THE ROYAL INSTITUTE OF TECHNOLOGY

THE DIVISION OF OPTIMIZATION AND SYSTEMS THEORY

Curve Fitting Using Calculus in Normed Spaces

A BACHELOR'S THESIS IN

OPTIMIZATION

Authors:

Eva Andersson (871019-0029)
evanders@kth.se

Filippa Kjaerboe (890331-0905)
kjaerboe@kth.se

Otto Sjöholm (890425-0373)
osjoholm@kth.se

Marlene Sandström (890301-8565)
marsand@kth.se

Supervisor:

Associate Professor
Amol Sasane

May 11, 2011

Abstract

Curve fitting is used in a variety of fields, especially in physics, mathematics and economics. The method is often used to smooth noisy data and for doing path planning. In this bachelor thesis calculus of variations will be used to derive a formula for finding an optimal curve to fit a set of data points. We evaluate a cost function (defined on the set of all curves f on the interval $[a, b]$) given by $F(f) = \int_a^b (f''(x))^2 \mathrm{d}x + \lambda \sum_{i=1}^n (f(x_i) - y_i)^2$. The integral term represents the smoothness of the curve, the interpolation error is given by the summation term and $\lambda > 0$ is defined as the interpolation parameter. An ideal curve minimizes the interpolation error and is relatively smooth. This is problematic since a smooth function generally has a large interpolation error when doing curve fitting, and therefore the interpolation parameter λ is needed to decide how much consideration should be given to each attribute. For the cost function F a larger value of λ decreases the interpolation error of the curve. The analytical calculations performed made it possible to construct a MATLAB program, that could be used to solve the minimization problem. In the result part some examples are presented for different values of λ. The conclusion is that a larger value of the interpolation parameter λ is generally needed when using more data points and if the points are closely placed on the x-axis. Further on, a method called Ordinary Cross Validation (OCV) is evaluated to find an optimal value of λ. This method gave good results, except for the case when the points could almost be fitted with a straight line.

Sammanfattning

Kurvanpassning används inom flera olika ämnesområden, särskilt fysik, matematik och ekonomi. Metoden används ofta vid anpassning av mätdata och vid banplanering. I denna kandidatexamensuppsats används variationskalkyl för att ta fram en optimal kurva som passar mätdata. Vi utvärderar kostnaden $F(f) = \int_a^b (f''(x))^2 \mathrm{d}x + \lambda \sum_{i=1}^n (f(x_i) - y_i)^2$, som är definierad på mängden av alla kurvor f på intervallet $[a, b]$. Intergraltermen representerar kurvans släthet medan interpolationsfelet ges av summatermen där $\lambda > 0$ definieras som interpolationsparametern. En ideal kurva minimerar interpolationsfelet och är relativt slät. En svårighet är att en slät funktion ofta har en stor felkvadratsumma och därför används konstanten λ för att bestämma vilken av de två egenskaperna som ska väga tyngst. En ökning av värdet på λ ger ett mindre interpolationsfel för kurvan. Våra analytiska beräkningar gav oss en metod för att skriva om problemet, vilket ledde till att ett MATLAB-program kunde konstrueras för att lösa minimeringsproblemet. I resultatdelen presenteras exempel med olika värden på konstanten λ. Slutsatsen är att det generellt sett behövs ett högre λ-värde när man använder många punkter och när punkterna ligger nära varandra på x-axeln. Vidare i rapporten utvärderas Ordinary Cross Validation (OCV), för att hitta ett optimalt värde på λ. Denna metod gav bra resultat, förutom då punkterna nästan kunde anpassas med en rät linje.

Contents

Chapter 1

Introduction

1.1 Curve Fitting as an Optimization Problem

The main purpose of this thesis is to fit a curve to a given set of data points, using calculus of variations. Data is given as points $(x_1, y_1), ..., (x_n, y_n) \in \mathbb{R}^2$. Our task is to find a function that fits the points well, meaning that the interpolation error is small. The function should also be relatively smooth and a condition for smoothness is defined in the report. These two conditions will give rise to an optimization problem. The aim with this report is to find analytical methods to solve the minimization problem and then use the results to construct a MATLAB program.

1.2 Applications of Curve Fitting

Curve fitting can be used to analyse data in several different fields such as mathematics, statistics, economics, physics and biology. There exist several methods of curve fitting, whereof cubic splines is the one we will look into and use in this thesis. A spline is a function defined piecewise as polynomials. Two examples of where splines are particularly useful are path planning and statistics. In statistics the main use is to smooth noisy data. Other sorts of curve fitting are linear and nonlinear regression, where one fits a line respectively a function to a set of data. Nonlinear curve fitting is often used in biology, for example to analyse the diversity of species and species distribution. [2] [3]

1.3 Structure of Thesis

Chapter 2
Optimization in the one variable case will be considered and theorems that are needed for calculus in normed spaces will be recalled.
Chapter 3
To find an optimal solution one needs to define the derivative between normed spaces. The conditions of linearity and continuity for the derivative are also investigated.
Chapter 4
Properties are defined that needs to be satisfied, in order to solve the optimization problem. The convexity of the cost function is also shown in this chapter.
Chapter 5
The analytical problem is rewritten to a numerical problem that can be solved using MATLAB. In the MATLAB program the user can choose how much consideration should be taken to the interpolation error compared to the smoothness of the curve.

Chapter 6
This part consists of graphical results, where different sets of data are used. The method of Ordinary Cross Validation is evaluated and used to find an optimal value of the interpolation parameter λ.
Chapter 7
In the last chapter our work is discussed, and future development on the subject is suggested.

Chapter 2

Background

In calculus of variations one defines a functional L, whose argument lives in a set of functions: $f \overset{L}{\mapsto} L(f)$ and depends on a function f. Calculus of variations is about finding a function that minimizes or maximizes the functional. This thesis will focus on the problem of fitting a curve to a set of data points, given by $(x_1, y_1), ..., (x_n, y_n) \in \mathbb{R}^2$ within an interval $[a, b]$, see Figure 2.2. The theories and solutions can also be used to fit data from empirical studies.

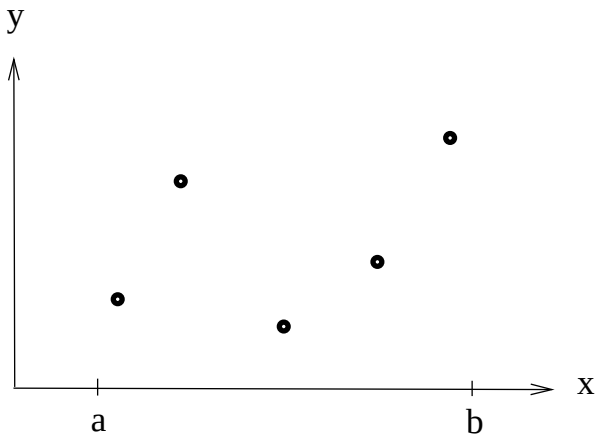

Figure 2.1: *Points on an interval* $[a, b]$.

Ideally, we want a function f which gives a small error $f(x_i) - y_i$ for each i and is relatively smooth. We will write this as an optimization problem in a space of functions. Calculus of variations will be used to derive a formula for the optimal curve, which will be the desired function that fits the data. The optimal curve is piecewise defined by cubic polynomials, in other words a cubic *spline*. A spline is a function consisting of piecewise polynomial functions on the interval $[a, b]$. The interval $[a, b]$ is divided into subintervals and the spline S is defined by different polynomials $P_1, P_2, ..., P_n$:

$$S(x) = \begin{cases} P_1(x) & a < x < x_1, \\ P_2(x) & x_1 < x < x_2, \\ P_3(x) & x_2 < x < x_3, \\ \quad \vdots & \\ P_n(x) & x_{n-1} < x < b. \end{cases} \tag{2.1}$$

As mentioned before we will pose the curve fitting problem as an optimization problem. In order to do this, we first need to investigate the interpolation error and smoothness of the curve.

The interpolation error is defined as:

$$\sum_{i=1}^{n}(f(x_i) - y_i)^2. \tag{2.2}$$

A small interpolation error means that you get a curve that fits the points very well, but it will probably not be smooth. Furthermore the function might take very large or small values at the extreme points.

We introduce

$$I = \int_a^b (f''(x))^2 \mathrm{d}x \tag{2.3}$$

as a way of measuring smoothness of f. One way to evaluate smoothness is to observe how the second derivative of a function varies. Large fluctuations will lead to a large value of the integral I. This means that the function is not particularly smooth. A small value of the integral I would mean that function f changes slowly and therefore is smoother. In addition to having a smooth function, we also want the interpolation error to be small. This is problematic, consider the example of fitting the points with a straight line as in 2.2. In this case $f''(x) = 0$, for $x \in [a, b]$, meaning that I equals zero, so the curve will be as smooth as possible. Instead you get a large interpolation error, assuming the points are not on a straight line.

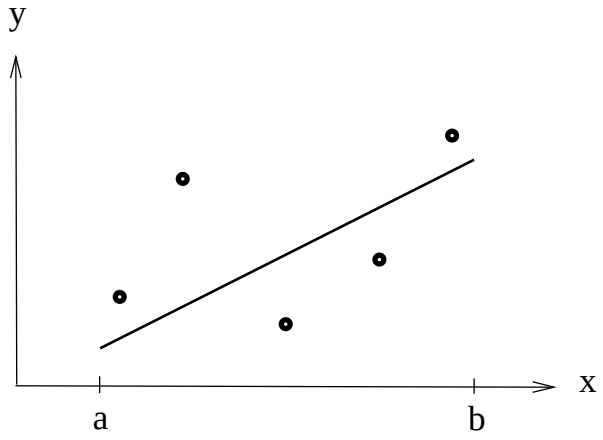

Figure 2.2: *Curve fitting with a very smooth function.*

On the other hand, consider just joining the points with straight lines as in Figure 2.3. This will give us an interpolation error equal to zero, but a very large smoothness term I. The conclusion is that there exists a trade off between interpolation error and smoothness of the curve.

Since we want to minimize both the interpolation error and the smoothness term this leads to the following minimization problem. We use (2.2) and (2.3) to define the cost $F(f)$ of any $f : [a, b] \to \mathbb{R}$ as

$$F(f) := \int_a^b (f''(x))^2 \mathrm{d}x + \lambda \sum_{i=1}^{n}(f(x_i) - y_i)^2 \tag{2.4}$$

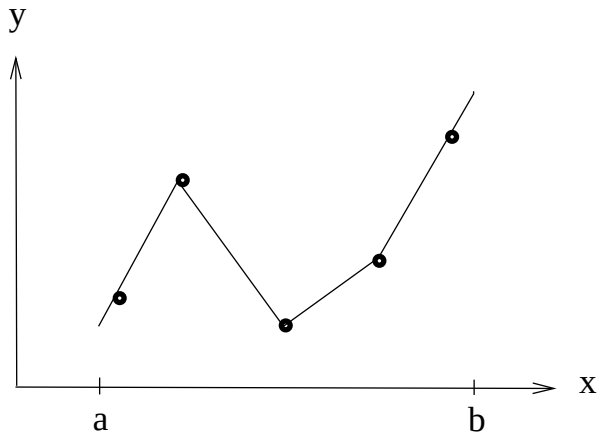

Figure 2.3: *Curve fitting without interpolation error.*

where $\lambda > 0$ is called the interpolation parameter which implies how much consideration should be given to the interpolation error, compared to the smoothness of the function.

Further on in the report, the following minimization problem will be considered:

$$(P) : \begin{cases} \text{minimize} & F(f) \\ \text{subject to} & f : [a, b] \to \mathbb{R}. \end{cases} \tag{2.5}$$

The one variable case will be examined first and then the corresponding methods in normed spaces will be investigated, to find an appropriate function f that minimizes F.

2.1 Optimization in One Variable

Consider the function $g : \mathbb{R} \to \mathbb{R}$. To find a local minimizer in ordinary calculus one first calculates the derivative $g'(x)$. If $g'(x)$ equals zero for a point x_0, then x_0 is a critical point. This means that x_0 can be a saddle point or a point that maximizes or minimizes the function g. In our problem we are interested in minimizing g, which leads us to look at convexity of the function.

The function g is convex if it satisfies the following, for every two points x_1 and x_2 and $t \in [0, 1]$:

$$g(tx_1 + (1 - t)x_2) \leq tg(x_1) + (1 - t)g(x_2). \tag{2.6}$$

In the one variable case one can look at the second derivative to see if the function is convex. If the second derivative $g''(x) \geq 0, \forall x \in \mathbb{R}$ we know that the function is convex. These results are stated in two theorems [1] below:

Theorem 2.1.1 *If $x_0 \in \mathbb{R}$ is a minimizer of $g : \mathbb{R} \to \mathbb{R}$, then $g'(x_0) = 0$.*

Theorem 2.1.2 *If $g''(x) \geq 0, \forall x$ and $g'(x_0) = 0$, then x_0 is a minimizer of g.*

We need to find equivalent results for our case, in order to find the function f that is optimal for the problem (P).

2.2 Definitions

2.2.1 Vector Space

Definition A *vector space* over $\mathbb{R}$, is a set X together with two functions, $+ : X \times X \to X$, called *vector addition*, and $\cdot : \mathbb{R} \times X \to X$, called *scalar multiplication* that satisfy following:

V1. For all $x_1, x_2, x_3 \in X, x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3$.
V2. There exists an element, denoted by $\mathbf{0}$ (called the *zero vector*) such that for all $x \in X$, $x + \mathbf{0} = \mathbf{0} + x = x$.
V3. For every $x \in X$, there exists an element, denoted by $-x$, such that $x + (-x) = \mathbf{0}$.
V4. For all $x_1, x_2 \in X, x_1 + x_2 = x_2 + x_1$.
V5. For all $x \in X, 1 \cdot x = x$.
V6. For all $x \in X$ and all $\alpha, \beta \in \mathbb{R}, \alpha \cdot (\beta \cdot x) = (\alpha\beta) \cdot x$.
V7. For all $x \in X$ and all $\alpha, \beta \in \mathbb{R}, (\alpha + \beta) \cdot x = \alpha \cdot x + \beta \cdot x$.
V8. For all $x_1, x_2 \in X$ and all $\alpha \in \mathbb{R}, \alpha \cdot (x_1 + x_2) = \alpha \cdot x_1 + \alpha \cdot x_2$.

To solve our problem (P) we need to specify the domain for the function F. For an example, that the first and second derivatives of f exist. We do this by defining a vector space:

$$C^{(2)}[a,b] = \left\{ f : [a,b] \to \mathbb{R} : \forall x \in [a,b], f'(x), f''(x) \text{ exist}, f(x), f'(x), f''(x) \text{ are continuous in } [a,b] \right\}.$$

In order to make $C^{(2)}[a,b]$ a vector space over $\mathbb{R}$, we define addition and scalar multiplication as follows $\forall x \in [a,b]$, for $f, g \in C^{(2)}[a,b]$ and $\alpha \in \mathbb{R}$: [6]

$$(f+g)(x) = f(x) + g(x), \tag{2.7}$$
$$(\alpha f)(x) = \alpha f(x). \tag{2.8}$$

Consider the definition of the first derivative of a function in the one variable case:

$$f'(x_0) = \lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0}. \tag{2.9}$$

To define the derivative for functions between vector spaces, we also need to define distances between functions in $C^{(2)}[a,b]$. Since there is no way of defining a distance in a general vector space, a normed space is needed.

2.2.2 Normed Space

Definition Let X be a vector space over $\mathbb{R}$ or $\mathbb{C}$. A *norm* on X is a function $||\cdot|| : X \to [0, +\infty)$ such that:

N1. *(Positive definiteness)* For all $x \in X, ||x|| \geq 0$. If $x \in X$ then $||x|| = 0$ iff $x = 0$.
N2. For all $\alpha \in \mathbb{R}$ (respectively $\mathbb{C}$) and for all $x \in X, ||\alpha x|| = |\alpha| ||x||$.
N3. *(Triangle inequality)* For all $x, y \in X, ||x + y|| \leq ||x|| + ||y||$.

A normed space is a vector space equipped with a norm. In a normed vector space the distance between two vectors x and y is defined as $||x-y||$. $C^{(2)}[a,b]$ is a normed space for the following norm for f:

$$||f|| := \max_{x\in[a,b]} |f(x)| + \max_{x\in[a,b]} |f'(x)| + \max_{x\in[a,b]} |f''(x)|. \tag{2.10}$$

2.2.3 Fréchet Derivative

After defining a normed space and distances between vectors we can define the derivative of a function. There exist several notions for the derivative. In this report, the Fréchet derivative is used without exceptions. This notion is a generalization of the ordinary derivative of a function on the real numbers $f : \mathbb{R} \to \mathbb{R}$.

To define the derivative a Banach space is used, which is a complete vector space with a norm $||\cdot||$. A complete vector space is a space of functions comprising a complete biorthagonal system. A further explanation of these terms lie beyond the scope of this thesis but can be found in for example *Foundations of Modern Analysis* by A. Friedman.

The Fréchet derivative is defined in the following way.

Definition Let V and W be Banach spaces and U be an open subspace of V. A function $f : U \to W$ is Fréchet differentiable at x_0 if there exists a bounded linear operator $L : V \to W$ such that:

$$\lim_{x\to x_0} \frac{||F(x) - F(x_0) - L(x-x_0)||_V}{||x-x_0||_W}. \tag{2.11}$$

Or equivalently, that for any $\epsilon > 0$ there exists a $\delta > 0$ so $\forall x \in X$ satisfying $0 < ||x-x_0||_W < \delta$, then:

$$\frac{||F(x) - F(x_0) - L(x-x_0)||_V}{||x-x_0||_W} \leq \epsilon. \tag{2.12}$$

The Fréchet derivative of a function F of variable x is henceforth denoted by $F'(x)$.

2.3 Theorems for Finding a Minimizer

To solve the problem (P) we need to seek similar theorems as for the one variable case. In the following theorems the notation $\mathbf{0}$ is used for the zero transformation, which is defined as the transformation which sends every vector X to the zero vector in $\mathbb{R}$. [1]

Theorem 2.3.1 *If $x_0 \in X$ is a minimizer of $F : X \to \mathbb{R}$ then $F'(x_0) = \mathbf{0}$.*

The second result that we will use is that for a convex mapping $F : X \to \mathbb{R}$, the above necessary conditions are sufficient.

Theorem 2.3.2 *If $F : X \to \mathbb{R}$ is convex and $F'(x_0) = \mathbf{0}$, then x_0 is a minimizer of F.*

Chapter 3

Calculation of Derivative

The purpose of the analytical calculations is to solve (P), as mentioned in section 2.5. To use Theorem 2.3.1, it must be shown that the function F is Fréchet differentiable at f_0. In this chapter it will be confirmed that the function F has a derivative that equals a linear and continuous transformation L. This is a step in the right direction, since this enables us to show later on in the report that this is the zero linear transformation.

3.1 Finding the Linear Transformation

We start our investigation by making a guess for $F'(f_0)$ and then check if our guess is correct, meaning it satisfies the conditions of linearity and continuity. Let $f, f_0 \in C^{(2)}[a,b]$ and calculate:

$$F(f) - F(f_0) = \int_a^b ((f''(x))^2 - (f_0''(x))^2)\mathrm{d}x + \lambda \sum_{i=1}^{n} \left((f(x_i) - y_i)^2 - (f_0(x_i) - y_i)^2\right)$$

$$= \int_a^b (f''(x) + f_0''(x))(f''(x) - f_0''(x))\mathrm{d}x + \lambda \sum_{i=1}^{n} (f(x_i) + f_0(x_i) - 2y_i)(f(x_i) - f_0(x_i))$$

$$\approx \int_a^b 2f_0''(x)(f''(x) - f_0''(x))\mathrm{d}x + \lambda \sum_{i=1}^{n} 2(f_0(x_i) - y_i)(f(x_i) - f_0(x_i)) = L(f - f_0).$$

where $L : C^{(2)}[a,b] \to \mathbb{R}$ is given by:

$$L(h) = \int_a^b 2f_0''(x)h''(x)\mathrm{d}x + \lambda \sum_{i=1}^{n} 2(f_0(x_i) - y_i)h(x_i). \tag{3.1}$$

So the guess for the derivative is

$$F'(f_0) = L. \tag{3.2}$$

For the guess to hold there are some requirements on L:

1) The transformation L has to be linear.

2) The transformation L has to be continuous.

3.1.1 Linearity

The next step in the investigation is to check if the transformation L is linear.

Definition A mapping T of a vector space X into a vector space Y is said to be a linear transformation if, for all $x_1, x_2 \in X$ and all scalars c

$$T(x_1 + x_2) = T(x_1) + T(x_2), \tag{3.3}$$

$$T(cx_1) = cT(x_1) \tag{3.4}$$

are satisfied.

Let $h_1, h_2 \in C^{(2)}[a, b]$, then it can be shown that:

$$L(h_1 + h_2) = Lh_1 + Lh_2. \tag{3.5}$$

Further on, let $h \in C^{(2)}[a, b]$, $\alpha \in \mathbb{R}$. Then

$$L(\alpha h) = \alpha L(h) \tag{3.6}$$

can be proven, which means that the function L is a linear transformation.

3.1.2 Continuity

After confirming that L is a linear transformation, we need to verify that L is continuous. [6]

Theorem 3.1.1 *A linear transformation $L : X \to Y$ is continuous if and only if $\forall x \in X$ there $\exists M > 0$ such that $||Lx||_Y \leq M||x||_X$.*

Thus the next step is to see if there $\exists M$ such that, $\forall h \in C^{(2)}[a, b]$, $|Lh| \leq M \parallel h \parallel$. Let $h \in C^{(2)}[a, b]$. Using the norm from (2.10) we obtain

$$\parallel h \parallel = \parallel h \parallel_\infty + \parallel h' \parallel_\infty + \parallel h'' \parallel_\infty, \tag{3.7}$$

which gives:

$$\parallel h \parallel \geq \parallel h \parallel_\infty, \quad \parallel h \parallel \geq \parallel h' \parallel_\infty \text{ and } \parallel h \parallel \geq \parallel h'' \parallel_\infty . \tag{3.8}$$

The result can be used to verified that:

$$|Lh| = \left| \int_a^b 2f_0''(x)h''(x)dx + \lambda \sum_{i=1}^{n} 2(f_0(x_i) - y_i)h(x_i) \right| \tag{3.9}$$

$$= \underbrace{\left[\int_a^b 2|f_0''(x)|dx + \lambda \sum_{i=1}^{n} 2|f_0(x_i) - y_i| \right]}_{=:M} \parallel h \parallel$$

$$= M \parallel h \parallel . \tag{3.10}$$

This shows that the transformation L is continuous.

3.2 Demonstrating that F is Differentiable at f_0

It is now proven that the transformation L satisfies the requirements of continuity and linearity. Recall the guess from (3.2). The question whether the assumption

$$F'(f_0) = L$$

is correct or not still remains. Using the definition of the Fréchet Derivative from section 2.2.3

$$\frac{\parallel F(f) - F(f_0) - L(f - f_0) \parallel}{\parallel f - f_0 \parallel} \leq \epsilon \tag{3.11}$$

where $\epsilon > 0$, it can be verified that the function F is differentiable at the point f_0.

Expanding the numerator in (3.11), we can see that:

$$F(f) - F(f_0) - L(f - f_0) = \int_a^b \left(f''(x) - f_0''(x)\right)^2 dx \ + \lambda \sum_{i=1}^n \left(f(x_i) - f_0(x_i)\right)^2. \tag{3.12}$$

Further, it can be verified that this leads to

$$|F(f) - F(f_0) - L(f - f_0)| = [b - a + \lambda n] \parallel f - f_0 \parallel^2 \tag{3.13}$$

by using (3.8).

For all $f \in C^{(2)}[a, b]$ such that $0 <\parallel f - f_0 \parallel$, let:

$$\begin{aligned} \frac{\parallel F(f) - F(f_0) - L(f - f_0) \parallel}{\parallel f - f_0 \parallel} &\leq (b - a + \lambda n) \parallel f - f_0 \parallel \\ &< \delta(b - a + \lambda n) \\ &= \epsilon. \end{aligned} \tag{3.14}$$

Given any $\epsilon > 0$, choose:

$$\delta := \frac{\epsilon}{b - a + \lambda n}. \tag{3.15}$$

When $\delta > 0$ and $\forall f \in C^{(2)}[a, b]$ such that $0 <\parallel f - f_0 \parallel< \delta$, it yields:

$$\begin{aligned} \frac{\parallel F(f) - F(f_0) - L(f - f_0) \parallel}{\parallel f - f_0 \parallel} &< (b - a + \lambda n) \cdot \delta \qquad (3.16) \\ &= (b - a + \lambda n) \cdot \frac{\epsilon}{b - a + \lambda n} \\ &= \epsilon. \qquad (3.17) \end{aligned}$$

Hence:

$$\boxed{F'(f_0) = L = \int_a^b 2f_0''(x)h''(x)\mathrm{d}x + \lambda \sum_{i=1}^n 2(f_0(x_i) - y_i)h(x_i).} \tag{3.18}$$

This proves that the guess (3.2) is correct.

Chapter 4

When is the Derivative Zero?

In the previous chapter it is proven that the derivative of the function F equals the transformation L. To use Theorem 2.3.1, the derivative has to equal zero. One way to show this is to prove that some specific properties holds for f_0, if $F'(f_0) = \mathbf{0}$.

4.1 Properties of a Minimizer

If $F'(f_0) = \mathbf{0}$, we can show that f_0 satisfies the properties listed below:

(1) (a) $f_0^{(4)}(x) = 0$ for all $x \in [a,b] \setminus \{a, x_1, x_2, \ldots, x_{n-1}, x_n, b\}$,

(2) (a) $f_0'''(a^+) = 0$,
(b) $f_0'''(x_i^+) - f_0'''(x_i^-) = -\lambda(f_0(x_i) - y_i)$ for $i = 1, 2, \ldots, n$,
(c) $f_0'''(b^-) = 0$,

(3) (a) $f_0''(a) = 0$,
(b) $f_0''(b) = 0$.

In previous sections we have only proven that $F'(f_0) = L$. $L = \mathbf{0}$ means that the transformation L maps any vector in $C^{(2)}[a,b]$ to zero. More explicitly:

$$\int_a^b 2f_0''(x) \cdot h''(x)dx + 2\lambda \sum_{i=1}^{n} (f_0(x_i) - y_i)h(x_i) = 0 \qquad \forall h \in C^{(2)}[a,b]. \tag{4.1}$$

In the next three sections of the report, brief proofs will be given for the properties.

4.1.1 First Property

Start with (4.28) and the property

$$(1a) \quad f_0^{(4)}(x) = 0 \text{ for all } x \in [a,b] \setminus \{a, x_1, x_2, \ldots, x_{n-1}, x_n, b\}$$

and assume that $L = \mathbf{0}$. Set $\Omega := [a,b] \setminus \{a, x_1, \ldots, x_n, b\}$, and assume that $f_0 \in C^{(4)}(\Omega)$.

Using the conventions

$$\begin{cases} x_o := a, \\ x_{n+1} := b, \end{cases} \tag{4.2}$$

it can be shown that it is always possible to find an $h_{1a} \in [x_i, x_{i+1}]$, $\forall i = 0, \ldots, n$ that equals zero at the endpoints. In this case fix $i \in 0, 1, \ldots, n$ and let $h_{1a} \in C^{(2)}[a, b]$ be such that:

$$h_{1a}(x) = 0 \quad \forall x \notin [x_i, \ldots, x_n],$$
$$\begin{cases} h_{1a}(x_i) = 0, \\ h_{1a}(x_{i+1}) = 0, \\ h'_{1a}(x_i) = 0, \\ h'_{1a}(x_{i+1}) = 0. \end{cases}$$

Recalling from equation (4.1), where $L = \mathbf{0}$:

$$\int_a^b 2f_0''(x) \cdot h_{1a}''(x)dx + 2\lambda \sum_{i=1}^{n} (f_0(x_i) - y_i) \underbrace{h_{1a}(x_i)}_{=0} = 0. \tag{4.3}$$

Together with the chosen h_{1a}, this formula can be rewritten as:

$$\int_{x_i}^{x_{i+1}} f_0''(x) \cdot h_{1a}''(x)dx = 0. \tag{4.4}$$

Further, integrating by parts two times gives:

$$\int_{x_i}^{x^{i+1}} f_0^{(4)}(x) h_{1a}(x)dx = 0. \tag{4.5}$$

Claim: This implies that $f_0^{(4)}(x) = 0$ in (x_i, x_{i+1}).

To show the reverse implication, still assume $L = \mathbf{0}$ and $f_0 \in C^4(\Omega)$. Suppose that $\exists \psi \in (x_i, x_{i+1})$ such that $f_0^{(4)}(\psi) \neq 0$. Without loss of generality also assume $f_0^{(4)}(\psi) > 0$. Since $f_0^{(4)}$ is continuous $\exists (\alpha, \beta) \subset (x_i, x_{i+1})$ where $f_0^{(4)}$ remains positive.

Now take an $h_{1b} \in C^2[\alpha, \beta]$, that is always positive. For example:

$$h_{1b}(x) = \begin{cases} (x - \alpha)^3(\beta - x)^3 & \forall x \in [\alpha, \beta], \\ 0 & \forall x \notin [\alpha, \beta]. \end{cases}$$

When using h_{1b} instead of h_{1a} in (4.5), it gives that:

$$\int_\alpha^\beta \underbrace{f_0^{(4)}(x)}_{>0} \underbrace{h_{1b}(x)}_{>0} dx = 0. \tag{4.6}$$

This is a contradiction and therefore

$$\boxed{f_0^{(4)}(x) = 0 \text{ for } x \in [a, b] \setminus \{a, x_1, x_2, \ldots, x_{n-1}, x_n, b\}} \tag{4.7}$$

holds.

Integration of $f_0^{(4)} = 0$ shows us that f_0 is going to be a piecewise cubic function in each interval (x_i, x_{i+1}), on the form

$$f_0(x) = A_i + B_i x + C_i x^2 + D_i x^3,$$

where A_i, B_i, C_i and D_i are constants.

4.1.2 Second Property

The second set of properties for the minimizing function f_0 are the following

$$\begin{aligned} &(2a) \quad f_0'''(a^+) = 0, \\ &(2b) \quad f_0'''(x_i^+) - f_0'''(x_i^-) = -\lambda(f_0(x_i) - y_i) \text{ for } i = 1, 2, \ldots, n, \\ &(2c) \quad f_0'''(b^-) = 0, \end{aligned} \tag{4.8}$$

where

$$\begin{aligned} f(c^+) &= \lim_{c \to h} f(h) \quad \text{for } c > h, \\ f(c^-) &= \lim_{c \to h} f(h) \quad \text{for } c < h. \end{aligned}$$

Property (2*b*)

The small distance $\epsilon > 0$ is such that $a < x_{i-1} < x_i - 2\epsilon < x_i < x_i + 2\epsilon < x_{i+1} < b$. Then the following h_{2b} is defined for proving $(2b)$:

$$\begin{aligned} &h_{2b} \in C^{(2)}[a, b], \\ &h_{2b} = \begin{cases} 1 & \text{for } x_i - \epsilon \leq x \leq x_i + \epsilon, \\ 0 & \text{for } x \leq x_i - 2\epsilon \text{ or } x \geq x_i + 2\epsilon. \end{cases} \end{aligned} \tag{4.9}$$

Figure 4.1 is an example of what h_{2b} could look like.

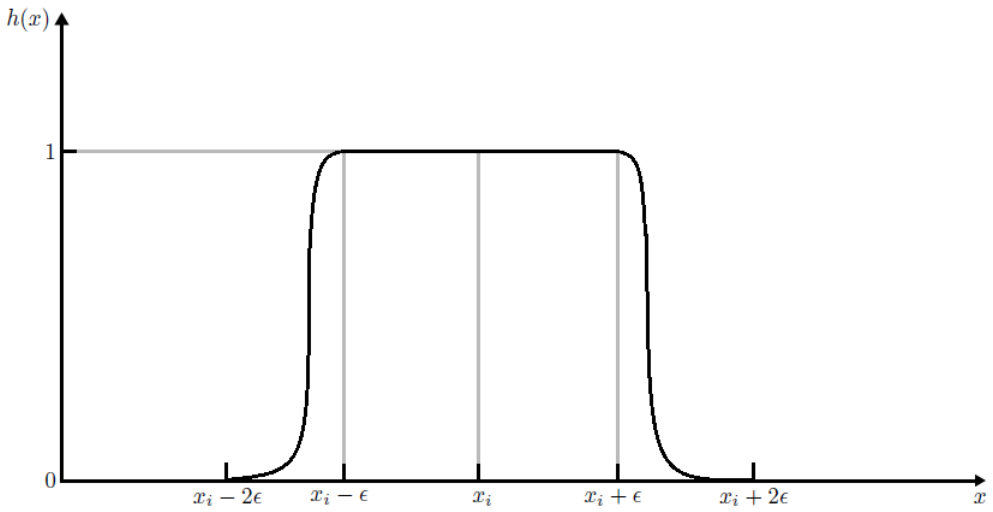

Figure 4.1: *Illustration of the function h_{2b}.*

Property $(2b)$ is shown using our chosen h_{2b} above, combined with property $(1a)$. As the starting point we will use our previously derived expression for optimal f_0, (3.18):

$$\int_a^b f_0''(x) h_{2b}''(x) dx + \lambda \sum_{j=1}^{n} (f_0(x_j) - y_j) h_{2b}(x_j) = 0.$$

If the function h_{2b} is inserted, the integral is separated and integrated by parts it yields:

$$f_0''(x)h_{2b}'(x)\Big|_{x_i-2\epsilon}^{x_i} - \int_{x_i-2\epsilon}^{x_i} f_0'''(x)h_{2b}'(x)dx$$
$$+f_0''(x)h_{2b}'(x)\Big|_{x_i}^{x_i+2\epsilon} - \int_{x_i}^{x_i+2\epsilon} f_0'''(x)h_{2b}'(x)dx + \lambda(f_0(x_i)-y_i)h_{2b}(x_i) = 0. \tag{4.10}$$

Inserting h_{2b} into the first and third term makes them sum up to zero, therefore (4.10) is reduced to:

$$-\int_{x_i-2\epsilon}^{x_i} f_0'''(x)h_{2b}'(x)dx - \int_{x_i}^{x_i+2\epsilon} f_0'''(x)h_{2b}'(x)dx + \lambda(f_0(x_i)-y_i)h_{2b}(x_i) = 0.$$

Integrating by parts once more and using property $(1a)$ gives:

$$-f_0'''(x_i^-)h_{2b}(x_i) + f_0'''(x_i-2\epsilon)h_{2b}(x_i-2\epsilon)$$
$$+f_0'''(x_i^+)h_{2b}(x_i) - f_0'''(x_i+2\epsilon)h_{2b}(x_i+2\epsilon) + \lambda(f_0(x_i)-y_i)h_{2b}(x_i) = 0.$$

Recalling the definition of h_{2b} from (4.9) and inserting it in the above expression gives the result of property $(2b)$, namely:

$$\boxed{f_0'''(x_i^+) - f_0'''(x_i^-) = -\lambda(f_0(x_i)-y_i)h_{2b}(x_i) \text{ for } i = 1,2,\ldots,n.} \tag{4.11}$$

This result can be interpreted graphically. The discontinuity of the third derivative of f_0 at x_i has the size of $\lambda \cdot h_{2b}(x_i)$ multiplied with the error of the approximation function at x_i.

Properties (2a) and (2c)

Property $(2a)$ is proven using one other carefully chosen function h_{2a}, together with property $(1a)$. First, introduce an $\epsilon > 0$ such that $a < a+\epsilon < a+2\epsilon \leq x_1 < x_2 < \ldots < x_n < b$, then choose h_{2a} such that:

$$h_{2a} \in C^{(2)}[a,b],$$
$$h_{2a} = \begin{cases} 1 & \text{for } a \geq x \geq a+\epsilon, \\ 0 & \text{for } x \geq a+2\epsilon. \end{cases} \tag{4.12}$$

The starting point will once again be (3.18):

$$\int_a^b f_0''(x)h_{2a}''(x)dx + \lambda\sum_{j=1}^{n}(f_0(x_j)-y_j)h_{2a}(x_j) = 0.$$

Observe that with our function h_{2a} specified as (4.12) the sum reduces to zero. Integration by parts two times then yields:

$$f_0''(a+2\epsilon)h_{2a}'(a+2\epsilon) - f_0''(a)h_{2a}'(a) - f_0'''(x)h_{2a}(x)\Big|_a^{a+2\epsilon} + \int_a^{a+2\epsilon} f_0^{(4)}(x)h_{2a}(x)dx = 0. \tag{4.13}$$

With the defined h_{2a} inserted into (4.13) the result of $(2a)$ is obtained:

$$\boxed{f_0'''(a^+) = 0.} \tag{4.14}$$

The calculation for property $(2c)$ is completely analogous. The only difference is that $\epsilon > 0$ now satisfies $a < x_1 < x_2 < \ldots < x_n \leq b - 2\epsilon < b - \epsilon < b$ and h_{2a} is defined as:

$$\begin{aligned} &h_{2c} \in C^{(2)}[a,b], \\ &h_{2c} = \begin{cases} 1 & \text{for } b - \epsilon \leq x \leq b, \\ 0 & \text{for } x \leq b - 2\epsilon. \end{cases} \end{aligned} \tag{4.15}$$

The result of $(2c)$ is:

$$\boxed{f_0'''(b^-) = 0.} \tag{4.16}$$

The graphical interpretation of $(2a)$ and $(2c)$ is that f_0 is a quadratic function at the end points a and b of the interval.

4.1.3 Third Property

The third property is given by the following equalities:

$$\begin{aligned} &(3a) \quad f_0''(a) = 0, \\ &(3b) \quad f_0''(b) = 0. \end{aligned} \tag{4.17}$$

Showing property $(3a)$ we introduce an $\epsilon > 0$ so that $a < a+\epsilon < a+2\epsilon \leq x_1 < x_2 < \ldots < x_n < b$. Then h_{3a} is defined as:

$$\begin{aligned} &h_{3a} \in C^{(2)}[a,b], \\ &h_{3a} = \begin{cases} 1 & \text{for } a \leq x \leq a + \epsilon, \\ 0 & \text{for } x \leq a \text{ and } x \geq a + 2\epsilon. \end{cases} \end{aligned} \tag{4.18}$$

Our starting point will also this time be (3.18):

$$\int_a^b f_0''(x) h_{3a}''(x) dx + \lambda \sum_{j=1}^{n} (f_0(x_j) - y_j) h_{3a}(x_j) = 0.$$

With the specified h_{3a} inserted and integration by parts carried out the above expression becomes:

$$f_0''(a+2\epsilon) h_{3a}'(a+2\epsilon) - f_0''(a) h_{3a}'(a) - \int_a^{a+2\epsilon} f_0'''(x) h_{3a}'(x) dx = 0. \tag{4.19}$$

Using (4.18), property $(1a)$ and integrating by parts, property $(3a)$ is obtained as a result:

$$\boxed{f_0''(a) = 0.} \tag{4.20}$$

The proof of $(3b)$ is completely analogous except for the definition of h_{3b}. Now $\epsilon > 0$ satisfies $a < x_1 < x_2 < \ldots < x_n \leq b - 2\epsilon < b - \epsilon < b$ and h_{3b} is defined as:

$$\begin{aligned} &h_{3b} \in C^{(2)}[a,b], \\ &h_{3b} = \begin{cases} 1 & \text{for } b - \epsilon \leq x \leq b\epsilon, \\ 0 & \text{for } x \leq b - 2\epsilon \text{ and } x = b. \end{cases} \end{aligned} \tag{4.21}$$

The result obtained in this case is:

$$\boxed{f_0''(b) = 0.} \tag{4.22}$$

The graphical interpretation of $(3a)$ and $(3b)$ is that the function f_0 is linear at the end points a and b of the interval.

4.2 Consequence of the Three Properties

In the last few parts of this report the following theorem has been proven.

Theorem 4.2.1 *If $f_0 \in C^{(2)}[a,b]$ and $F'(f_0) = \mathbf{0}$ then properties (1), (2) and (3) are satisfied.*

Recall that we want to find the function f_0 which satisfies $F'(f_0) = \mathbf{0}$. It has been shown that if $F'(f_0) = \mathbf{0}$ then properties (1), (2) and (3) are satisfied, but the converse implication has not yet been proven. The theorem and the proof that if the three properties are satisfied, then $F'(f_0) = \mathbf{0}$ will be given next.

Theorem 4.2.2 *If $f_0 \in C^{(2)}[a,b]$ and the following properties are satisfied:*

(1) *(a)* $f_0^{(4)}(x) = 0$ *for all* $x \in [a,b] \setminus \{a, x_1, x_2, \ldots, x_{n-1}, x_n, b\}$,

(2) *(a)* $f_0'''(a^+) = 0$,
(b) $f_0'''(x_i^+) - f_0'''(x_i^-) = -\lambda(f_0(x_i) - y_i)$ *for* $i = 1, 2, \ldots, n$,
(c) $f_0'''(b^-) = 0$,

(3) *(a)* $f_0''(a) = 0$,
(b) $f_0''(b) = 0$,

then

$$F'(f_0) = \mathbf{0}. \tag{4.23}$$

Proof The starting point will be the integral:

$$\int_a^b f_0''(x)h''(x)dx. \tag{4.24}$$

The integral is separated for each interval and then integrated by parts:

$$\begin{aligned}
&\int_a^{x_1} f_0''(x)h''(x)dx + \sum_{j=1}^{n-1}\left(\int_{x_j}^{x_{j+1}} f_0''(x)h''(x)dx\right) + \int_{x_n}^{b} f_0''(x)h''(x)dx \\
&= f_0''(x)h'(x)\Big|_a^{x_1} + \sum_{j=1}^{n-1}\left(f_0''(x)h'(x)\Big|_{x_j}^{x_{j+1}}\right) + f_0''(x)h'(x)\Big|_{x_n}^{b} \\
&- \int_a^{x_1} f_0'''(x)h'(x)dx - \sum_{j=1}^{n-1}\left(\int_{x_j}^{x_{j+1}} f_0'''(x)h'(x)dx\right) - \int_{x_n}^{b} f_0'''(x)h'(x)dx.
\end{aligned}$$

Due to $(3a)$ and $(3b)$ the expression is reduced to:

$$f_0''(x_1)h'(x_1) + \sum_{j=1}^{n-1}\Big(f_0''(x_{j+1})h'(x_{j+1}) - f_0''(x_j)h'(x_j)\Big) - f_0''(x_n)h'(x_n)$$
$$-\int_a^{x_1} f_0'''(x)h'(x)dx - \sum_{j=1}^{n-1}\Big(\int_{x_j}^{x_{j+1}} f_0'''(x)h'(x)dx\Big) - \int_{x_n}^{b} f_0'''(x)h'(x)dx.$$

Observe that the first three terms pairwise cancel out each other and therefore sum up to zero. Due to property $(1a)$ the expression becomes:

$$-f_0'''(x)h(x)\Big|_a^{x_1} - \sum_{j=1}^{n-1}\Big(f_0'''(x)h(x)\Big|_{x_j}^{x_{j+1}}\Big) - f_0'''(x)h(x)\Big|_{x_n}^{b}.$$

Inserting limits and using properties $(2a)$ and $(2c)$ we arrive at:

$$-f_0'''(x_1^-)h(x_1) - \sum_{j=1}^{n-1}\Big(f_0'''(x_{j+1}^-)h(x_{j+1}) - f_0'''(x_j^+)h(x_j)\Big) + f_0'''(x_n^+)h(x_n).$$

The first and third term are added to the sum and we can conclude from (4.24) that:

$$\int_a^b f_0''(x)h''(x)dx = \sum_{j=1}^{n}\Big(f_0'''(x_j^+) - f_0'''(x_j^-)\Big)h(x_j).$$

Rewriting the sum using $(2b)$ and multiplying by two yields:

$$2\int_a^b f_0''(x)h''(x)dx + 2\sum_{j=1}^{n}\Big(f_0(x_i) - y_i\Big)h(x_j) = 0. \tag{4.25}$$

This is precisely the equality we wanted to prove, namely that $\forall h \in C^{(2)}[a,b]$

$$(F'(f_0))(h) = 0 \tag{4.26}$$

or shorter,

$$F'(f_0) = \mathbf{0}. \tag{4.27}$$

Thereby the proof is complete. But as yet, we do not know whether or not a minimum of the cost function F has been found. This problem is taken care of in the next section.

4.3 Convexity of the Cost Function

We need to verify that the function $F : C^{(2)}[a,b] \longrightarrow \mathbb{R}$ is convex in order to conclude that a found extreme point is a minimum. Recalling from (2.4), F is given by:

$$F(f) = \int_a^b (f''(x))^2 dx + \lambda\sum_{i=1}^{n}(f(x_i) - y_i)^2, \text{ for } f \in C^{(2)}[a,b].$$

The function F is convex only if the maps

$$f \longmapsto \textstyle\int_a^b (f''(x))^2 dx \tag{4.28}$$
$$f \longmapsto \lambda(f(x_i) - y_i)^2 \tag{4.29}$$

for $i = 1, ..., n$ are convex.

Definition A set K in a linear vector space is said to be convex if, given $x_1,\ x_2 \in K$, all points of the form $tx_1 + (1-t)x_2$ with $0 \leq t \leq 1$ are in K.

This means that the line segment between any two points in a convex set is also in the set. As mentioned in section 4.3, the convexity of the function is defined as follows.

Definition Let X be a vector space. A mapping $G : X \to \mathbb{R}$ is then said to be convex if $G(tx_1 + (1-t)x_2) \leq tG(x_1) + (1-t)G(x_2)$ for all $x_1,\ x_2 \in X$ and all $t,\ 0 \leq t \leq 1$.

First the convexity of (4.28) is verified. In our case $X = \mathbb{R}$, which is a convex set. Using previous definition of a convex mapping, if $f_1, f_2 \in C^{(2)}[a,b]$, then for $t \in [0,1]$ it can be shown that

$$\int_a^b ((tf_1 + (1-t)f_2)''(x))^2 dx \leq t\int_a^b (f_1''(x))^2 dx + (1-t)\int_a^b (f_2''(x))^2 dx \tag{4.30}$$

and thus the map $f \longmapsto \int_a^b (f''(x))^2 dx$ is convex.

Continuing with checking the convexity of (4.29) we fix $i \in 1, ..., n$ and once again use the definition for convex mappings:

$$\lambda((tf_1 + (1-t)f_2)(x_i) - y_i)^2 \leq t(\lambda(f_1(x_i) - y_i)^2) + (1-t)(\lambda f_2(x_i) - y_i)^2). \tag{4.31}$$

Thus the maps, for all $i = 1, ..., n$,

$$f \longmapsto \lambda(f(x_i) - y_i)^2 : C^{(2)}[a,b] \longrightarrow \mathbb{R}$$

are convex as well.

Since all of the maps are convex, F is also convex.

4.4 Consequences of Theorem 4.2.2

The significance of Theorem 4.2.2 for this optimization problem can not be understated. We already know as a result from section 4.3 that the cost function F is a convex function. The convexity is enough to conclude that as soon as a function f_0 is found satisfying the above three properties we know for sure that this function is the optimal solution to the problem (P). Hereby, the problem has changed from finding the function which best interpolates certain points to finding the sum of cubic functions satisfying properties (1), (2) and (3).

Chapter 5

Construction of Solution

This chapter is about solving the problem numerically using the theory from the analytical calculations. In the previous chapter it was shown that if F satisfies the three properties, the function F is differentiable and $F'(f_0) = \mathbf{0}$. One could also see that the function is convex which means that Theorem 2.3.2 is satisfied. This shows that f_0 is an optimal solution to the problem (P), and it is now possible to compute a numerical solution.

5.1 The Numerical Problem

We need f_0 to be a piecewise cubic function and denote these cubic functions, also known as splines, as P_k:

$$P_k(x) = A_k(x - x_k)^3 + B_k(x - x_k)^2 + C_k(x - x_k) + D_k$$

on the interval $[x_k, x_{k+1}]$ for $k = 0, ..., n$.

Recall from Theorem 4.2.2 that we want to construct f_0 such that it satisfies the following conditions:

(1) (a) $f_0^{(4)}(x) = 0$ for all $x \in [a, b] \setminus \{a, x_1, x_2, \ldots, x_{n-1}, x_n, b\}$,

(2) (a) $f_0'''(a^+) = 0$,
(b) $f_0'''(x_i^+) - f_0'''(x_i^-) = -\lambda(f_0(x_i) - y_i)$ for $i = 1, 2, \ldots, n$,
(c) $f_0'''(b^-) = 0$,

(3) (a) $f_0''(a) = 0$,
(b) $f_0''(b) = 0$.

Using that f_0 should be twice differentiable and condition $(2b)$ yields

$$\begin{cases} P_k(x_{k+1}) = P_{k+1}(x_{k+1}) & (f_0 \text{ continuous}), \\ P_k'(x_{k+1}) = P_{k+1}'(x_{k+1}) & (f_0' \text{ continuous}), \\ P_k''(x_{k+1}) = P_{k+1}''(x_{k+1}) & (f_0'' \text{ continuous}), \\ P_k'''(x_{k+1}) = P_{k+1}'''(x_{k+1}) + \lambda(P_{k+1}(x_{k+1}) - y_{k+1}) & (\text{using (2b) above}), \end{cases} \tag{5.1}$$

for $k = 1, ..., n-1$.

The function P_k and its derivatives are defined as

$$\begin{cases} P_k(x) = A_k(x-x_k)^3 + B_k(x-x_k)^2 + C_k(x-x_k) + D_k, \\ P_k'(x) = 3A_k(x-x_k)^2 + 2B_k(x-x_k) + C_k, \\ P_k''(x) = 6A_k(x-x_k) + 2B_k, \\ P_k'''(x) = 6A_k, \end{cases} \tag{5.2}$$

for $k = 1, ..., n-1$.

Inserting (5.2) into (5.1) gives

$$\begin{cases} A_k(x_{k+1}-x_k)^3 + B_k(x_{k+1}-x_k)^2 + C_k(x_{k+1}-x_k) + D_k = D_{k+1}, \\ 3A_k(x_{k+1}-x_k)^2 + 2B_k(x_{k+1}-x_k) + C_k = C_{k+1}, \\ 6A_k(x_{k+1}-x_k) + 2B_k = 2B_{k+1}, \\ 6A_k = 6A_{k+1} + \lambda(D_{k+1} - y_{k+1}), \end{cases} \tag{5.3}$$

for $k = 1, ..., n-1$.

Using the conditions $(2a)$, $(2c)$, $(3a)$ and $(3b)$ we get the following:

$$\begin{cases} f_0'''(a_+) = 0 \\ f_0''(a) = 0 \end{cases} \Rightarrow \begin{cases} P_0'''(a) = 0 \Rightarrow A_0 = 0, \\ P_0''(a) = 0 \Rightarrow 2B_0 = 0, \end{cases} \tag{5.4}$$

$$\begin{cases} f_0'''(b_-) = 0 \\ f_0''(b) = 0 \end{cases} \Rightarrow \begin{cases} P_0'''(b) = 0 \Rightarrow A_n = 0, \\ P_0''(b) = 0 \Rightarrow B_n = 0. \end{cases} \tag{5.5}$$

Using (5.3), (5.4) and (5.5) gives us the mathematical equations that will be used in our MATLAB program later on:

$$\underbrace{\begin{pmatrix} 1 & (x_{k+1}-x_k) & (x_{k+1}-x_k)^2 & (x_{k+1}-x_k)^3 \\ 0 & 1 & 2(x_{k+1}-x_k) & 3(x_{k+1}-x_k)^2 \\ 0 & 0 & 1 & 3(x_{k+1}-x_k) \\ 0 & 0 & 0 & 6 \end{pmatrix}}_{=:H_k} \begin{pmatrix} D_k \\ C_k \\ B_k \\ A_k \end{pmatrix}$$

$$+ \underbrace{\begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ -\lambda & 0 & 0 & -6 \end{pmatrix}}_{=:M} \begin{pmatrix} D_{k+1} \\ C_{k+1} \\ B_{k+1} \\ A_{k+1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -\lambda y_{k+1} \end{pmatrix}$$

for $k = 1, ..., n-2$.

For the special case when $k = 0$ at the starting point of the interval we get:

$$\underbrace{\begin{pmatrix} 1 & x_1 - a \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}}_{=:H_0} \begin{pmatrix} D_0 \\ C_0 \end{pmatrix} + \underbrace{\begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ -\lambda & 0 & 0 & -6 \end{pmatrix}}_{=:M} \begin{pmatrix} D_1 \\ C_1 \\ B_1 \\ A_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -\lambda y_{k+1} \end{pmatrix}.$$

For the end point when $k = n-1$ we get:

$$\underbrace{\begin{pmatrix} 1 & (x_n - x_{n-1}) & (x_n - x_{n-1})^2 & (x_n - x_{n-1})^3 \\ 0 & 1 & 2(x_n - x_{n-1}) & 3(x_n - x_{n-1})^2 \\ 0 & 0 & 1 & 3(x_n - x_{n-1}) \\ 0 & 0 & 0 & 6 \end{pmatrix}}_{=:H_{n-1}} \begin{pmatrix} D_{n-1} \\ C_{n-1} \\ B_{n-1} \\ A_{n-1} \end{pmatrix} +$$

$$+ \underbrace{\begin{pmatrix} -1 & 0 \\ 0 & -1 \\ 0 & 0 \\ -\lambda & 0 \end{pmatrix}}_{=:N} \begin{pmatrix} D_n \\ C_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -\lambda y_n \end{pmatrix}.$$

Collecting the system of equations yields one big system of equations. The matrix Λ is defined as follows:

$$\Lambda := \begin{pmatrix} H_0 & M & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & H_1 & M & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & H_2 & M & 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & \cdots & 0 & H_{n-3} & M & 0 & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & H_{n-2} & M & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & H_{n-1} & N \end{pmatrix}. \tag{5.6}$$

Since the dimension of H_0 and N is 4×2 and the dimension for M and H_k of $k = 1, ..., n-1$ is 4×4, the dimension for Λ will be $4n \times 4n$.

The vectors X and Y are defined as:

$$X := \begin{pmatrix} D_0 \\ C_0 \\ \hline D_1 \\ C_1 \\ B_1 \\ A_1 \\ \hline D_2 \\ C_2 \\ B_2 \\ A_2 \\ \hline \vdots \\ \hline D_{n-1} \\ C_{n-1} \\ B_{n-1} \\ A_{n-1} \\ \hline D_n \\ C_n \end{pmatrix}, \quad Y := \begin{pmatrix} 0 \\ 0 \\ 0 \\ -\lambda y_1 \\ \hline 0 \\ 0 \\ 0 \\ -\lambda y_2 \\ \hline \vdots \\ \hline 0 \\ 0 \\ 0 \\ -\lambda y_{n-1} \\ \hline 0 \\ 0 \\ 0 \\ -\lambda y_n \end{pmatrix}.$$

Now the system of equations can be written as $\Lambda X = Y$, and a MATLAB program can be constructed to solve our problem (P).

5.2 Uniqueness of the Solution

We want to investigate whether the solution is unique and therefore we make a claim, that if a solution exists it is unique. This can be proven using contradiction. Suppose that there are two optimal solutions $f_1, f_2 \in C^{(2)}[a,b]$. For $t \in \mathbb{R}$, consider $\varphi : \mathbb{R} \to \mathbb{R}$ given by

$$\begin{aligned}
\varphi(t) &= F((1-t)f_1 + tf_2) \\
&= \int_a^b ((1-t)f_1''(x) + tf_2''(x))^2 dx + \lambda \sum_{i=1}^{n} ((1-t)f_1(x_i) + tf_2(x_i) - y_i)^2 \\
&= \int_a^b ((1-t)f_1''(x) + tf_2''(x))^2 dx + \lambda \sum_{i=1}^{n} (f_1(x_i) + t(f_2(x_i) - f_1(x_i)) - y_i)^2 \\
&= At^2 + Bt + C \qquad (5.7)
\end{aligned}$$

where

$$A = \int_a^b (f_2''(x) - f_1''(x))^2 dx + \lambda \sum_{i=1}^{n} (f_2(x_i) - f_1(x_i))^2. \qquad (5.8)$$

If $A = 0$, then $(f_2 - f_1)''(x) = 0 \ \forall x \in [a,b]$ because of (2.7). Hence, integrating two times gives:

$$(f_2 - f_1)(x) = \alpha x + \beta \Rightarrow f_2(x) = f_1(x) + \alpha x + \beta. \qquad (5.9)$$

Also, suppose $\lambda > 0$, then $\lambda \sum_{i=1}^{n} (f_2(x_i) - f_1(x_i))^2 = 0$ gives $\sum_{i=1}^{n} (\alpha x_i + \beta)^2 = 0$:

$$\begin{pmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}.$$

If $n > 2$, then $\alpha = \beta = 0$. Inserting this into (5.9) gives $f_1 = f_2$, which is a contradiction. The conclusion is that if $\lambda > 0$ and $n \geq 2$, then $A > 0$ and we would get the contradiction that for some value of $t \in (0,1)$, $(1-t)f_1 + tf_2$ gives a cost smaller than that of f_1 or f_2. Therefore the solution has to be unique if $\lambda > 0$ and $n \geq 2$.

What happens if one of these two conditions are violated? If $\lambda = 0$, any f of the form $f(x) = \alpha x + \beta$ where $x \in [a,b]$ is an optimal solution, since the corresponding cost is zero. For $n < 2$ any straight line passing through (x_1, y_1) is optimal.

Chapter 6

Results

6.1 Graphical Solutions of the Problem (P)

In the illustrations presented in this chapter, results from the MATLAB program will be shown for some different chosen vectors x and y. The program has been set to use the interval $x \in [0, 1]$ in the examples and the MATLAB code used can be found in Appendix A. Observe that for $\lambda = 0$ there exist infinitely many solutions. This because no consideration is taken to the interpolation error, meaning that any straight line is an optimal solution to the problem.

6.1.1 Example 1: Varying the Interpolation Parameter

$$x = \begin{pmatrix} 0.1 \\ 0.3 \\ 0.5 \\ 0.6 \\ 0.9 \end{pmatrix} \qquad y = \begin{pmatrix} 3 \\ 6 \\ 7 \\ 2 \\ 5 \end{pmatrix}$$

In Figure 6.1 we can see how different values of the interpolation parameter λ effects the curve fitting. Observe that for a small λ the curve is very smooth but has a large interpolation error.

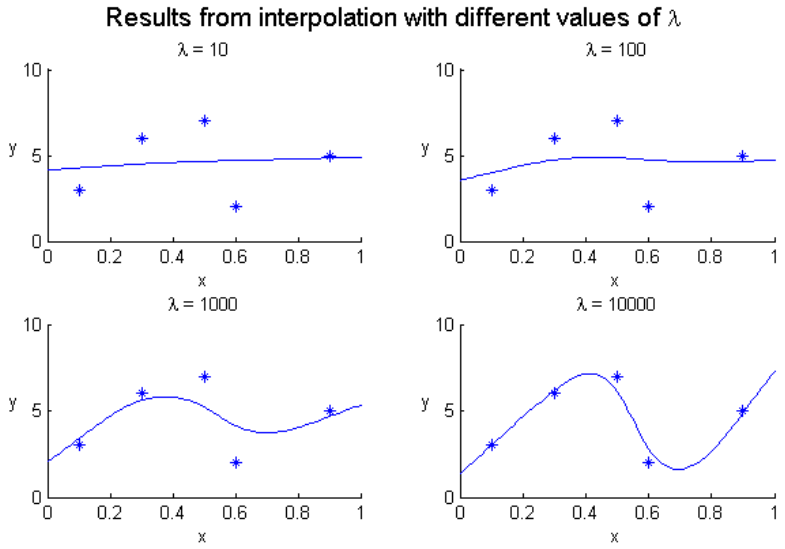

Figure 6.1: *The figure shows results of the curve fitting with different values of* λ.

When increasing λ the curve becomes less smooth, but the interpolation error decreases. The choice of λ is subjective, in some experiments a smoother curve is preferable and in others it is more important with a small interpolation error. In Figure 6.1 a relatively smooth curve with a small interpolation error is seen at $\lambda = 10^4$.

6.1.2 Example 2: Closely Placed Points

In this example one element in vector x is changed and it is deliberately chosen so that two points are close to each other on the x-axis.

$$x = \begin{pmatrix} 0.1 \\ 0.3 \\ 0.5 \\ 0.51 \\ 0.9 \end{pmatrix} \qquad y = \begin{pmatrix} 3 \\ 6 \\ 7 \\ 2 \\ 5 \end{pmatrix}$$

In Figure 6.2 it is harder to determine a good value of λ, since it depends on how smooth you want the function to be. Observe that a much larger value of λ is needed to decrease the interpolation error to the same levels as in the first example.

For $\lambda = 10^6, 10^7$ and 10^8 the curve takes a very low minimum value. In this example it is very important to define the purpose of the interpolation, and decide whether to accept very large or small values of y.

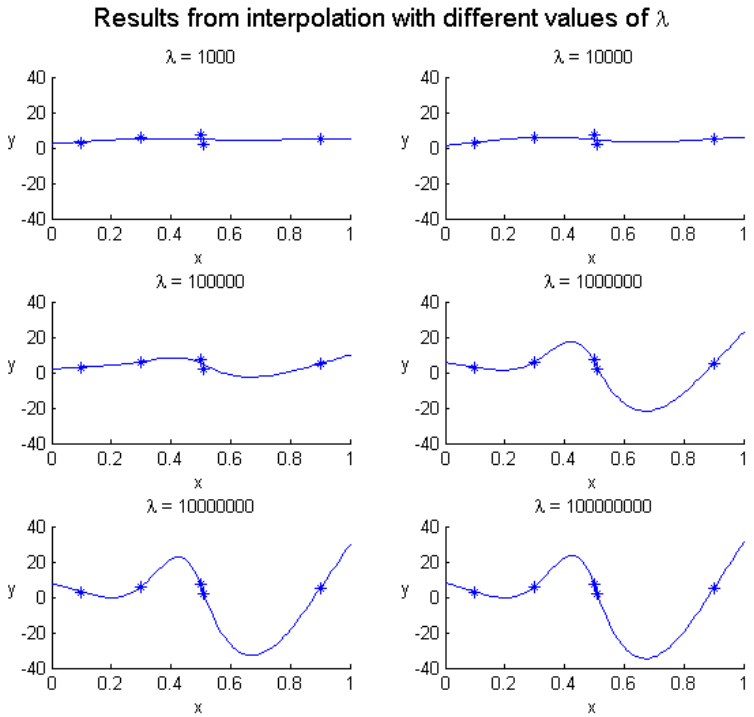

Figure 6.2: *The figure shows results of the curve fitting with different values of λ when two points are chosen very close to each other.*

6.1.3 Example 3: Increased Number of Points

In this example the number of points are increased from 5 to 10.

$$x = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.3 \\ 0.35 \\ 0.5 \\ 0.6 \\ 0.8 \\ 0.85 \\ 0.9 \\ 0.95 \end{pmatrix} \qquad y = \begin{pmatrix} 3 \\ 6 \\ 7 \\ 2 \\ 5 \\ 1 \\ 4 \\ 8 \\ 3 \\ 7 \end{pmatrix}$$

In Figure 6.3 the number of coordinate points is doubled in comparison to the previous examples. In this example a large value of λ is needed to get a good fit. For $\lambda = 10^6$ the curve fits relatively good through the points. A conclusion from the examples is the difficulty of finding a perfect value of λ. In next section a method for finding λ will be presented.

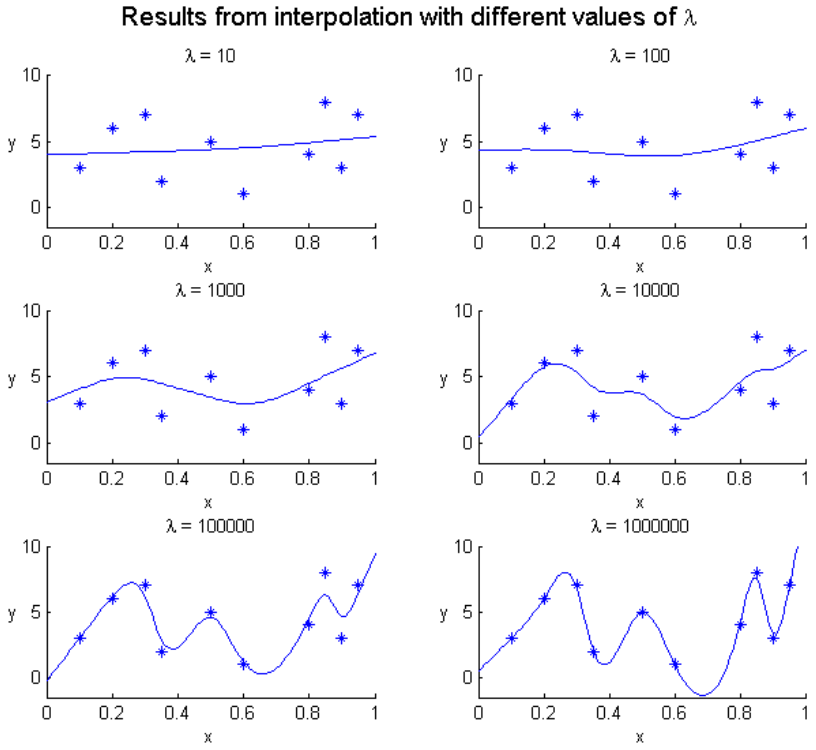

Figure 6.3: *The figure shows results of the curve fitting with different values of λ for many points.*

6.2 Optimal Value of the Interpolation Parameter

The choice of the interpolation parameter λ is very important. There exist different methods for estimating an optimal value of λ. To expand our project we will estimate the interpolation parameter, by using Ordinary Cross Validation, OCV.[5] This method is also called the "leaving-out-one"- method. In the book *Spline Models for Observational Data*, the cost function is given on the form:

$$F_w = \frac{1}{n}\sum_{i=1,k\neq i}^{n}(f(x_i)-y_i)^2 + \lambda_w\int_a^b (f''(x))^2dx. \tag{6.1}$$

There are two different summations in (6.1). The first one over all $i = 1,\dots,n$ except when $i = k$ and secondly one over all $k = 1,\dots,n$. This shows that the method always leaves one point out. Further, if $f_{\lambda_w}^{[k]}$ is a minimizer of the cost function $F_w(f)$, then the OCV-function is defined as:

$$V_0(\lambda_w) = \frac{1}{n}\sum_{k=1}^{n}\left(y_k - f_{\lambda_w}^{[k]}(x_k)\right)^2. \tag{6.2}$$

This means that $\hat{\lambda}_w$ is the parameter that minimizes V_0. Now let us recall the expression of our cost function F from (2.4) and observe that it differs from (6.1):

$$F = \lambda\sum_{i=1}^{n}(f(x_i)-y_i)^2 + \int_a^b (f''(x))^2dx.$$

One result of optimization is that the optimal solution $\hat{x}$ of the cost function will not change if it is multiplied by a positive scalar. Neither will it change if a scalar is added to the cost function. If $\hat{f}$ is the optimal solution minimizing F, then obviously $F(\hat{f}) \leq F(f) \quad \forall f$. Then if $\alpha > 0$ is a scalar and $c \in \mathbb{R}$, $\alpha F(\hat{f}) + c \leq \alpha F(f) + c \quad \forall f$. If we compute F_w/λ_w this still has the same optimal solution as F_w, given that λ_w is positive. Term wise identification then implies that if

$$\lambda = \frac{1}{\lambda_w n} \tag{6.3}$$

then F and F_w have the same optimal solution. Since the MATLAB program is created for solving the problem (P), some modifications using the relation (6.3) were needed in order to fit the OCV to our problem formulation.

6.2.1 Results using OCV

In the article [5], the OCV method of estimating λ is applied to a certain set of points positioned in a special way. The value of the function $f(x) = 4.26e^{-x} - 4e^{-2x} + 3e^{-3x}$ is determined at a certain number of x-values and thereafter a normal distributed error with expectation value zero is added to position the points. Hereby the points are positioned not exactly on the function curve but rather a small distance from it. In Figure 6.4 the function is plotted on the interval $[0, 3]$, with 70 random points positioned using an error with variance 0.3. The MATLAB code for the program can be read in Appendix A.

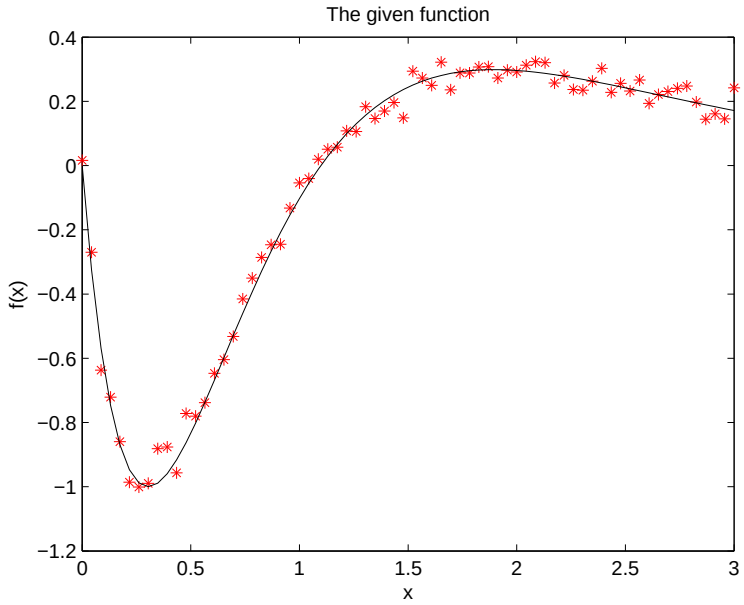

Figure 6.4: *The function* $f(x) = 4.26e^{-x} - 4e^{-2x} + 3e^{-3x}$ *and randomly generated points.*

The variance of the errors are small enough to visibly make a pretty good approximation of the function. We could therefore expect that OCV would produce a λ value giving a reasonably smooth approximation function. The result is seen in Figure 6.5.

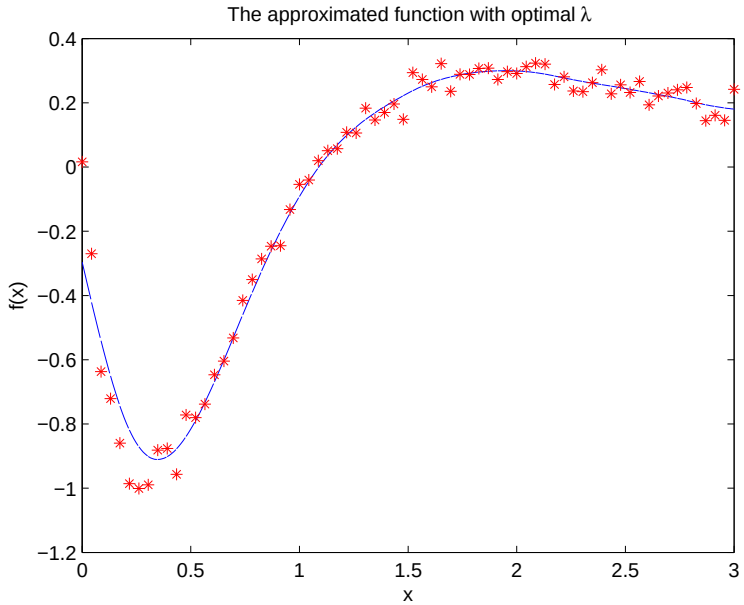

Figure 6.5: *The approximated function using OCV and randomly generated points.*

Figure 6.6 shows the OCV function $V_0(\lambda_w)$ for the random points in a log-log-plot. We can clearly see how the function comes to a minimum at approximately $\lambda_w = 10^{-2}$. This indicates that the optimal λ value in the OCV-sense has been found.

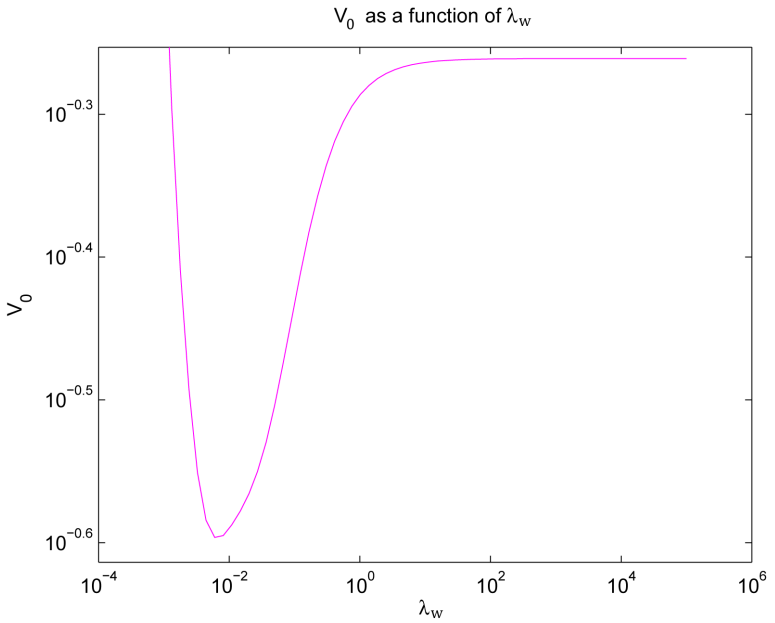

Figure 6.6: *The cross validation function $V_0(\lambda_w)$ on the interval $x \in [0, 3]$.*

The OCV method does not always give a good estimate of λ. If we use the same function from the previous example and only change the interval from $[0, 3]$ to $[3, 6]$, Figure 6.7 of $V_0(\lambda_w)$ is obtained.

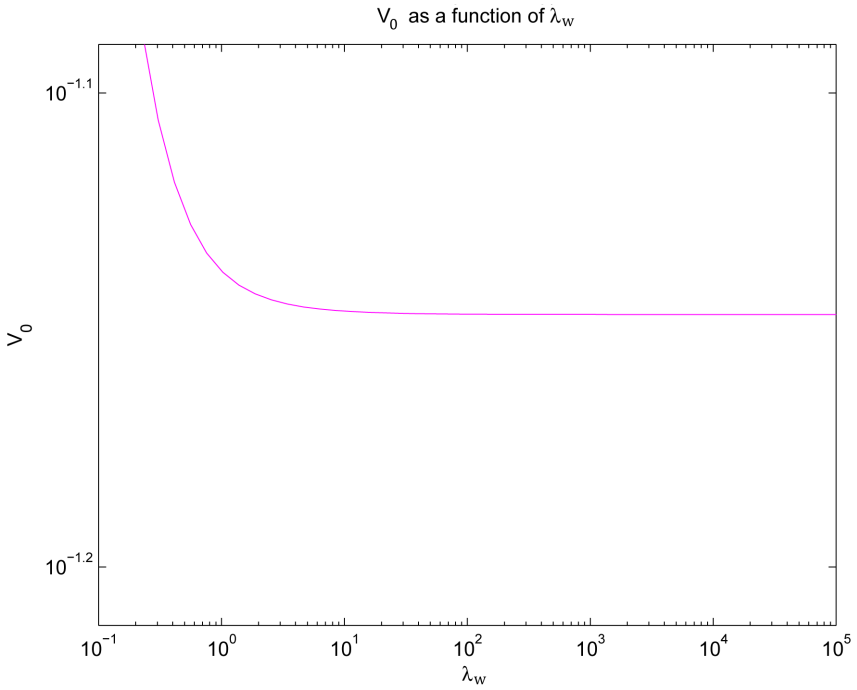

Figure 6.7: *The cross validation function $V_0(\lambda_w)$ on the interval $x \in [3, 6]$.*

The interpretation of Figure 6.7 is that V_0 has no minimum value for λ_w on the interval $[10^{-4}, 10^6]$. So the value of V_0 decreases as long as λ_w increases. This means that $\lambda = 1/(\lambda_w \cdot n)$ takes the minimum value on the interval, which will be of order 10^{-5}. The smoothness term will dominate the cost function and the solution to the problem will almost be a straight line.

Obviously Ordinary Cross Validation does not work for all problems, so it should not be used without consideration for estimating λ. Nevertheless if a minimum of V_0 exists, then OCV seems to give a fairly good estimate of the optimal value of the interpolation parameter λ.

Chapter 7

Discussion & Conclusions

7.1 Error Analysis

As always, when doing numerical models there is a loss of accuracy due to cancellation of significant digits. In our problem one concern is the condition number of the matrix Λ. The condition number increases as Λ becomes larger, which leads to a greater loss of significant digits. For the examples given in this report, the error can be neglected for two reasons. The first reason is that the λ values used are not large enough to cause an extremely high condition number. And secondly, the values used in the calculation are not needed to be of extremely high accuracy. A problem could occur if one wants to fit a curve with extremely small interpolation error using data points with many significant numbers. In that case the program would come to a scenario where it can not produce a better fit, due to the problem of cancellation. In real applications, this problem is probably very rare.

7.2 Conclusions

The first result obtained in this thesis was that the derivative of the cost function F is a linear and continuous mapping. This allowed setting the derivative of the cost function equal to zero. Since F was shown to be a convex function, this method was obviously rewarding. The reason to this is briefly that a convex function takes a minimum value only where the derivative equals zero. Eventually it was proven that for the derivative to be zero, it was enough that three properties were satisfied by the function f. It would be interesting to investigate if there exist any other properties that could be used. It is difficult to define a function and then find properties for it to satisfy. This makes it hard for us to construct other properties and functions that could be used instead.

From the three properties, a solution to the optimization problem stated as (P) was created in the form of a MATLAB program. The results obtained by the program showed different characteristics. Furthermore, the interpolation parameter λ could be examined in greater detail for a more general conclusion to be made about it. It would be a good idea to explore other mathematical techniques like General Cross Validation (GCV) for estimation of λ, to see under what circumstances they work well. Our main observation concerning the interpolation parameter is that it seems to be very hard to estimate it well for an arbitrary set of data. This because it is hard to construct a method that suits all purposes. For example, in statistical studies smoother curves are often preferred while a smaller interpolation error often is required in path planning. For this reason a general method to compute the optimal interpolation parameter for arbitrary

data points probably does not even exist.

The optimization problem examined by this thesis has yet some unexplored paths. Imagine a situation where you have data collected with different accuracy so that some data points are considered more important when fitting the curve to the data. In this case the data points can be weighted so that some points are considered more significant. Points could even be given an infinite weight so that the function would have to pass through them.

7.3 Alternative Methods to Cubic Splines

In this report cubic splines have been used for curve fitting because it simplified the construction of a numerical method, to solve the problem. There exist several other methods for solving curve fitting problems using calculus. Lagrange polynomial interpolation, linear and nonlinear regression are some of the more common methods used. It would interesting to examine the other methods more carefully and compare the results. The comparison could be used to highlight the advantages and the possible disadvantages of the cubic spline method.

Polynomial interpolation produces a polynomial curve of order n, that goes through all data points. As the number of data points increases, attention has to be paid to that the curve oscillates more. When instead using cubic splines the maximal order of the polynomials are already decided, which will prevent oscillations.

Linear and nonlinear regression take the standard deviation in consideration when fitting a curve to the data points. Consequently, different sets of data can give rise to the same optimal curve. Therefore the fitting depends more on the placement than the number of points. Without knowing more precisely what kind of function you are looking for, this makes a relatively bad approximation. For a random set of data, where a trend can't be observed, it is better to use cubic splines which will provide an optimal curve.

Bibliography

[1] Adams, Robert. *Calculus: A Complete Course*. Pearson Education. (2010).

[2] Arlinghaus, Sandra. *Practical handbook of Curve Fitting*. CRC Press, Inc. (1994).

[3] Egerstedt, Magnus, et al. *Control Theoretic Splines, Optimal Control, Statistics and Path Planning*. Princeton University Press (2010).

[4] Milne, William. *Numerical Calculus, Aproximations, Interpolation, Finite Differences, Numerical Integration and Curve Fitting*. Princeton University Press (1949).

[5] Wahba, Grace. *Spline Models for Observational Data*. Society for Industrial & Applied Mathematics (1990).

[6] Rudin, Walter. *Principles of mathematical analysis*. McGraw-Hill Higher Education. (1976).

Appendix A

MATLAB Code

Main Program for Computing the Solution

```
clear all, close all, clc

a = input('Please write an a: ');
b = input('Please write a b: ');
n = input('Write in number of points: ');
lambda = input('Write in value of lambda: ');
disp('Write in coordinate points:')

x=zeros(n);
y=zeros(n);

for i = 1:n
    x(i) = input('x: ');
    y(i) = input('y: ');
end

M = [-1 0 0 0
    0 -1 0 0
    0 0 -1 0
    -lambda 0 0 -6];
N = [-1 0
    0 -1
    0 0
    -lambda 0];
H0 = [1 x(1)-a
    0 1
    0 0
    0 0];

Lambda = zeros(4*n,4*n);
Lambda(1:4,1:2)=H0;
Lambda( 1:4,3:6)=M;

for i=(1:n-2)
   H=[1 (x(i+1)-x(i)) (x(i+1)-x(i))^2 (x(i+1)-x(i))^3
    0 1 2*(x(i+1)-x(i)) 3*(x(i+1)-x(i))^2
    0 0 1 3*(x(i+1)-x(i))
    0 0 0 6];
   Lambda(4*(i-1)+5:4*(i-1)+8,4*(i-1)+7:4*(i-1)+10)=M;
   Lambda(4*(i-1)+5:4*(i-1)+8,4*(i-1)+3:4*(i-1)+6)=H;
```

```
end

Hn_1 = [1 (x(n)-x(n-1)) (x(n)-x(n-1))^2 (x(n)-x(n-1))^3
        0 1 2*(x(n)-x(n-1)) 3*(x(n)-x(n-1))^2
        0 0 1 3*(x(n)-x(n-1))
        0 0 0 6];
Lambda(4*n-3:4*n,4*n-5:4*n-2)=Hn_1;
Lambda(4*n-3:4*n,4*n-1:4*n)=N;

Y=[];

for i=1:n
    vektor = [0 0 0 -lambda*y(i)]';
    Y = [Y
        vektor];
end

X = Lambda\Y

xplot=a:0.01:x(1);

p_0 = X(1) + X(2)*(xplot-a);
plot(xplot,p_0)
hold on

for i=(1:n-1)
    xplot=x(i):0.01:x(i+1);
    p=X(4*(i-1)+3)+X(4*(i-1)+4)*(xplot-x(i))+X(4*(i-1)+5)*(xplot-x(i)).^2
    +X(4*(i-1)+6)*(xplot-x(i)).^3;
    plot(xplot,p)
    hold on
end

xplot=x(n):0.01:b;

p_n = X(4*n-1) + X(4*n)*(xplot-x(n));
plot(xplot,p_n)
hold on

plot(x,y,'*')
```

Program for Finding λ Using OCV

```
clear all, close all, clc

global a b n x y steps

a = input('Please write an a: ');
b = input('Please write a b: ');
n = input('Write in number of points: ');

x=zeros(n,1);
y=zeros(n,1);
```

```
E=0;
V=0.03;

x = linspace(a,b,n)';
y_func = 4.26*(exp(-x) - 4*exp(-2*x) + 3*exp(-3*x));
y_rand = normrnd(E,V,n,1);
y = y_func + y_rand;

figure(1)
plot(x,y,'r*')
hold on
plot(x,y_func,'black')
title('The given function')
xlabel('x')
ylabel('f(x)')

steps = 100;

global a b n x y steps;

xlambda_w = logspace(-8,5,steps);
lam_min = 1;

OCV_sum = [];
for lam = 1:steps
    lambda_w = xlambda_w(lam);
    sum = 0;
    for k = 2:n-1
        y_k = [y(1:k-1); y(k+1:end)];
        p = func_find_lambda(x,y,k,lambda_w);
        sum = sum + (y_k(k) - p(k))^2;
    end
    sum = sum/(n-2);
    OCV_sum = [OCV_sum, sum];
    if sum == min(OCV_sum)
        lam_min = lam;
    end
end

lambda_w_min = xlambda_w(lam_min);
disp(min(OCV_sum))
disp(OCV_sum(lam_min))

figure(2)
loglog(xlambda_w,OCV_sum,'m')
ylim([min(OCV_sum)-0.01 OCV_sum(end)+0.01]);
title('V_O as a function of \lambda')
xlabel('\lambda')
ylabel('V_0')

plot_splines(lambda_w_min)
```

Function for Computing the Solution

```
function [p] = func_find_lambda(x,y,k,lambda_w)

    global a b n

    lambda = 1/(lambda_w*n);

    x = [x(1:k-1); x(k+1:n)];
    y = [y(1:k-1); y(k+1:n)];

    M = [-1 0 0 0
         0 -1 0 0
         0 0 -1 0
         -lambda 0 0 -6];

    N = [-1 0
         0 -1
         0 0
         -lambda 0];

    H0 = [1 x(1)-a
         0 1
         0 0
         0 0];

    Lambda = zeros(4*(n-1),4*(n-1));
    Lambda(1:4,1:2)=H0;
    Lambda(1:4,3:6)=M;

    for i=(1:n-3)
        H=[1 (x(i+1)-x(i)) (x(i+1)-x(i))^2 (x(i+1)-x(i))^3
            0 1 2*(x(i+1)-x(i)) 3*(x(i+1)-x(i))^2
            0 0 1 3*(x(i+1)-x(i))
            0 0 0 6];
        Lambda(4*(i-1)+5:4*(i-1)+8,4*(i-1)+7:4*(i-1)+10)=M;
        Lambda(4*(i-1)+5:4*(i-1)+8,4*(i-1)+3:4*(i-1)+6)=H;
    end

    Hn_1 = [1 (x(end)-x(end-1)) (x(end)-x(end-1))^2 (x(end)-x(end-1))^3
         0 1 2*(x(end)-x(end-1)) 3*(x(end)-x(end-1))^2
         0 0 1 3*(x(end)-x(end-1))
         0 0 0 6];
    Lambda(4*(n-1)-3:4*(n-1),4*(n-1)-5:4*(n-1)-2)=Hn_1;
    Lambda(4*(n-1)-3:4*(n-1),4*(n-1)-1:4*(n-1))=N;

    Y=[];

    for i=1:n-1
        vektor = [0 0 0 -lambda*y(i)]';
        Y = [Y
            vektor];
    end

    X = Lambda\Y;

    p=[];
    for i = 1:n-2
        p = [p X(4*(i-1)+3)+X(4*(i-1)+4)*x(i)+X(4*(i-1)+5)*x(i).^2+X(4*(i-1)+6)*x(i).^3];
```

```
    end
    p = [p X(4*(n-2)+3)+X(4*(n-2)+4)*x(n-1)];

end
```

Function for Plotting the Spline

```
function [] = plot_splines(lambda_w)

    global a b n x y

    lambda = 1/(lambda_w)

    M = [-1 0 0 0
         0 -1 0 0
         0 0 -1 0
         -lambda 0 0 -6];
    N = [-1 0
         0 -1
         0 0
         -lambda 0];
    H0 = [1 x(1)-a
         0 1
         0 0
         0 0];

    Lambda = zeros(4*n,4*n);
    Lambda(1:4,1:2)=H0;
    Lambda( 1:4,3:6)=M;

    for i=(1:n-2)
       H=[1 (x(i+1)-x(i)) (x(i+1)-x(i))^2 (x(i+1)-x(i))^3
        0 1 2*(x(i+1)-x(i)) 3*(x(i+1)-x(i))^2
        0 0 1 3*(x(i+1)-x(i))
        0 0 0 6];
       Lambda(4*(i-1)+5:4*(i-1)+8,4*(i-1)+7:4*(i-1)+10)=M;
       Lambda(4*(i-1)+5:4*(i-1)+8,4*(i-1)+3:4*(i-1)+6)=H;
    end

    Hn_1 = [1 (x(n)-x(n-1)) (x(n)-x(n-1))^2 (x(n)-x(n-1))^3
            0 1 2*(x(n)-x(n-1)) 3*(x(n)-x(n-1))^2
            0 0 1 3*(x(n)-x(n-1))
            0 0 0 6];
    Lambda(4*n-3:4*n,4*n-5:4*n-2)=Hn_1;
    Lambda(4*n-3:4*n,4*n-1:4*n)=N;


    Y=[];

    for i=1:n
        vektor = [0 0 0 -lambda*y(i)]';
        Y = [Y
            vektor];
```

```
    end

    X = Lambda\Y;
    xplot=a:0.01:x(1);
    p_0 = X(1) + X(2)*(xplot-a);

    figure(3)
    plot(xplot,p_0)
    hold on

    for i=(1:n-1)
        xplot=x(i):0.01:x(i+1);
        p=X(4*(i-1)+3)+X(4*(i-1)+4)*(xplot-x(i))+X(4*(i-1)+5)*(xplot-x(i)).^2
        +X(4*(i-1)+6)*(xplot-x(i)).^3;
        plot(xplot,p)
        hold on
    end

    xplot=x(n):0.01:b;

    p_n = X(4*n-1) + X(4*n)*(xplot-x(n));
    plot(xplot,p_n)
    hold on

    plot(x,y,'red*')
    title('The approximated function with optimal \lambda')
    xlabel('x')
    ylabel('f(x)')

end
```