algorithm

Input: Initial guess θ_1 , a nonsingular B_1 (often the choice is $B_1 = I_m$), $\nabla J(\cdot)$ and a termination tolerance $\varepsilon > 0$.

$t = 1$

while $\|\nabla J(\theta_t)\| > \varepsilon$ **do**

1. Perform a backtracking line search to find η_t .
2. Calculate $\nabla J(\theta_t)$.
3. Estimate θ_{t+1} iteratively through $\theta_{t+1} = \theta_t - \eta_t B_t^{-1} \nabla J(\theta_t)$
4. Compute $B_{t+1}^{-1} = \left[I_m - \frac{\bar{\theta}_t \bar{J}_t^T}{\bar{\theta}_t^T \bar{J}_t} \right] B_t^{-1} \left[I_m - \frac{\bar{J}_t \bar{\theta}_t^T}{\bar{\theta}_t^T \bar{J}_t} \right] + \frac{\bar{\theta}_t \bar{\theta}_t^T}{\bar{\theta}_t^T \bar{J}_t}$
5. $t = t + 1$

end while

To illustrate, we will use the BFGS algorithm to estimate the parameters of the model developed for the following simulated data:

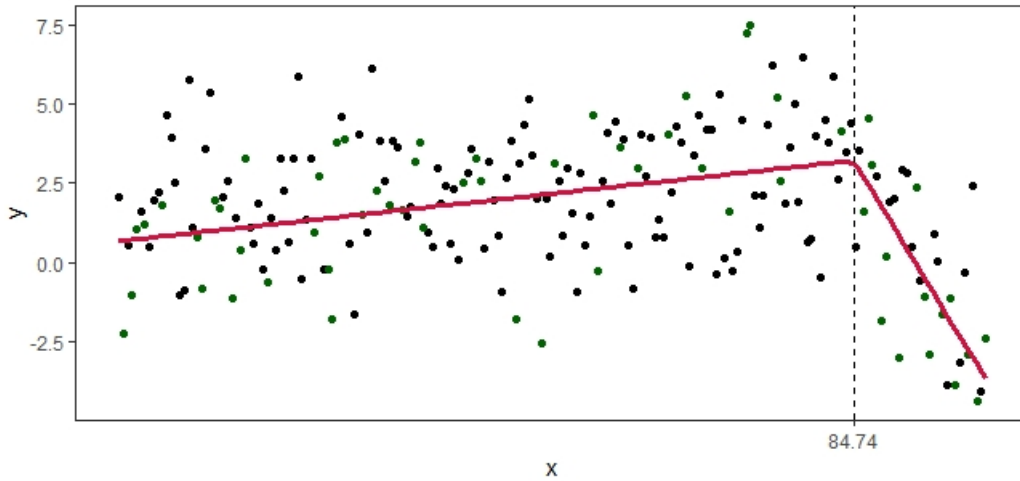


Figure 3.6: *Generated curve example*

The real model has the following functional form:

$$Y_i = 0.67 + 0.03x_i - 0.48(x_i - 84.74) + \epsilon_i \quad (3.80)$$

The BFGS is an iterative algorithm, estimates are expected to improve as the iterations progress, and this should be true mainly for knot estimation, which is mainly a change from the original model. In this specific case, 150 iterations were specified. Let us look carefully at iterations 1, 50, 120, and 150(last iteration).

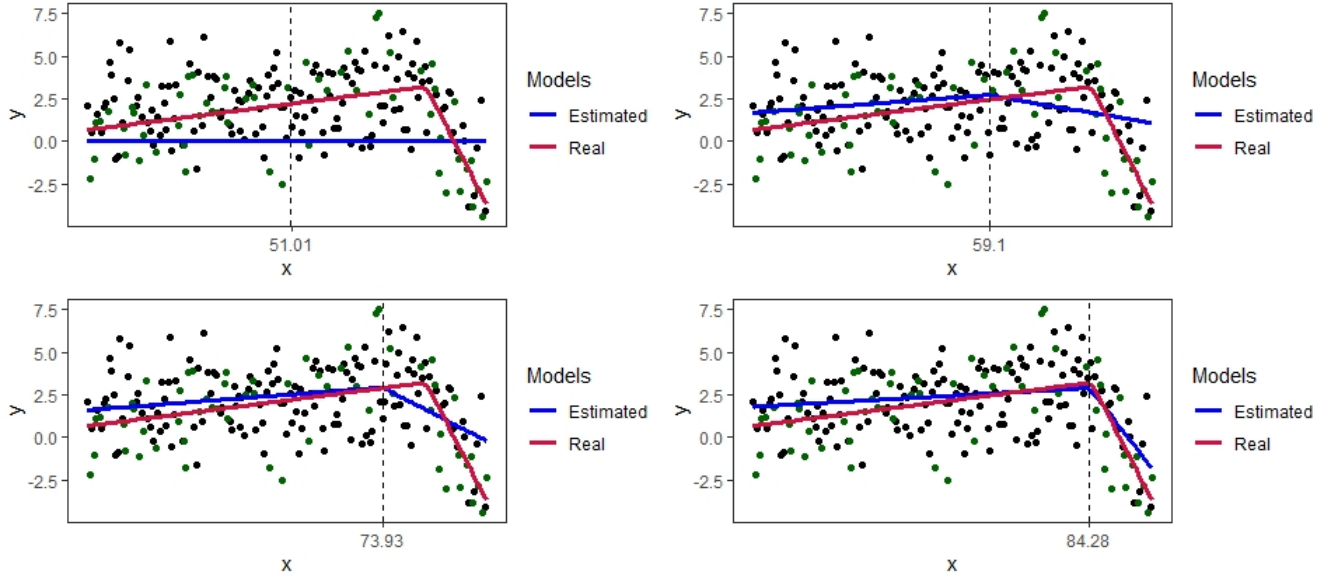


Figure 3.7: *Estimated curves at four iterations*

In Figure 3.7 we can see that the estimated curve is close to the curve that generated the data along with the iterations, and consequently, we can infer that the estimates have also improved. On the upper left side, the curve estimated in the first iteration is well away from the actual curve. On the upper right side, the estimated curve for iteration 50 has improved a little, but it still has problems estimating the knot. On the lower left side, the curve estimated at iteration 120 has improved considerably, with the estimated knot close to the actual knot. Finally, on the lower right side, the last iteration presents the curve estimated adequately, with the knot estimate very close to the real knot. The final knot estimate was 84.28, while the knot that generated the data is 84.77, really very close. This example showed in more detail that BFGS does what it should do with the development model, improve knot estimates over the iterations.

We will see the knot estimates over all iterations:

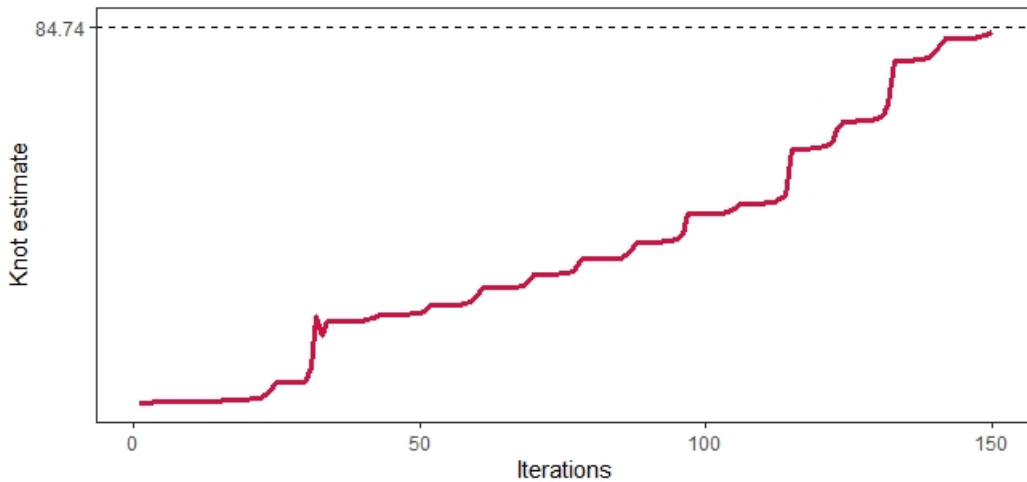


Figure 3.8: *Knot convergence over the iterations*

We can see that the estimates have improved towards the real knot 84.74. Now we will see that the cost function behaved throughout the iterations:

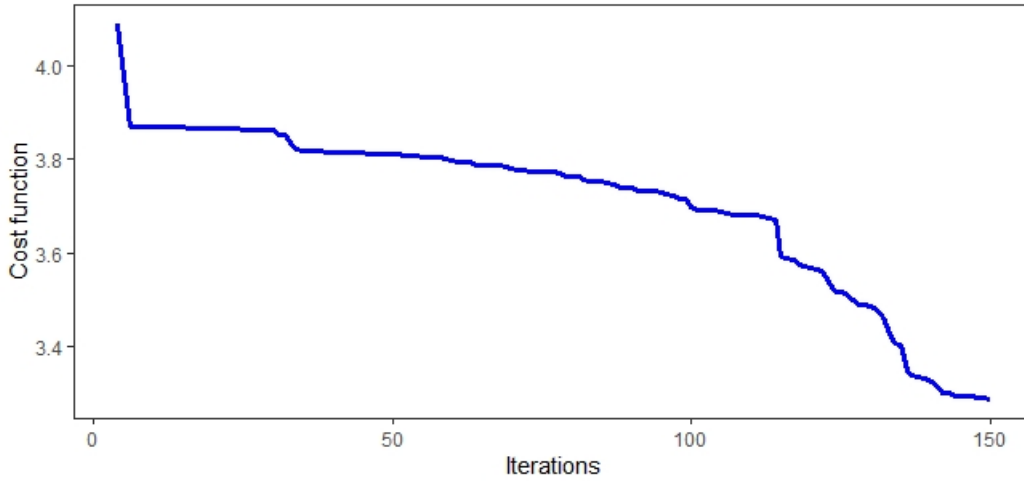


Figure 3.9: *Cost function over the iterations*

As expected, a cost function decreased over the iterations. In this way, parameter estimates were improved to predict observations better.

3.5 Estimation of Number of Knots

The parameters estimate β , and the knots t depend on the number of knots of each variable, which are parameters as well. Therefore, the number of knots are estimated before estimating all β and t . For this estimation, we will look carefully at how the cost functions behave as the number of knots in a unit increases. More specifically, let's do a detailed evaluation on $\lambda = 0.1, 0.3, 1, 3$ and $\alpha = 0, 1, 2, 3, 4, 5$ in 6 different models. This analysis aims to verify how the cost functions behave on average when the number of knots tested is around the real number of knots that generated the data.

The explained analysis is given below:

1. We generate 100 samples of the same size with a λ and a number of knots α using the two cost functions.
2. We performed the above step for $\lambda = 0.1, 0.3, 1, 3$ and $\alpha = 0, 1, 2, 3, 4, 5, 6$.
3. Now, we take the mean of the cost functions values from these 100 samples for each combination of λ and α .
4. We performed this entire process for six different models that generated the data.

Let us look at how the cost functions behave for each of the six models. The first three models are with $K = 1$, and the last three are with $K = 3$.

The first model is with one knot and $K=1$:

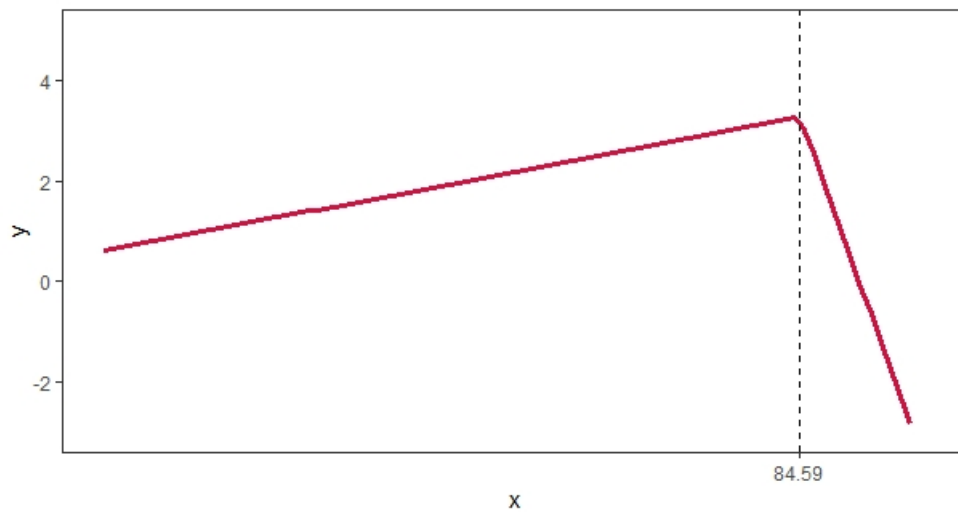


Figure 3.10: *Generated curve with one knot and $K=1$*

The average cost functions for this model are given below:

$\lambda \alpha$	0	1	2	3	4	5
0.1	5.756	4.045	4.086	4.129	4.171	4.211
0.3	5.756	4.132	4.256	4.364	4.463	4.559
1	5.756	4.396	4.720	4.935	5.128	5.292
3	5.756	4.941	5.403	5.664	5.746	5.757

Table 3.1: *Average cost functions for $J_1(\theta)$*

$\lambda \alpha$	0	1	2	3	4	5
0.1	5.756	4.015	4.025	4.035	4.045	4.052
0.3	5.756	4.045	4.085	4.125	4.164	4.202
1	5.756	4.150	4.295	4.439	4.583	4.726
3	5.756	4.449	4.893	5.337	5.780	6.222

Table 3.2: *Average cost functions for $J_2(\theta)$*

We can notice that the two cost functions obtained their minimum value for all λ values when $\alpha = 1$. Therefore, this suggests that the estimate of α is one, and models with $\alpha > 1$ need not to be performed based on the cost function value.

The second model is with two knots and $K=1$:

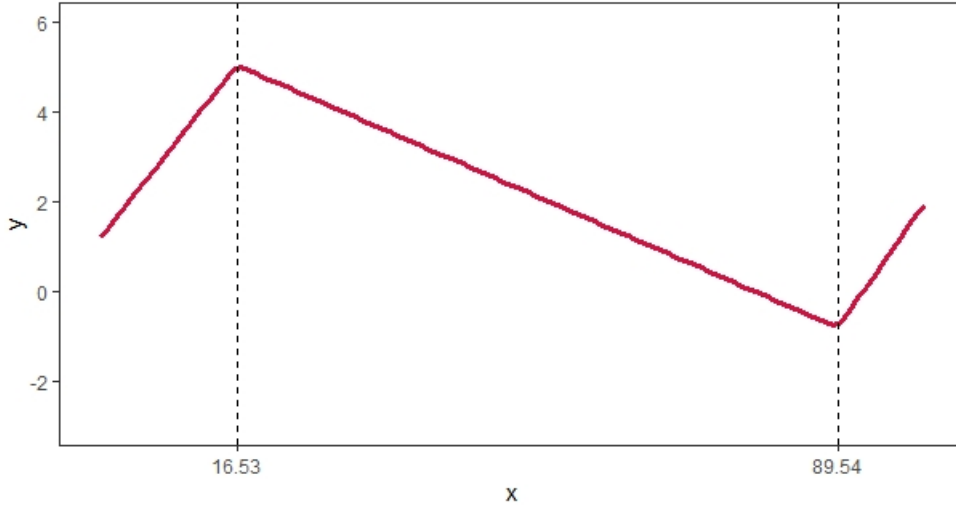


Figure 3.11: *Generated curve with two knots and $K=1$*

The average cost functions for this model are given below:

$\lambda \alpha$	0	1	2	3	4	5
0.1	4.959	4.366	4.132	4.168	4.220	4.258
0.3	4.959	4.420	4.314	4.428	4.508	4.591
1	4.959	4.584	4.694	4.814	4.892	4.931
3	4.959	4.899	4.951	4.959	4.959	4.959

Table 3.3: *Average cost functions for $J_1(\theta)$*

$\lambda \alpha$	0	1	2	3	4	5
0.1	4.959	4.363	4.049	4.033	4.042	4.049
0.3	4.959	4.393	4.109	4.123	4.161	4.199
1	4.959	4.497	4.318	4.437	4.580	4.722
3	4.959	4.797	4.916	5.334	5.778	6.219

Table 3.4: *Average cost functions for $J_2(\theta)$*

For $J_1(\theta)$ the models estimated with $\lambda = 0.1, 0.3$ obtained the smallest average cost function when the number of knots is equal to 2, and for $\lambda = 1, 3$ the model penalized more than it should, thus having the smallest cost function for a $\alpha = 1$. For $J_2(\theta)$ the models that obtained the smallest average cost function when $\alpha = 2$ was with $\lambda = 0.3, 1$. Therefore, this suggests for the two cost functions two values of λ adequately estimated the α that generated the data, so we should not estimate models for $\alpha > 2$ to avoid computational cost.

The third model is with three knots and $K=1$:

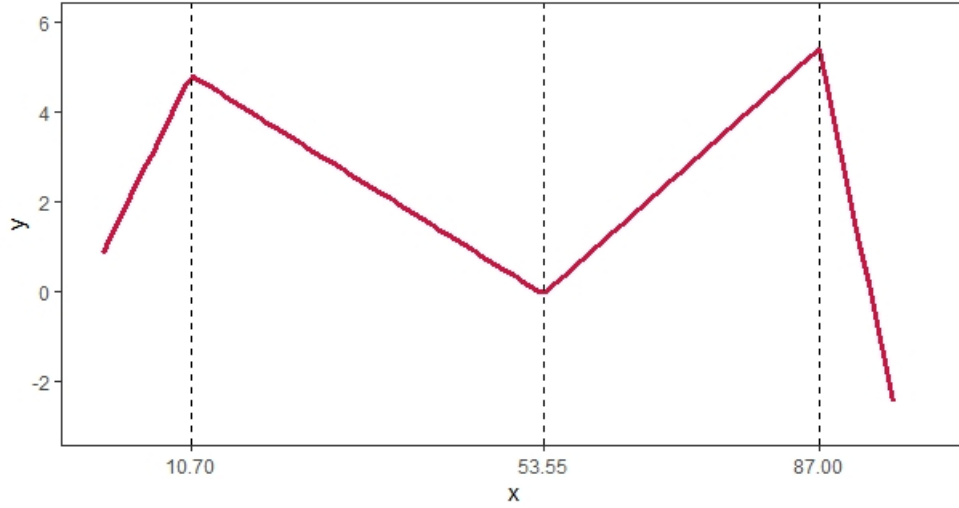


Figure 3.12: *Generated curve with three knots and $K=1$*

The cost functions means for this model are given below:

$\lambda \alpha$	0	1	2	3	4	5
0.1	7.767	7.665	4.748	4.487	4.642	4.794
0.3	7.767	7.674	5.202	5.329	5.681	5.988
1	7.767	7.701	6.749	7.046	7.458	7.665
3	7.767	7.753	7.767	7.767	7.767	7.767

Table 3.5: *Average cost functions for $J_1(\theta)$*

$\lambda \alpha$	0	1	2	3	4	5
0.1	7.767	7.676	4.531	4.026	4.036	4.044
0.3	7.767	7.706	4.591	4.116	4.156	4.193
1	7.767	7.810	4.801	4.430	4.575	4.717
3	7.767	8.110	5.399	5.328	5.772	6.213

Table 3.6: *Average cost functions for $J_2(\theta)$*

For $J_1(\theta)$ the only estimated model that obtained the most minor average cost function in the $\alpha = 3$ was with $\lambda = 0.1$, the smallest value of lambda, suggesting smaller lambdas should be tested. For $J_2(\theta)$ all λ values obtained the smallest average cost function, suggesting in this case the cost function is better than $J_1(\theta)$ for estimating the number of knots.

In the three models seen, for some λ , at least one λ values for the two cost functions matches the number of knots that generated the data, which is very good as it suggests an estimation process for the number of knots that is faster instead of simply testing with various values of α with fixed λ and checking which one fits best. Therefore, this suggests we can develop some estimation with the forward structure. We check the cost function with the increase in one unit in the number of knots, and from the moment that the cost function increases, we must stop, and the number of knots will be estimated where it obtained the smallest cost function value.

The fourth model is with three knots and $K=3$:

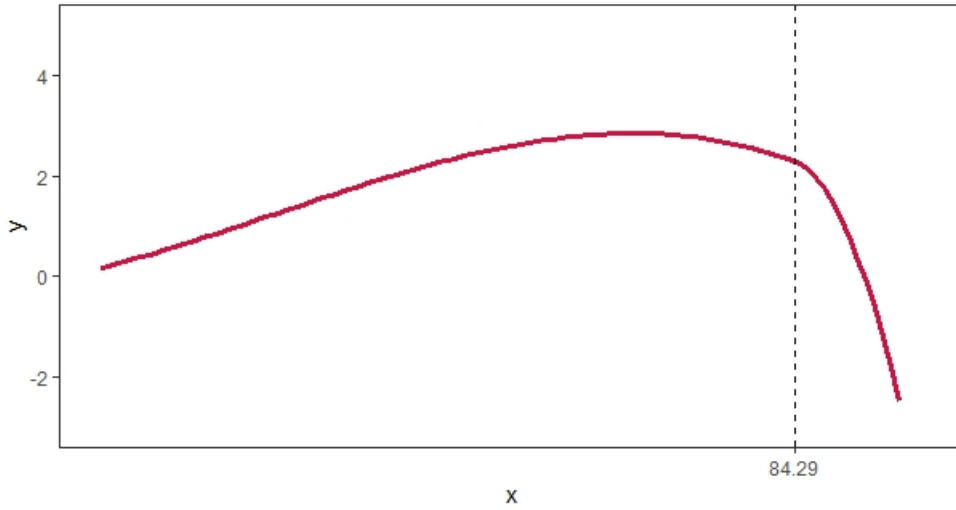


Figure 3.13: *Generated curve with one knot and $K=3$*

The average cost functions for this model are given below:

$\lambda \alpha$	0	1	2	3	4	5
0.1	4.152	4.013	3.989	3.986	3.981	3.976
0.3	4.152	4.014	3.989	3.986	3.981	3.978
1	4.152	4.013	3.990	3.987	3.984	3.983
3	4.152	4.013	3.991	3.990	3.989	3.990

Table 3.7: *Average cost functions for $J_1(\theta)$*

$\lambda \alpha$	0	1	2	3	4	5
0.1	4.152	4.029	4.019	4.031	4.041	4.049
0.3	4.152	4.059	4.079	4.121	4.161	4.199
1	4.152	4.163	4.288	4.435	4.580	4.722
3	4.152	4.463	4.887	5.333	5.776	6.219

Table 3.8: *Average cost functions for $J_2(\theta)$*

For $J_1(\theta)$ no model obtained the smallest average cost function at $\alpha = 1$, with the average cost function always being in much larger values of λ . Therefore, values larger than λ tend to obtain better results in estimating α in this situation. For $J_2(\theta)$ the models with $\lambda = 0.3, 1$ obtained the smallest values of the average cost function at $\alpha = 1$. In this case, models with $\alpha > 1$ do not need to be estimated based on the cost function.

The fifth model is with two knots and $K=3$:

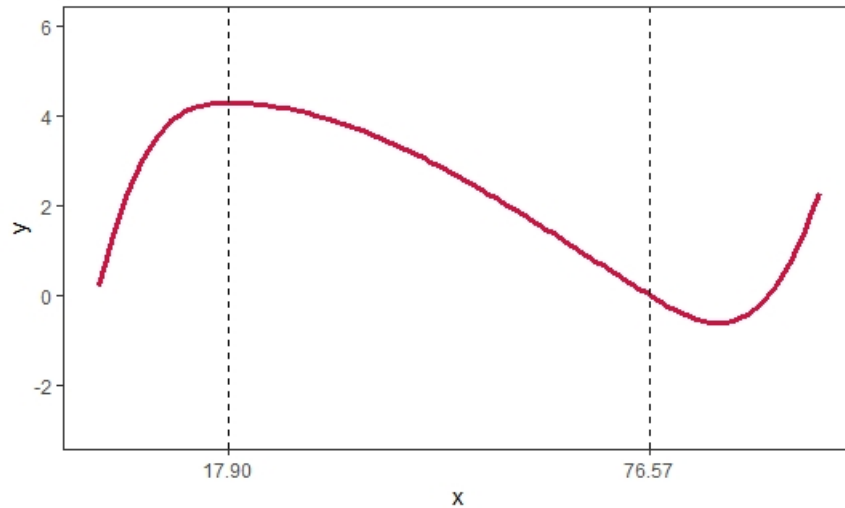


Figure 3.14: *Generated curve with two knots and $K=3$*

The cost functions means for this model are given below:

$\lambda \alpha$	0	1	2	3	4	5
0.1	4.095	4.081	3.992	3.985	3.981	3.976
0.3	4.095	4.079	3.992	3.985	3.982	3.979
1	4.095	4.080	3.993	3.987	3.985	3.984
3	4.095	4.081	3.995	3.991	3.992	3.993

Table 3.9: *Average cost functions for $J_1(\theta)$*

$\lambda \alpha$	0	1	2	3	4	5
0.1	4.095	4.095	4.022	4.030	4.041	4.050
0.3	4.095	4.125	4.082	4.120	4.161	4.200
1	4.095	4.230	4.291	4.434	4.579	4.723
3	4.095	4.529	4.890	5.331	5.776	6.220

Table 3.10: *Average cost functions for $J_2(\theta)$*

For $J_1(\theta)$ no model obtained the smallest mean cost function in $\alpha = 1$, the most suitable α was three with $\lambda = 3$, thus suggesting that larger values of λ should be tested. For $J_2(\theta)$, the average cost function with the smallest value at $\alpha = 2$ was with $\lambda = 0.1, 0.3$. Again, models with $\alpha > 2$ should not be tested, making choosing number of knots faster.

The sixth model is with three knots and $K=3$:

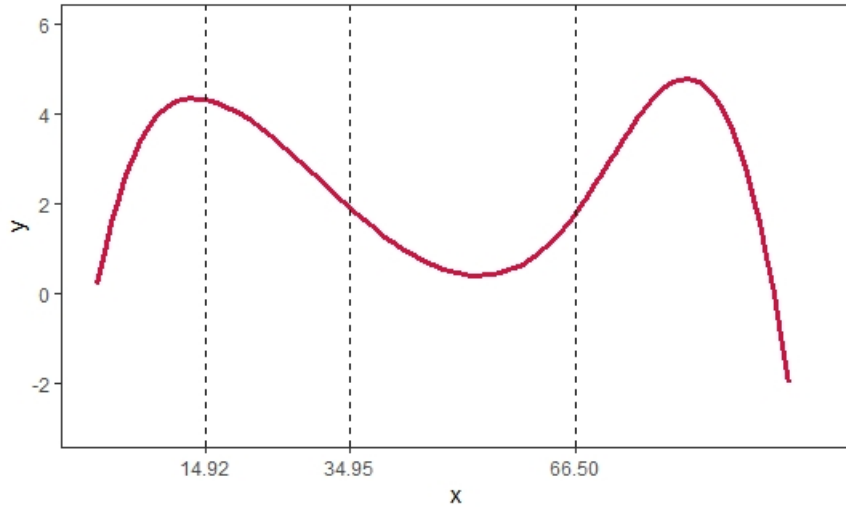


Figure 3.15: *Generated curve with three knots and $K=3$*

The average cost functions for this model are given below:

$\lambda \alpha$	0	1	2	3	4	5
0.1	7.395	4.068	3.994	3.983	3.979	3.976
0.3	7.395	4.068	3.994	3.983	3.980	3.979
1	7.395	4.066	3.995	3.986	3.987	3.986
3	7.395	4.067	3.999	3.994	3.999	4.003

Table 3.11: *Average cost functions for $J_1(\theta)$*

$\lambda \alpha$	0	1	2	3	4	5
0.1	7.395	4.082	4.024	4.026	4.038	4.049
0.3	7.395	4.112	4.084	4.116	4.158	4.199
1	7.395	4.220	4.293	4.430	4.578	4.723
3	7.395	4.521	4.892	5.328	5.775	6.219

Table 3.12: *Average cost functions for $J_2(\theta)$*

For $J_1(\theta)$ the models with $\lambda = 1, 3$ obtained the smallest values of the average cost function at $\alpha = 3$. For $J_2(\theta)$ the models with $\lambda = 0.1, 0.3$ obtained the smallest values of the average cost function at $\alpha = 2$, one knot less than the number of knots that generated the data.

In all the models presented, we can observe that in several situations for the two cost functions at some value of λ , the average cost function is close to the number of knots that generated the data. Therefore, it makes sense to think of an estimation method for number of knots that considers the cost function and the addition of knot number in a unit. Then check whether it is worth increasing the number of knots based on the value of the cost function.

The number of knots estimation used in this dissertation makes use of the forward process. Models are estimated with the addition of α in one unit until the cost function increases, at this point the number of knots is estimated.

The estimation method is given below:

Algorithm 4 number of knots estimation algorithm

Input: Training data X, y , α max L , features number p and cost function $J(\theta)$.

```

for  $j = 1, \dots, p$  do
   $J = []$ 
  for  $m = 0, 1, 2, \dots, L$  do
     $\alpha_j = m$ 
    Fit  $Y$  with  $X_j$  with  $\hat{\alpha}_j = \alpha_j$  using  $J(\theta)$ 
     $J[m + 1] = J(\hat{\theta})$ 
    if  $J(\hat{\theta}) > \min(J)$  then
       $\hat{\alpha}_j = \alpha_j - 1$ 
      stop the inner loop
    end if
  end for
end for

```

3.6 Summary of the parameter estimation procedure

Now, we can estimate all the parameters of the proposed models. The summary for estimating all parameters is given below:

1. The hyperparameter vectors K and λ must be fixed to estimate the parameters.
2. All α_j are estimated by Algorithm 4.
3. With all knot numbers estimated, we now estimate all parameters of our model with the BFGS method given by Algorithm 3.

The hyperparameters of the models are chosen through the two most common methods: grid search or random search. The grid search method is more used when we have a few hyperparameters. In our case, the random search method is more commonly used because it is faster since we will probably have many hyperparameters and generally obtain good results.

3.7 Loss functions comparasion

A big question for this dissertation is to find out in which situations the two developed models given in (3.1) and (3.2) are better, either by adequately estimating the number of knots or by predicting the observations well and also analyzing as the hyperparameters λ are chosen in each situation, provide an interpretation and suggest possible reasonable λ values for future uses of the proposed models. A model with a cost function that predicts observations well but does not estimate number of knotss well is not attractive, as well as a model that predicts observations well but does not estimate knots well is not attractive either. So, we must check how each cost function behaves in different situations to know when to expect better results.

For this purpose, we tested the two proposed models in different situations of sample size, λ values, number of knots, and models that generated the data. More specifically, we generate six different models varying the knots from 1 to 3 and K equal to 1 or 3, which are the models from the previous section given in (3.10), (3.11), (3.12), (3.13), (3.14), (3.15). For each model, we test different scenarios:

- $n = 300, 900, 1500, 3000$
- 100 samples for each sample size

- $\lambda = 0, 0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30$
- 5 knots as a maximum number for estimating the number of knots.

At this point, in all situations, the number of knots will be estimated according to the method given in Algorithm 4. For each situation, we carry out the following procedure:

- We split the sample into 70% for train and 30% for test.
- For each λ , we estimate the knot number.
- The λ and $\hat{\alpha}$ chosen will be the one to obtain the most minor mean square error through a 5-fold cross-validation.
- We estimate the model in the training sample with final λ and α , then predict the test set observations and check some metrics like mean square error and standard deviation.
- We performed the entire process above for 100 different samples and checked the proportion of final $\hat{\alpha}$ and λ and the mean of the mean square error and standard deviation of the test set.

First, let us analyze how knot number estimates behave for the six models presented. We will present graphs of the proportion of the estimated α .

3.7.1 Knot number estimation and hyperparameter chosen

The first model, with one knot and $K=1$, given in 3.10:

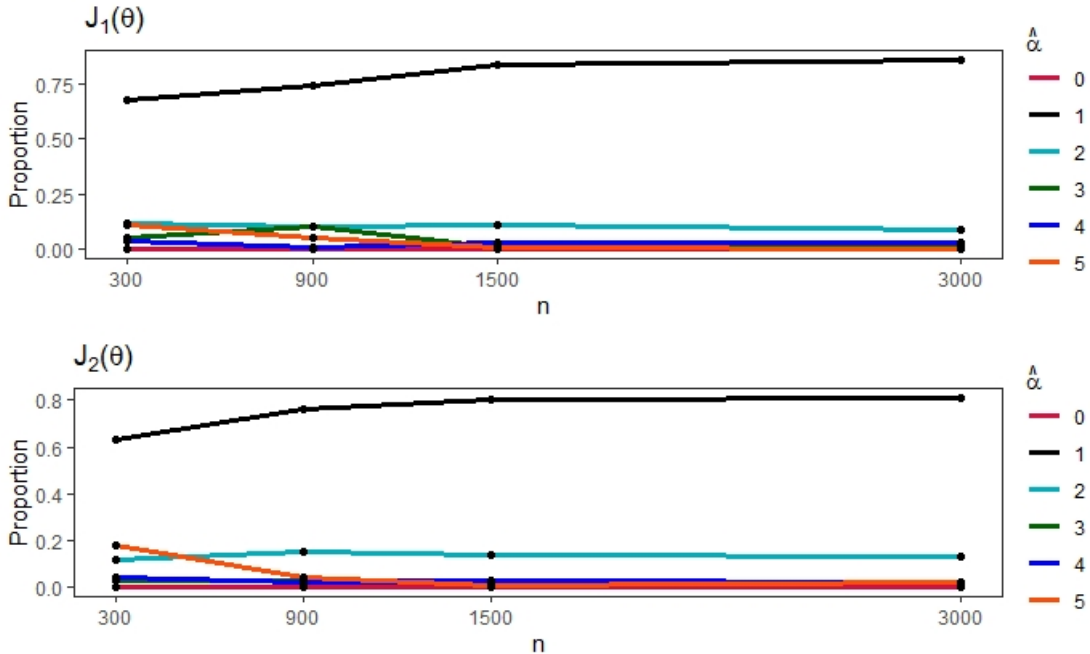


Figure 3.16: $\hat{\alpha}$ proportion comparison with one knot and $K = 1$

For the two cost functions, $\hat{\alpha} = 1$ mainly was chosen, and also the proportion increases as the sample size grows. Furthermore, even with the smaller sample size, it already estimates the knot number correctly, suggesting that the sample size is not a problem. The two models estimated the knot number well and there is no significant difference for this purpose.

$n \lambda$	0	0.01	0.03	0.1	0.3	1	3	10	30
300	0.13	0.22	0.23	0.20	0.21	0.01	0.00	0.00	0.00
900	0.10	0.30	0.26	0.21	0.13	0.00	0.00	0.00	0.00
1500	0.10	0.29	0.29	0.26	0.06	0.00	0.00	0.00	0.00
3000	0.06	0.41	0.29	0.22	0.02	0.00	0.00	0.00	0.00

Table 3.13: λ proportion chosen for $J_1(\theta)$

$n \lambda$	0	0.01	0.03	0.1	0.3	1	3	10	30
300	0.33	0.14	0.07	0.19	0.14	0.03	0.05	0.05	0.00
900	0.18	0.29	0.12	0.19	0.03	0.06	0.07	0.06	0.00
1500	0.15	0.33	0.11	0.13	0.01	0.04	0.03	0.20	0.00
3000	0.11	0.38	0.12	0.03	0.00	0.07	0.08	0.21	0.00

Table 3.14: λ proportion chosen for $J_2(\theta)$

In this case, the two cost functions performed better with smaller λ , with the most chosen being $\lambda = 0.01$. This makes sense because it is a simple functional form with just one knot and because the model estimated the knot number most of the time.

The second model, with two knots and $K=1$, given in 3.11:

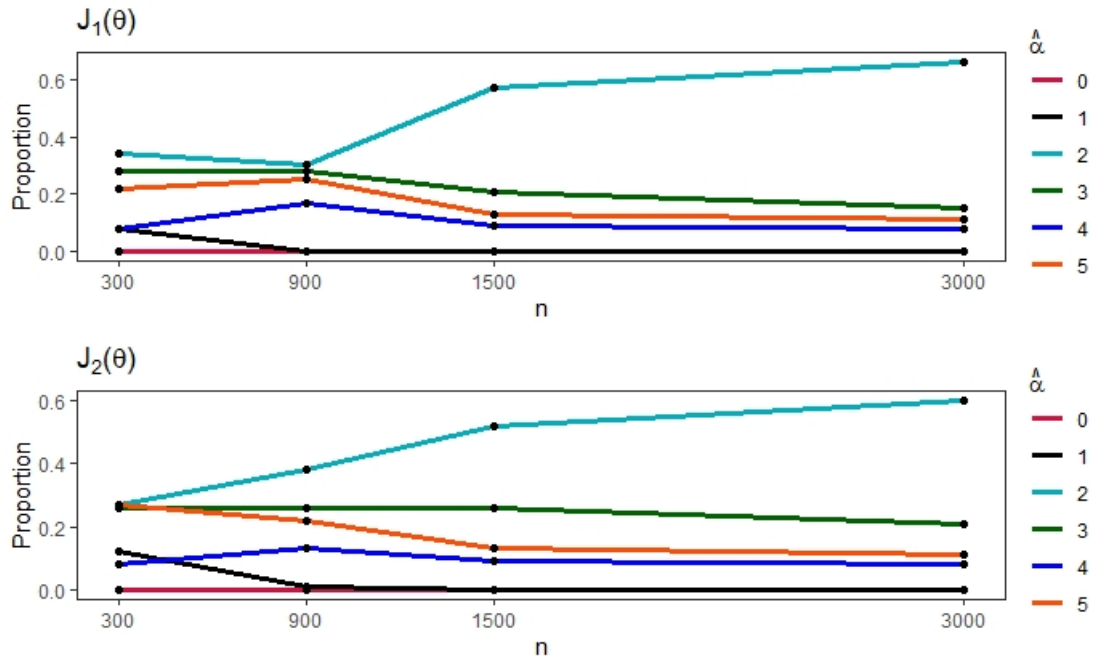


Figure 3.17: $\hat{\alpha}$ proportion comparison with two knots and $K = 1$

For the two cost functions, mostly $\hat{\alpha} = 2$ was chosen. However, with a sample size of 300, the models estimated another $\hat{\alpha} \neq 2$ in similar proportions. As the sample size increases, the models estimate the knot number better, suggesting that the larger the sample, the better the estimate. The two models had similar behavior, estimating the knot number better with the sample sizes of 1500 and 3000. Therefore, there is no significant difference between them.

$n \lambda$	0	0.01	0.03	0.1	0.3	1	3	10	30
300	0.14	0.29	0.23	0.18	0.16	0.00	0.00	0.00	0.00
900	0.31	0.35	0.24	0.10	0.00	0.00	0.00	0.00	0.00
1500	0.29	0.50	0.15	0.06	0.00	0.00	0.00	0.00	0.00
3000	0.25	0.53	0.21	0.01	0.00	0.00	0.00	0.00	0.00

Table 3.15: λ proportion chosen for $J_1(\theta)$

$n \lambda$	0	0.01	0.03	0.1	0.3	1	3	10	30
300	0.53	0.06	0.07	0.12	0.12	0.10	0.00	0.00	0.00
900	0.52	0.07	0.11	0.22	0.06	0.02	0.00	0.00	0.00
1500	0.47	0.28	0.09	0.11	0.03	0.02	0.00	0.00	0.00
3000	0.33	0.28	0.28	0.06	0.01	0.04	0.00	0.00	0.00

Table 3.16: λ proportion chosen for $J_2(\theta)$

In this case, the two cost functions performed better with smaller λ , with the most chosen being 0.01 or 0. As shown in Figure 3.17, the knot number was estimated correctly most of the time, suggesting that it is a more straightforward model to estimate.

The third model, with three knots and $K=1$, given in 3.12:

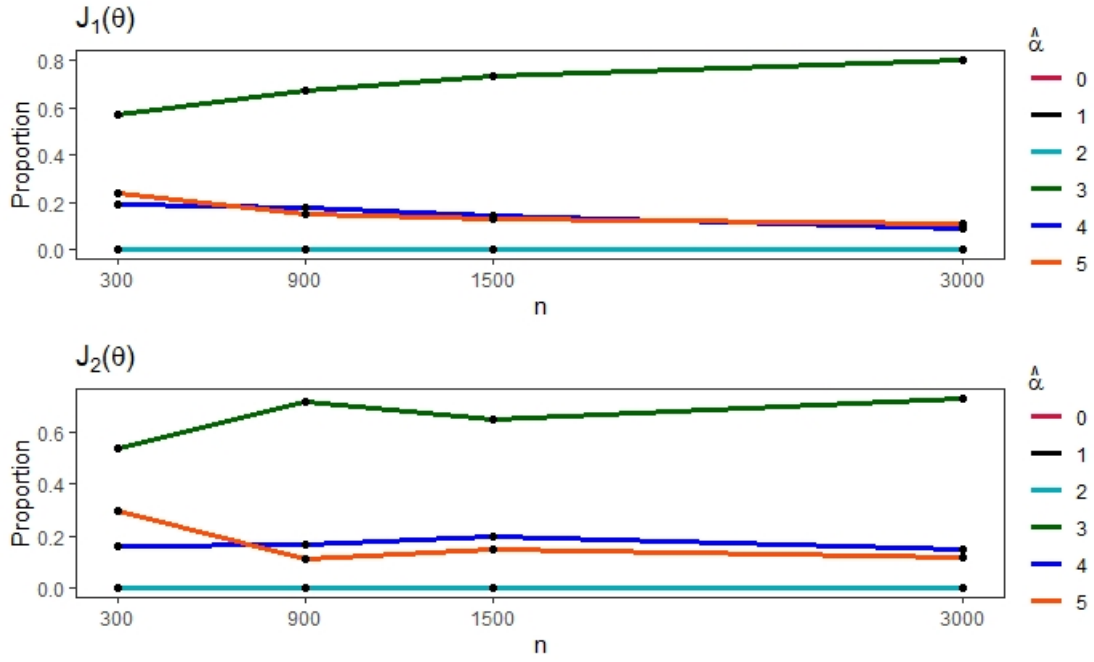


Figure 3.18: $\hat{\alpha}$ proportion comparison with three knots and $K = 1$

For the two cost functions, mainly $\hat{\alpha} = 3$ was chosen. The only estimated number of knotss were 3, 4, and 5. As the sample size increases, the models better estimate the number of knots, but it is not as important because even with the smallest sample size (300) both models estimate the correct knot number the vast majority of the time. The two models had very similar behavior, estimating the knot number well at all sample sizes, only a slight advantage for $J_1(\theta)$ with the sample size of 300. Therefore, there is no significant difference between them.

$n \lambda$	0	0.01	0.03	0.1	0.3	1	3	10	30
300	0.21	0.32	0.33	0.14	0.00	0.00	0.00	0.00	0.00
900	0.32	0.45	0.21	0.02	0.00	0.00	0.00	0.00	0.00
1500	0.33	0.41	0.26	0.00	0.00	0.00	0.00	0.00	0.00
3000	0.32	0.65	0.03	0.00	0.00	0.00	0.00	0.00	0.00

Table 3.17: λ proportion chosen for $J_1(\theta)$

$n \lambda$	0	0.01	0.03	0.1	0.3	1	3	10	30
300	0.55	0.04	0.02	0.14	0.16	0.09	0.00	0.00	0.00
900	0.40	0.17	0.14	0.15	0.06	0.08	0.00	0.00	0.00
1500	0.43	0.21	0.17	0.12	0.07	0.00	0.00	0.00	0.00
3000	0.40	0.38	0.16	0.02	0.04	0.00	0.00	0.00	0.00

Table 3.18: λ proportion chosen for $J_2(\theta)$

In both cost functions, smaller λ performed better. As shown in Figure 3.18, the model estimated the knot number most of the time and with this it may have helped to estimate the location of knots .

The fourth model, with one knot and $K=3$, given in 3.13:

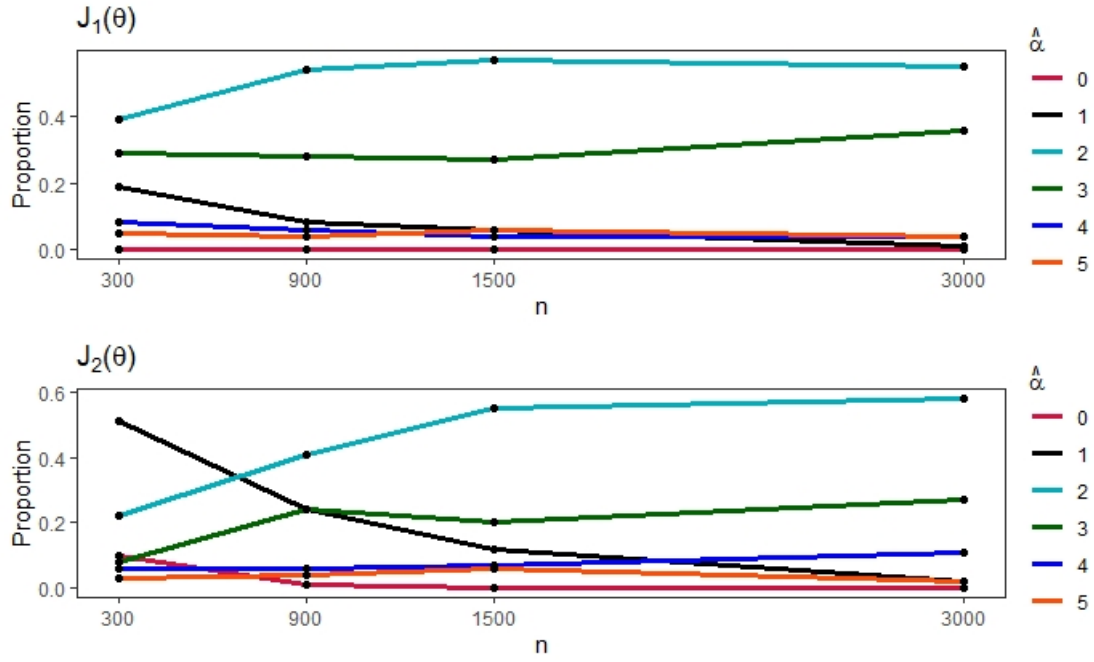


Figure 3.19: $\hat{\alpha}$ proportion comparison with one knot and $K = 3$

For both models, mostly $\hat{\alpha} = 2, 3$ were chosen, which are greater than the knot number that generated the data, which is not expected to happen in practice. One possible reason for this is λ values are not suitable for this case, only with the smallest sample size that $J_2(\theta)$ obtained the largest proportion at $\hat{\alpha} = 1$

$n \lambda$	0	0.01	0.03	0.1	0.3	1	3	10	30
300	0.04	0.01	0.04	0.03	0.05	0.04	0.15	0.25	0.39
900	0.08	0.03	0.02	0.02	0.07	0.07	0.07	0.23	0.41
1500	0.06	0.05	0.01	0.02	0.06	0.08	0.16	0.25	0.31
3000	0.06	0.01	0.02	0.07	0.03	0.17	0.21	0.15	0.28

Table 3.19: λ proportion chosen for $J_1(\theta)$

$n \lambda$	0	0.01	0.03	0.1	0.3	1	3	10	30
300	0.27	0.05	0.10	0.29	0.18	0.09	0.02	0.00	0.00
900	0.43	0.11	0.15	0.20	0.10	0.01	0.00	0.00	0.00
1500	0.47	0.11	0.17	0.21	0.02	0.02	0.00	0.00	0.00
3000	0.44	0.23	0.21	0.12	0.00	0.00	0.00	0.00	0.00

Table 3.20: λ proportion chosen for $J_2(\theta)$

The fifth model, with two knots and $K=3$, given in 3.14:

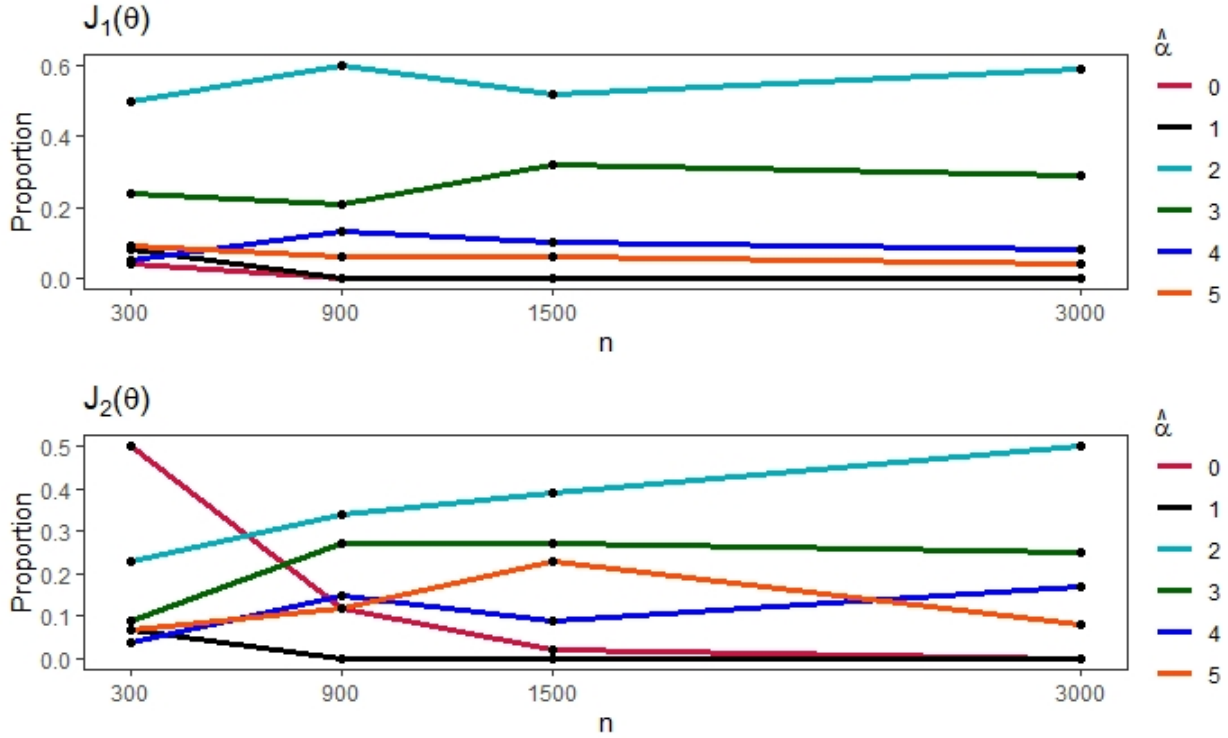


Figure 3.20: $\hat{\alpha}$ proportion comparison with two knots and $K = 3$

For the two cost functions, mainly $\hat{\alpha} = 2$ was chosen. For $J_2(\theta)$ the smallest sample size(300) obtained the highest proportion with no knots. This is probably due to insufficient sample size for the model complexity or α larger than it should be for this case. As the sample grows, the proportion of correctly estimated number knots increases. Therefore, the larger the sample, the better the knot number estimate. For smaller sample sizes, $J_1(\theta)$ is more suitable for estimating the knot number much better while there is not so much difference for larger sizes.

$n \lambda$	0	0.01	0.03	0.1	0.3	1	3	10	30
300	0.05	0.02	0.03	0.04	0.03	0.08	0.18	0.24	0.33
900	0.05	0.01	0.04	0.03	0.04	0.09	0.22	0.30	0.22
1500	0.07	0.04	0.04	0.06	0.13	0.16	0.26	0.19	0.05
3000	0.11	0.03	0.05	0.07	0.08	0.20	0.25	0.21	0.00

Table 3.21: λ proportion chosen for $J_1(\theta)$

$n \lambda$	0	0.01	0.03	0.1	0.3	1	3	10	30
300	0.23	0.24	0.15	0.26	0.10	0.02	0.00	0.00	0.00
900	0.64	0.15	0.08	0.13	0.00	0.00	0.00	0.00	0.00
1500	0.68	0.16	0.11	0.05	0.00	0.00	0.00	0.00	0.00
3000	0.82	0.07	0.10	0.01	0.00	0.00	0.00	0.00	0.00

Table 3.22: λ proportion chosen for $J_2(\theta)$

For $J_1(\theta)$, the biggest λ performed better. This may have been caused by the increase in flexibility caused by non-linearity. For $J_2(\theta)$, most of the time the smallest possible $\lambda(0)$ was chosen, only for the smallest sample size(300) that another λ was chosen, which may be the main cause of the knot number was not estimated correctly as shown in Figure 3.20.

The sixth model, with three knots and $K=3$, given in 3.15:

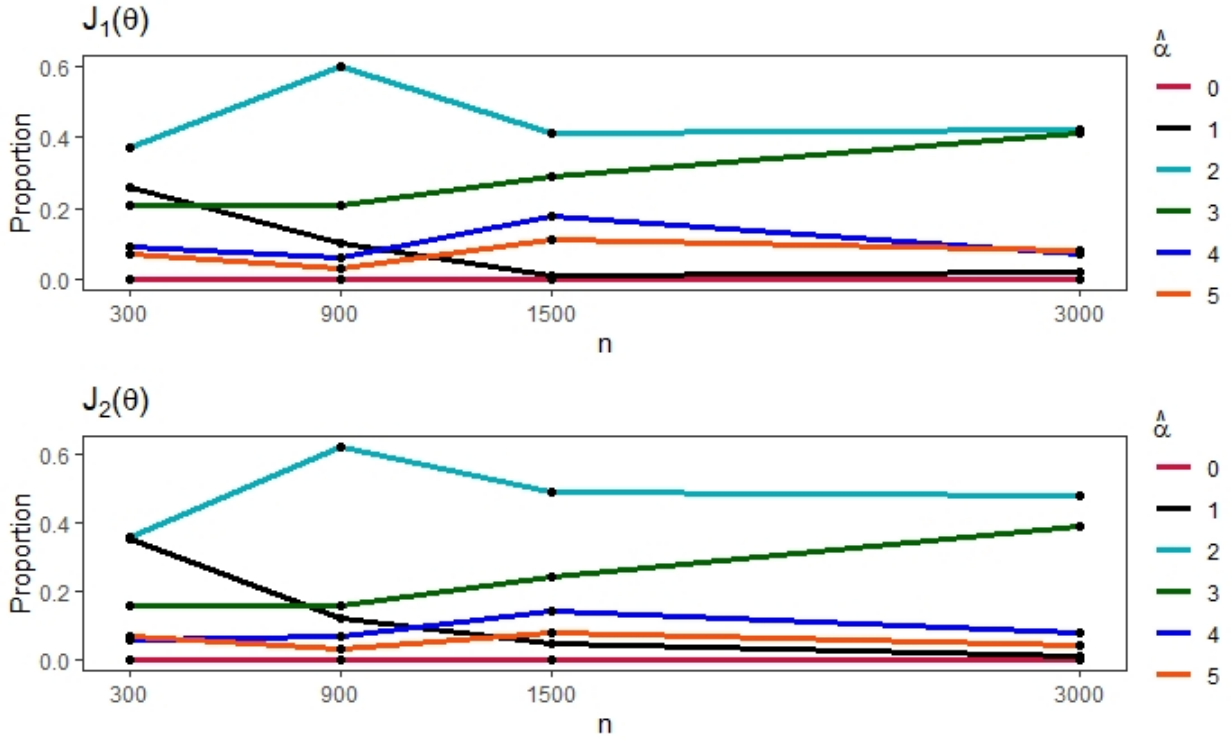


Figure 3.21: $\hat{\alpha}$ proportion comparison with three knots and $K = 3$

For the two cost functions, mainly $\hat{\alpha} = 2$ was chosen, one knot less than the knot number that generated the data, which is not a problem in this case because it is a non-linear fit. When $K \geq 2$, the number of knots is not a problem if it is smaller than the real knot number, since the added flexibility of non-linearity overcomes this problem. As the sample grows, the proportion of correctly estimated knot number increases but is still less than that of $\hat{\alpha} = 2$. The two models have very similar behavior, hence there is no superiority between them for this purpose.

$n \lambda$	0	0.01	0.03	0.1	0.3	1	3	10	30
300	0.04	0.06	0.04	0.04	0.06	0.10	0.15	0.25	0.26
900	0.08	0.04	0.07	0.03	0.08	0.12	0.16	0.25	0.17
1500	0.09	0.06	0.07	0.07	0.11	0.14	0.19	0.23	0.04
3000	0.10	0.09	0.04	0.04	0.04	0.16	0.25	0.21	0.07

Table 3.23: λ proportion chosen for $J_1(\theta)$

$n \lambda$	0	0.01	0.03	0.1	0.3	1	3	10	30
300	0.26	0.10	0.07	0.17	0.10	0.05	0.11	0.14	0.00
900	0.62	0.07	0.05	0.11	0.05	0.05	0.01	0.04	0.00
1500	0.34	0.11	0.16	0.30	0.05	0.02	0.00	0.02	0.00
3000	0.36	0.12	0.26	0.18	0.03	0.05	0.00	0.00	0.00

Table 3.24: λ proportion chosen for $J_2(\theta)$

For $J_1(\theta)$, the biggest λ performed better. This may have been caused by the increase in flexibility caused by non-linearity. For $J_2(\theta)$, every time the smallest possible $\lambda(0)$ has been chosen.

3.7.2 Predictive performance

Let us analyze how well the models predicted the observations by checking the mean of the mean square error and standard deviation. We will do this through a table with the mean square error and the standard deviation between parentheses. The smallest mean square error between the two cost functions will be bold. Also, in some situations, the errors are equal but have been rounded to three decimal places. In these cases, the error in bold is the smallest with all decimal places. Although the six generated models are different, they will have the same mean square error and standard deviation because their noise is the same.

For the first model, with one knot and $K=1$, given in 3.10:

Models n	300	900	1500	3000
Real	3.975(5.578)	4.021(5.691)	4.000(5.687)	4.007(5.667)
$J_1(\theta)$	4.107 (5.730)	4.066(5.751)	4.020(5.718)	4.015 (5.677)
$J_2(\theta)$	4.109(5.727)	4.064 (5.749)	4.020 (5.716)	4.016(5.678)

Table 3.25: Predictive performance comparison between cost functions

For the second model, with two knots and $K=1$, given in 3.11:

Models n	300	900	1500	3000
Real	3.975(5.578)	4.021(5.691)	4.000(5.687)	4.007(5.667)
$J_1(\theta)$	4.173 (5.860)	4.109(5.801)	4.032 (5.734)	4.023 (5.690)
$J_2(\theta)$	4.186(5.863)	4.108 (5.795)	4.033(5.731)	4.023(5.692)

Table 3.26: Predictive performance comparison between cost functions

For the third model, with three knots and $K=1$, given in 3.12:

Models n	300	900	1500	3000
Real	3.975(5.578)	4.021(5.691)	4.000(5.687)	4.007(5.667)
$J_1(\theta)$	4.168(5.845)	4.106(5.800)	4.034 (5.733)	4.024 (5.691)
$J_2(\theta)$	4.168 (5.847)	4.101 (5.796)	4.035(5.733)	4.026(5.693)

Table 3.27: Predictive performance comparison between cost functions

For the fourth model, with one knot and $K=3$, given in 3.13:

Models n	300	900	1500	3000
Real	3.975(5.578)	4.021(5.691)	4.000(5.687)	4.007(5.667)
$J_1(\theta)$	4.120 (5.766)	4.082 (5.778)	4.028 (5.728)	4.021(5.686)
$J_2(\theta)$	4.153(5.794)	4.098(5.803)	4.031(5.733)	4.021 (5.686)

Table 3.28: Predictive performance comparison between cost functions

For the fifth model, with two knots and $K=3$, given in 3.14:

Models n	300	900	1500	3000
Real	3.975(5.578)	4.021(5.691)	4.000(5.687)	4.007(5.667)
$J_1(\theta)$	4.136 (5.781)	4.076 (5.764)	4.028 (5.725)	4.021 (5.684)
$J_2(\theta)$	4.145(5.775)	4.093(5.817)	4.031(5.730)	4.021(5.684)

Table 3.29: Predictive performance comparison between cost functions

For the sixth model, with three knots and $K=3$, given in 3.15:

Models n	300	900	1500	3000
Real	3.975(5.578)	4.021(5.691)	4.000(5.687)	4.007(5.667)
$J_1(\theta)$	4.135 (5.790)	4.089(5.786)	4.033 (5.732)	4.023 (5.687)
$J_2(\theta)$	4.146(5.820)	4.087 (5.781)	4.035(5.732)	4.023(5.687)

Table 3.30: *Predictive performance comparison between cost functions*

In all six generated models, both cost functions obtained a predictive performance close to the model that generated the data, which was expected since the generated models have the same functional form as the tested models. The big question is to know when one model is better than another in some aspects. There was no significant difference between the two proposed models in predictive performance because in both, as the sample size increases, the difference between the real and estimated models stays closer, better predicting the observations. Furthermore, the models' performance is very close to that of all sample sizes, so in these cases, there is no superiority between them.

Chapter 4

Predictive Performance with p variables

One of the main objectives of this dissertation is to verify if the proposed models are applicable to make predictions in practice. To do this, we performed two types of analysis, one with simulations and the other with several real data. In the simulations, tests were carried out in different scenarios, with different sample sizes and noise variances. For performances on real data, we tested several models on six datasets with different sample sizes and number of features.

For both analyses, we carry out the following procedure:

- We split the sample into 70% for train and 30% for test.
- We performed the random search method with 30 iterations using 5-fold cross-validation to optimize the hyperparameters of the models in the training set.
- For each combination of hyperparameters, we estimate the knots number.
- All hyperparameters chosen will be the one to obtain the smallest mean square error of the cross-validation from the random search method.
- We estimate the model parameters in the training sample with final hyperparameters, then predict the test set observations and check the metrics like mean square error and standard deviation of the square error.

We compared the two proposed models with several predictive methods well known in the literature. The models chosen are of different functional forms, containing neural network structures, boosting, regression trees, regularization methods, and a model that uses the same knot structure. The model that uses the structure very similar to the proposed models and more known in the literature is the B-splines model.

The models tested were:

1. The proposed model with cost function $J_1(\theta)$;
2. The proposed model with cost function $J_2(\theta)$;
3. B-Splines;
4. Ridge Regression;
5. Random Forest;
6. Adaboost tree;
7. Neural Network Multilayer Perceptron.

The chosen hyperparameters and details for all models are given below:

The proposed model with cost function $J_1(\theta)$

1. $\lambda_j \in [0, 0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30], j = 1, \dots, p$
2. $K_j \in [1, 2, 3], j = 1, \dots, p$
3. $\alpha_j \max = 3, j = 1, \dots, p$

The proposed model with cost function $J_2(\theta)$

1. $\lambda \in [0, 0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]$
2. $K_j \in [1, 2, 3], j = 1, \dots, p$
3. $\alpha_j \max = 3, j = 1, \dots, p$

B-Splines

1. Polynomial degree of each variable $\in [1, 2, 3]$
2. Knot number of each variable $\in [0, 1, 2, 3]$

Ridge Regression

1. The regularization hyperparameter $\in (0, 10)$

Random Forest

1. Tree number $\in [50, 80, 100, 150]$
2. Max depth $\in [2, 3, 4]$
3. The minimum number of samples required to split an internal node $\in [10, 15, 20, 30]$
4. The minimum number of samples required to be at a leaf node $\in [5, 10, 15, 20, 25, 30, 50]$

Adaboost tree

1. Each regression tree has a max depth equals to 3.
2. Tree number $\in [50, 80, 100, 150]$
3. Learning rate $\in (0.9, 1.5)$

Neural Network Multilayer Perceptron

1. The backpropagation optimization method was LBFGS
2. Size and number of the hidden layers $\in [(10), (30), (50), (10, 10), (30, 30), (50, 50)]$
3. Activation function is identity or relu
4. Learning rate $\in [0.01, 0.03, 0.1, 0.3]$

The results are presented with the mean square error and standard deviation of the square error in parentheses from the test set. The model that obtained the best performance will be marked in bold, and when the proposed models are not the best but have competitive results, they will be marked in blue.

4.1 Simulations

In the simulations, the tests were always performed with the same functional form, given below:

$$Y_i = 1.27 + f_1(x_{i1}) + f_2(x_{i2}) + f_3(x_{i3}) + \epsilon_i, \epsilon_i \sim \text{Normal}(0, \sigma^2) \quad (4.1)$$

The first variable has $K_1 = 3$ and two knots at 17.87 and 76.59.

$$f_1(x_{i1}) = 0.65x_{i1} - 0.03x_{i1}^2 + 0.0006x_{i1}^3 - 0.0006(x_{i1} - 17.87)^3 I(x_{i1} > 17.87) \quad (4.2)$$

$$+ 0.0003(x_{i1} - 76.59)^3 I(x_{i1} > 76.59) \quad (4.3)$$

Its structure has the following form:

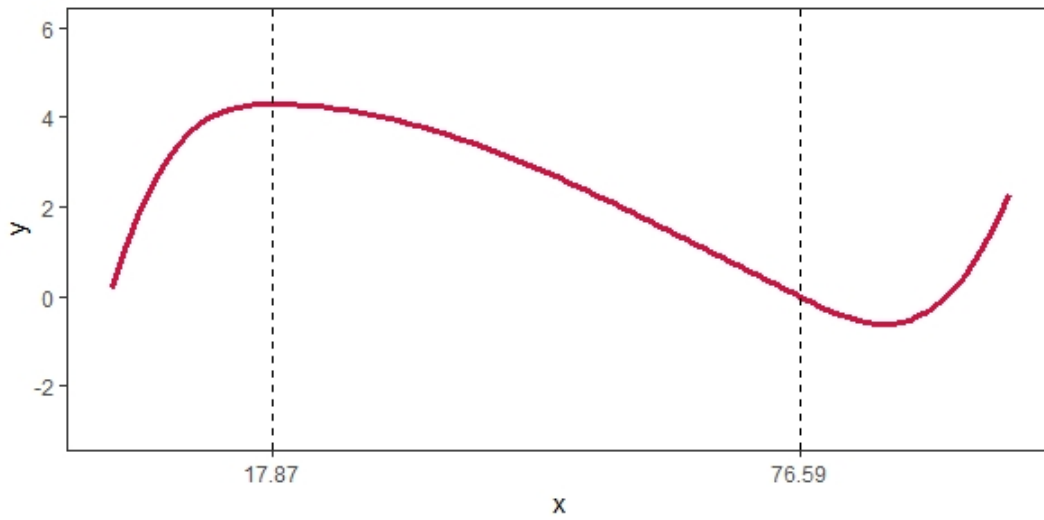


Figure 4.1: x_1 form

The second variable has $K_2 = 1$ and one knot at 90.63.

$$f_2(x_{i2}) = 0.02x_{i1} - 1.03(x_{i1} - 90.63)I(x_{i1} > 90.63) \quad (4.4)$$

Its structure has the following form:

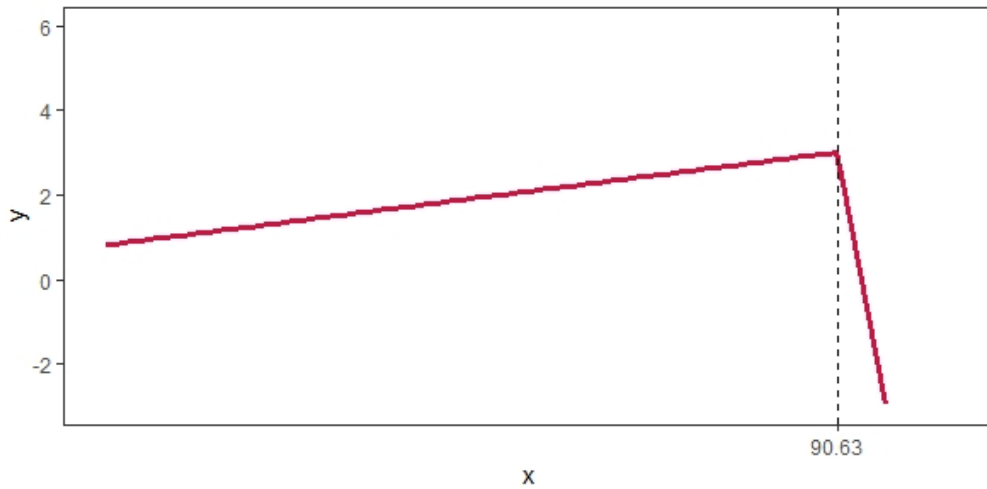


Figure 4.2: x_2 form

The third variable has $K_3 = 1$ with no knots.

$$f_3(x_{i3}) = 0.04x_{i3} \quad (4.5)$$

Its structure has the following form:

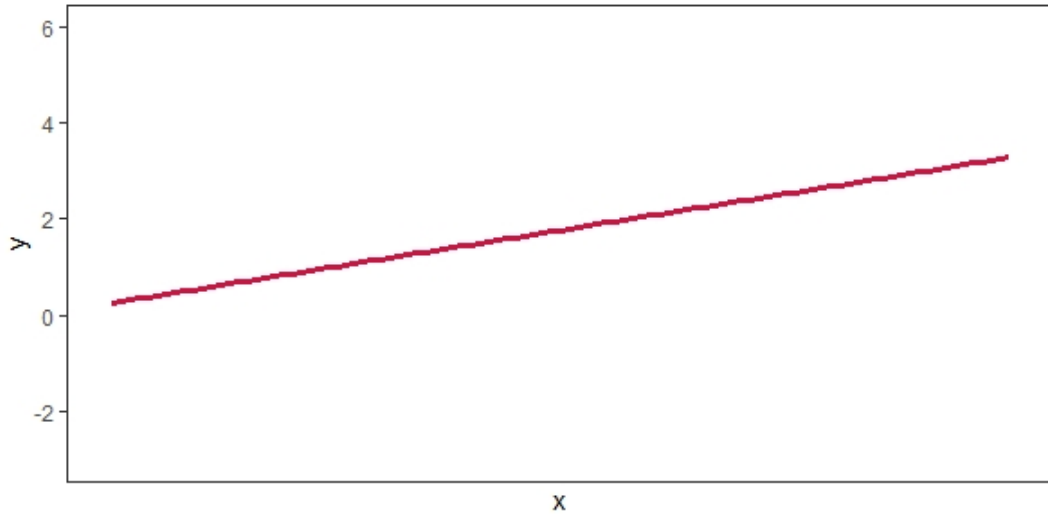


Figure 4.3: x_3 form

All models were tested in some scenarios given below:

1. $n = 300, 900, 1500, 3000$
2. The noise variance $\sigma^2 = 1, 2, 3$

4.1.1 Predictive results with $\sigma^2 = 1$

Models n	300	900	1500	3000
$J_1(\theta)$	1.06(1.52)	1.03(1.44)	0.94(1.39)	1.01(1.42)
$J_2(\theta)$	1.06(1.55)	1.01(1.43)	0.95(1.42)	1.02(1.43)
B-Splines	1.07(1.55)	0.99(1.40)	0.96(1.43)	1.02(1.45)
Ridge	2.31(3.35)	2.68(4.62)	2.38(4.05)	2.57(4.36)
Random Forest	1.03(1.41)	1.05(1.44)	0.95(1.43)	1.03(1.48)
Adaboost	1.18(1.69)	1.00(1.41)	0.97(1.45)	1.05(1.47)
MLP	1.91(2.72)	1.71(2.93)	1.42(1.97)	1.52(2.28)

Table 4.1: Simulations predictive results with $\sigma^2 = 1$

The two proposed methods had the best performance in all sample sizes or were competitive enough in the mean square error. The B-Splines model obtained very similar results to the proposed models and the best result when the sample size is 900, so there is no reason to choose one in this problem. The parameterization of the knots was not so worthwhile about predictions. It may be because the polynomial can capture the data structure without the knots location being improved. Also, the proposed models obtained better results most of the time concerning the Ridge, Adaboosting, and Neural Networks models. This is expected due to the nature of the simulated data. However, the random forest model obtained the best result when the sample size is 300, thus indicating that models with the knot structure may not always be the best.

4.1.2 Predictive results with $\sigma^2 = 2$

Models n	300	900	1500	3000
$J_1(\theta)$	4.21(6.24)	3.95(5.49)	3.74(5.58)	3.97(5.58)
$J_2(\theta)$	4.34(6.69)	3.94(5.48)	3.74(5.57)	3.97(5.60)
B Splines	4.23(6.52)	3.95(5.51)	3.80(5.68)	3.98(5.61)
Ridge	5.46(7.85)	5.77(8.14)	5.20(7.7)	5.70(8.62)
Random Forest	4.23(6.07)	4.07(5.39)	3.79(5.71)	3.99(5.62)
Adaboost	4.58(7.19)	4.31(5.5)	3.78(5.88)	4.10(5.75)
MLP	5.31(7.34)	4.89(6.75)	4.39(5.97)	4.70(6.73)

Table 4.2: Simulations predictive results with $\sigma^2 = 2$

The proposed models obtained the best results in all sample sizes. However, the B-Splines model had very similar results in the mean square error, so there is no reason to choose any of them. Furthermore, the proposed models had better results than the ridge, random forest, adaboosting, and neural networks predictive models.

4.1.3 Predictive results with $\sigma^2 = 3$

Models n	300	900	1500	3000
$J_1(\theta)$	9.92(16.06)	8.83(12.27)	8.43(12.59)	8.89(12.57)
$J_2(\theta)$	9.41(14.46)	8.78(12.13)	8.47(12.57)	8.89(12.55)
B Splines	9.36(14.45)	8.94(12.37)	8.55(12.71)	8.91(12.57)
Ridge	10.75(15.9)	10.80(14.64)	9.90(14.48)	10.76(15.75)
Random Forest	9.24(13.53)	9.05(12.27)	8.58(12.98)	8.99(12.55)
Adaboost	10.39(15.38)	9.19(12.12)	8.76(13.86)	9.04(12.83)
MLP	10.75(15.9)	10.07(13.61)	8.96(12.58)	9.59(13.52)

Table 4.3: Simulations predictive results with $\sigma^2 = 3$

The proposed models obtained the best results in almost all sample sizes. However, the B-Splines model again had similar results in the mean squared error, thus having no reason to choose any of them. Furthermore, the proposed models generally had better results than the ridge, random forest, adaboosting, and neural networks predictive models. However, the random forest was better when the sample size was 300.

4.2 Performance on Real Data

The best way to know if the proposed models are advantageous in practice is by testing them on real data with some of the most famous machine learning models. In this section, we test the models on six real datasets with different sample sizes, features, and various areas, such as civil engineering, biology, automobiles, and house prices. The data sets chosen are simple in pre-processing, observations with at least one missing data were excluded, and we used only numerical variables as the explanatory variables. All datasets were obtained from the UCI Machine Learning Repository [1].

The information about each dataset is presented below:

4.2.1 Airfoil Self-Noise Data Set

This dataset¹. has 1503 observations, five explanatory variables, and a target variable. This data set is about a series of aerodynamic and acoustic tests of two and three-dimensional airfoil blade sections conducted in an anechoic wind tunnel. The objective is to predict the scaled sound pressure level in decibels.

The details are given below:

Variables	Meaning
x_1	Frequency
x_2	Angle of attack
x_3	Chord length
x_4	Free-stream velocity
x_5	Suction side displacement thickness
y	Scaled sound pressure level

Table 4.4: *Details airfoil dataset*

4.2.2 Auto MPG Data Set

This dataset². has 398 observations, four explanatory variables, and a target variable. The data is for city cycle fuel consumption in miles per gallon, predicted in terms of some continuous attributes.

The details are given below:

Variables	Meaning
x_1	Displacement
x_2	Horsepower
x_3	Weight
x_4	Acceleration
y	Consumption in miles per gallon

Table 4.5: *Details auto mpg dataset*

4.2.3 Concrete Compressive Strength Data Set

This dataset³. has 1030 observations, eight explanatory variables, and a target variable. The target variable is concrete compressive strength which is the most important material in civil engineering and is a highly nonlinear function of a few attributes.

The details are given below:

¹Airfoil Self-Noise Data Set: <https://archive.ics.uci.edu/ml/datasets/Airfoil+Self-Noise>

²Auto MPG Data Set: <https://archive.ics.uci.edu/ml/datasets/auto+mpg>

³Concrete Compressive Strength Data Set: <https://archive.ics.uci.edu/ml/datasets/concrete+compressive+strength>

Variables	Meaning
x_1	Cement
x_2	Blast Furnace Slag
x_3	Fly Ash
x_4	Water
x_5	Superplasticizer
x_6	Coarse Aggregate
x_7	Fine Aggregate
x_8	Age
y	Concrete compressive strength

Table 4.6: *Details concrete dataset*

4.2.4 Combined Cycle Power Plant Data Set

This dataset⁴. has 9568 observations, four explanatory variables, and a target variable. The dataset was collected from a Combined Cycle Power Plant over six years (2006-2011), when the power plant was configured to work at full load. The features consist of hourly average environmental variables Temperature, Ambient Pressure, Relative Humidity, and Exhaust Vacuum to predict the plant's net hourly electrical energy output.

The details are given below:

Variables	Meaning
x_1	Temperature
x_2	Ambient Pressure
x_3	Relative Humidity
x_4	Exhaust Vacuum
y	Net hourly electrical energy

Table 4.7: *Details cycle power dataset*

4.2.5 QSAR aquatic toxicity Data Set

This dataset⁵. has 546 observations, five explanatory variables, and a target variable. This dataset was used to develop quantitative regression QSAR models to predict acute aquatic toxicity towards the fish *Pimephales promelas* (fathead minnow) on a set of 908 chemicals to predict acute aquatic toxicity towards *Daphnia Magna*. LC50 data(target) is the concentration that causes death in 50% of test *D magna* over a test duration of 48 hours. The features are about molecular descriptors, such as TPSA(Tot) (Molecular properties), SAacc (Molecular properties), MLOGP (Molecular properties), RDCHI (Connectivity indices) e GATS1p (2D autocorrelations).

The details are given below:

⁴Combined Cycle Power Plant Data Set: <https://archive.ics.uci.edu/ml/datasets/Combined+Cycle+Power+Plant>

⁵QSAR aquatic toxicity Data Set: <https://archive.ics.uci.edu/ml/datasets/QSAR+aquatic+toxicity>

Variables	Meaning
x_1	Molecular descriptor TPSA
x_2	Molecular descriptor SAacc
x_3	Molecular descriptor MLOGP
x_4	Molecular descriptor RDCHI
x_5	Molecular descriptor GATS1p
y	LC50

Table 4.8: *Details qsar dataset*

4.2.6 Real estate valuation data set Data Set

This dataset⁶. has 414 observations, five explanatory variables, and a target variable. Data set of real estate valuation collected from Taiwan to predict house prices.

The details are given below:

Variables	Meaning
x_1	House age
x_2	Distance to the nearest MRT station
x_3	Number of convenience stores in the living circle on foot
x_4	Latitude
x_5	Longitude
y	House price of unit area

Table 4.9: *Details valuation dataset*

4.2.7 Predictive Results

Models Datasets	airfoil	auto mpg	concrete	folds	qsar	valuation
$J_1(\theta)$	20.12(34.31)	15.29(36.17)	36.7(63.3)	17.23(31.72)	1.27(1.96)	38.02(75.6)
$J_2(\theta)$	20.94(34.74)	13.91(33.66)	40.71(70.92)	17.03(31.95)	1.31(2.19)	37.19(75.35)
B-Splines	18.8(30.55)	13.81(35.2)	55.78(86.9)	17.50(31.7)	1.34(1.85)	55.24(81.66)
Ridge	21.76(32.63)	15.83(28.73)	105.13(147.98)	20.09(34.32)	1.29(1.81)	53.25(94.34)
Random Forest	15.73(23.69)	14.44(37.78)	68.46(135.13)	18.83(34.17)	1.25(1.85)	33.07(75.45)
Adaboost	12.53(16.02)	17.15(36.93)	55.99(98.93)	29.98(44.35)	1.43(2.05)	42.6(74.31)
MLP	44.46(58.48)	17.2(28.13)	43.64(74.83)	22.5(37.73)	1.32(2.48)	48.71(111.1)

Table 4.10: *Real data predictive results*

The results in real data sets were quite attractive. To begin with, the proposed models were competitive in almost all datasets or were the ones with the best predictive performance. The random forest model was the best in the airfoil dataset, and the proposed models were not competitive enough. Secondly, the proposed models obtained the best predictive result than the other tested models in the concrete and fold datasets, suggesting that the models proposed in this dissertation can be chosen for some problems in practice, as they were better than well-structured models in the literature. Contrary to the simulations in the previous section, there were situations where the proposed models were superior to the B-Splines model, suggesting the proposal to add knots as parameters can make much difference in practice to improve the predictive performance.

⁶Real estate valuation data set Data Set: <https://archive.ics.uci.edu/ml/datasets/Real+estate+valuation+data+set>

Now, let us carefully check the proportion of datasets where the proposed models were superior in predictive performance than the other models.

Models	$J_1(\theta)$	$J_2(\theta)$
B-Splines	4/6	4/6
Ridge	6/6	5/6
Random Forest	2/6	3/6
Adaboost	5/6	6/6
MLP	6/6	6/6

Table 4.11: *Predictive performance comparison*

A great accomplishment is that the proposed models were superior in 4 of the 6 data sets than the most famous model with knots(B-Splines) structure. Of the four datasets, the valuation and concrete sets showed the most significant difference, suggesting that considering knots as parameters can make a difference in practice.

Another significant achievement is that the proposed models were superior in most sets than the ridge, neural networks, and boosting models. It was also superior to neural networks and AdaBoost in almost all datasets except one. This shows that our approach can be used in practice against these known algorithms once again.

Undoubtedly, the most competitive algorithm against our proposed models was the random forest, but even so, we gained predictive power in half of the datasets, making it feasible to compare against this model in practice.

Chapter 5

Conclusion

Models and statistical techniques must always follow the complexity of the data. The proposed models proved to be highly competitive with the parameterization of knots because they obtain the best performances in real sets against well-known machine learning methods.

In this dissertation, we mainly contribute to the field of predictive models, first by introducing new penalties for this type of model and using the BFGS algorithm's optimization methodology to estimate the parameters of the proposed models.

Also, the addition of knots location and number proved to be worthwhile to improve the performance of predictive power in general. For example, it won most of the times the most used methods in the literature that use the structure of knots.

The proposed models can be flexible enough to predict observations from the simplest to the most challenging cases, with many variables or observations, without overfitting and adequately selecting the knots number. In the case of many variables, the model can adjust to this case, giving simplicity to the most straightforward variables, either with few knots or lower K , and giving more complexity to variables with high importance for the model and doing it efficiently with the BFGS method.


The proposed models did not show much difference between the performances, both in the simulations and in the real dataset. In some situations, a model was superior in selecting the knot number but this did not affect the predictions at all. The two can be tested in practice to predict observations most of the time.

The convergence of knots is also something to work on in the future. The BFGS helps a lot, and we will study if when the sample increases, the estimate stays closer and closer to the true value of the parameters, which may suggest probability or almost sure convergence.

To conclude, in future works, we hope to offer the method's R package for any real dataset, and the user will be able to work on the predictions with greater freedom with the cost functions proposed in this dissertation.

Appendix A

Code

All the codes of this dissertation are in the following GitHub repository:  [link](#)

Bibliography

- [1] A. Asuncion and D. Newman. Uci machine learning repository, 2007. [63](#)
- [2] C. G. Broyden. Quasi-newton methods and their application to function minimisation. *Mathematics of Computation*, 21(99):368–381, 1967. [33](#)
- [3] C. G. Broyden. The convergence of a class of double-rank minimization algorithms: 2. the new algorithm. *IMA journal of applied mathematics*, 6(3):222–231, 1970. [33](#)
- [4] C. De Boor and C. De Boor. *A practical guide to splines*, volume 27. springer-verlag New York, 1978. [4](#)
- [5] R. Fletcher. A new approach to variable metric algorithms. *The computer journal*, 13(3):317–322, 1970. [33](#)
- [6] R. Fletcher and M. J. Powell. A rapidly convergent descent method for minimization. *The computer journal*, 6(2):163–168, 1963. [33](#)
- [7] J. Friedman, T. Hastie, R. Tibshirani, et al. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001. [4](#)
- [8] D. Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of computation*, 24(109):23–26, 1970. [33](#)
- [9] I. Griva, S. G. Nash, and A. Sofer. *Linear and nonlinear optimization*, volume 108. Siam, 2009. [28](#), [29](#), [34](#), [35](#)
- [10] T. Hastie and R. Tibshirani. Generalized additive models: some applications. *Journal of the American Statistical Association*, 82(398):371–386, 1987. [1](#), [3](#), [4](#)
- [11] N. J. Higham. *Accuracy and stability of numerical algorithms*. SIAM, 2002. [35](#)
- [12] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. [9](#)
- [13] R. Izbicki and T. M. dos Santos. *Aprendizado de máquina: uma abordagem estatística*. Rafael Izbicki, 2020. [4](#)
- [14] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013. [1](#), [5](#), [12](#)
- [15] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006. [34](#)
- [16] J. Nocedal and Y.-x. Yuan. Combining trust region and line search techniques. In *Advances in nonlinear programming*, pages 153–175. Springer, 1998. [29](#)
- [17] D. Ruppert. Selecting the number of knots for penalized splines. *Journal of computational and graphical statistics*, 11(4):735–757, 2002. [24](#)

- [18] D. F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, 24(111):647–656, 1970. [33](#)
- [19] A. R. d. S. Sousa, M. T. Severino, and F. G. Leonardi. Model selection criteria for regression models with splines and the automatic localization of knots. *arXiv preprint arXiv:2006.02649*, 2020. [10](#), [25](#)
- [20] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996. [9](#)
- [21] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005. [9](#)